TECHNISCHE UNIVERSITÄT WIEN

DISSERTATION

Distribution Recovery in Probabilistic Loops

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften unter der Leitung von

Univ. Prof. Ph.D Efstathia Bura
E105-08 – Institut für Stochastik und Wirtschaftsmathematik, TU Wien

und

Univ. Prof. Dott. Ric. Ezio Bartocci
E191-01 – Institut für Computer Engineering, TU Wien

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation

von

Andrey Kofnov
[redacted]

Diese Dissertation haben begutachtet:

1. **Univ. Prof., Ph.D. Efstathia Bura**
   Institut für Stochastik und Wirtschaftsmathematik, Technische Universität Wien, Österreich
2. **Univ. Prof., Dott. Ric. Ezio Bartocci**
   Institut für Computer Engineering, Technische Universität Wien, Österreich
3. **Prof., Ph.D. Mirco Tribastone**
   Systems Security, Modeling and Analysis, IMT School for Advanced Studies Lucca, Italy
4. **Prof., Ph.D. Max Tschaikowski**
   Department of Computer Science, Aalborg University, Denmark

Wien, am 3. April 2025

Kurzfassung

Diese kumulative Dissertation behandelt die Wiederherstellung von Verteilungen in probabilistischen Schleifen (Kapitel 2 & 3) sowie die Berechnung von oberen und unteren Schranken der kumulativen Verteilungsfunktion für die Ausgabe neuronaler Netze mit zufälligen Eingaben (Kapitel 4).

Viele stochastische dynamische Systeme mit kontinuierlichem Zustandsraum lassen sich als probabilistische Programme mit nichtlinearen, nicht-polynomialen Aktualisierungen in nicht verschachtelten Schleifen modellieren. Wir präsentieren zwei Methoden – eine approximative und eine exakte – zur automatischen Berechnung von momentenbasierten Invarianten für solche probabilistischen Programme in geschlossener Form als Funktion der Schleifeniteration, ohne auf Stichproben zurückzugreifen. Die exakte Methode ist für probabilistische Programme mit trigonometrischen und exponentiellen Aktualisierungen anwendbar und in das Tool Polar eingebettet. Die approximative Methode zur Momentenberechnung ist für beliebige nichtlineare Zufallsfunktionen geeignet, da sie die Theorie der polynomialen Chaos-Entwicklung nutzt, um nicht-polynomiale Aktualisierungen durch eine Summe orthogonaler Polynome zu approximieren. Dadurch wird das dynamische System in eine nicht-verschachtelte Schleife mit polynomialen Aktualisierungen überführt und somit mit dem Polar-Tool kompatibel, das die Momente beliebiger Ordnung der Zustandsvariablen berechnet. Wir evaluieren unsere Methoden anhand zahlreicher Beispiele, die von der Modellierung der Geldpolitik bis hin zu physikalischen Bewegungssystemen in unsicheren Umgebungen reichen. Die experimentellen Ergebnisse belegen die Vorteile unseres Ansatzes im Vergleich zum aktuellen Stand der Technik.

In Kapitel 3 stellen wir die K-Serien-Methode vor, um die Verteilung aller Zufallsvariablen, die in jeder Iteration einer probabilistischen Schleife erzeugt werden, aus ihren Momenten abzuleiten. Diese Methode ist direkt anwendbar auf die probabilistische Analyse von Systemen, die als probabilistische Schleifen dargestellt werden können, also auf Algorithmen, die nichtdeterministische Prozesse aus Bereichen wie Robotik, Makroökonomie, Biologie sowie Software- und cyber-physikalische Systeme ausdrücken und implementieren. Die K-Serien-Methode approximiert statisch die gemeinsamen und marginalen Verteilungen eines Vektors kontinuierlicher Zufallsvariablen, die in einer probabilistischen, nicht-verschachtelten Schleife mit nichtlinearen Zuweisungen aktualisiert werden, unter der Annahme einer endlichen Anzahl von Momenten der unbekannten Dichte. Darüber hinaus leitet K-Serien die Verteilung der Zufallsvariablen eines Systems symbolisch als Funktion der Schleifeniteration her. Die Dichteschätzungen mittels K-Serien sind präzise, effizient und schnell berechenbar. Wir demonstrieren die Anwendbarkeit und Leistungsfähigkeit unseres Ansatzes anhand mehrerer Benchmark-Beispiele aus der Fachliteratur.

Das Problem der Schätzung der Verteilung der Ausgabe eines neuronalen Netzwerks (NN), wenn der Input zufällig gestört wird, wird in Kapitel 4 behandelt. Dort leiten wir exakte obere und untere Schranken für die kumulative Verteilungsfunktion (CDF) der Ausgabe eines NN über dessen gesamten Definitionsbereich ab, wobei stochastische (rauschende) Eingaben berücksichtigt werden. Die oberen und unteren Schranken konvergieren mit zunehmender Auflösung zur tatsächlichen cdf

über ihrem Definitionsbereich.

Unsere Methode gilt für jedes Feedforward-NN, das kontinuierliche, monoton wachsende, stückweise zweimal stetig differenzierbare Aktivierungsfunktionen verwendet (z. B. ReLU, tanh und softmax), sowie für konvolutionale NNs, die über den Geltungsbereich konkurrierender Ansätze hinausgehen. Die Neuheit und das zentrale Werkzeug unserer Methode besteht darin, allgemeine NNs mit ReLU-NNs zu beschränken. Die auf ReLU-NNs basierenden Schranken werden dann verwendet, um die oberen und unteren Schranken der CDF der NN-Ausgabe abzuleiten.

Experimente zeigen, dass unsere Methode garantierte Schranken für die Vorhersage der Ausgabeverteilung über deren Definitionsbereich liefert und somit exakte Fehlergrenzen bietet, im Gegensatz zu konkurrierenden Ansätzen.

Abstract

This is a cumulative thesis on distribution recovery in probabilistic loops (Ch. 2 & 3) and the computation of upper and lower bounds of the cumulative distribution function for the output of neural networks with random inputs (Ch. 4).

Many stochastic continuous-state dynamical systems can be modeled as probabilistic programs with nonlinear non-polynomial updates in non-nested loops. We present two methods, one approximate and one exact, to automatically compute, without sampling, moment-based invariants for such probabilistic programs as closed-form solutions parameterized by the loop iteration. The exact method applies to probabilistic programs with trigonometric and exponential updates and is embedded in the Polar tool. The approximate method for moment computation applies to any nonlinear random function as it exploits the theory of polynomial chaos expansion to approximate non-polynomial updates as the sum of orthogonal polynomials. This translates the dynamical system to a non-nested loop with polynomial updates, and thus renders it conformable with the Polar tool that computes the moments of any order of the state variables. We evaluate our methods on many examples ranging from modeling monetary policy to several physical motion systems in uncertain environments. The experimental results demonstrate the advantages of our approach as compared with the current state-of-the-art.

In Chapter 3 we propose the K-series method to derive the distribution of all random variables generated at each iteration in a probabilistic loop from their moments. It is directly applicable to the probabilistic analysis of systems that can be represented as probabilistic loops; i.e., algorithms that express and implement non-deterministic processes ranging from robotics to macroeconomics and biology to software and cyber-physical systems. K-series statically approximates the joint and marginal distributions of a vector of continuous random variables updated in a probabilistic non-nested loop with nonlinear assignments given a finite number of moments of the unknown density. Moreover, K-series automatically derives the distribution of the systems' random variables symbolically as a function of the loop iteration. K-series density estimates are accurate, easy and fast to compute. We demonstrate the feasibility and performance of our approach on multiple benchmark examples from the literature.

The problem of estimating the distribution of the output of a neural network (NN) when the input is randomly perturbed is considered in Chapter 4. There we derive exact upper and lower bounds for the cumulative distribution function (cdf) of the output of a NN over its entire support subject to noisy (stochastic) inputs. The upper and lower bounds converge to the true cdf over its domain as the resolution increases. Our method applies to any feedforward NN using continuous monotonic piecewise twice continuously differentiable activation functions (e.g., ReLU, tanh and softmax) and convolutional NNs, which were beyond the scope of competing approaches. The novelty and instrumental tool of our approach is to bound general NNs with ReLU NNs. The ReLU NN-based bounds are then used to derive the upper and lower bounds of the cdf of the NN output. Experiments demonstrate that our method delivers guaranteed bounds of the predictive output distribution over its support, thus providing exact error guarantees, in contrast to competing approaches.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 3. April 2025

Andrey Kofnov

Contents

1 Introduction

1.1 Overview

The field of probabilistic programming has evolved as a critical component of modern statistical modeling, particularly in artificial intelligence, machine learning, and stochastic dynamical systems. Probabilistic programs (PPs) enable the modeling of systems with inherent randomness by allowing variables to be treated as random rather than deterministic. This has led to their widespread application in security protocols, automated reasoning, and inference problems.

Probabilistic programs are typically understood in two distinct ways:

- **Randomized Algorithms** – Programs designed to be executed, leveraging randomness to enhance performance, in particular runtime [MR95]. For instance, Monte Carlo methods, which trade computational efficiency for precision, or sorting algorithms like Randomized Quicksort, which achieves an expected worst-case time complexity of $\mathcal{O}(n \log n)$, significantly improving upon the $\mathcal{O}(n^2)$ worst-case of deterministic Quicksort, are among the most popular randomized algorithms. Randomness is also utilized in cryptographic protocols, consensus mechanisms, and load-balancing strategies to improve security, resilience, and efficiency.

- **Probabilistic (Generative) Models** – Programs that define and manipulate probability distributions to model uncertain phenomena. These models provide a structured way to represent and reason about randomness in complex systems. They are commonly expressed using probabilistic graphical models (PGMs) or formulated as mathematical equations governing stochastic relationships. Bayesian networks (BNs), a widely used class of PGMs, capture dependencies between variables through directed acyclic graphs, offering a compact and interpretable representation of conditional probabilities. However, a fundamental challenge in utilizing such models, especially Bayesian networks, lies in performing inference—extracting meaningful conclusions from observed data. This inference process often involves computationally intensive tasks, such as marginalization and evidence propagation, which become increasingly difficult as the complexity of the model grows.

Rapidly increasing reliance on AI and machine learning has led to the adoption of probabilistic modeling techniques [Gha15]. Instead of relying on rigid mathematical formulations, many researchers and practitioners now use probabilistic programming frameworks, which offer a flexible yet rigorous approach to describing stochastic processes. These frameworks excel in representing generative probabilistic models, wherein programs encode the mechanisms responsible for generating probability distributions, making them particularly useful for synthesizing new data samples or simulating uncertain environments. By leveraging probabilistic programming as a modeling tool, researchers can gain deeper insights into underlying distributions, streamline inference procedures,

and refine model parameters for improved accuracy. Furthermore, the integration of probabilistic programming with deep learning—through Bayesian deep learning and models capable of quantifying uncertainty—has opened new avenues for its application. This synergy has proven especially valuable in critical domains such as medical diagnostics, financial risk assessment, and autonomous decision-making, where uncertainty estimation is essential for reliable and interpretable predictions.

A special category of probabilistic programs includes (in)finite probabilistic loops (PLs), which define iterative processes where the number of iterations is either finite or potentially unbounded, depending on probabilistic conditions. These loops introduce additional complexity in reasoning about program behavior, as the execution flow depends on stochastic transitions, potentially leading to undecidable termination conditions. Finite probabilistic loops are often used in randomized algorithms and statistical simulations, whereas infinite loops appear in models describing long-running or perpetual stochastic processes, such as queuing systems and Markov decision processes. The analysis of such loops requires specialized techniques to determine loop termination probabilities, expected runtimes, and steady-state distributions.

One of the challenges in probabilistic loop analysis is understanding the behavior of infinite loops, where random variables evolve iteratively. Historically, researchers have tackled this challenge through the development of moment-based invariants and probabilistic loop analysis techniques. Early methods focused on polynomial updates in loops, leading to the introduction of Prob-solvable loops [BKS19], a subclass of probabilistic programs that allow for the automatic computation of statistical moments as closed-form expressions. However, these methods struggled to handle non-polynomial updates, such as trigonometric and exponential functions, which are essential in various real-world applications.

Significant efforts have been made to improve the analysis of finite PLs, particularly those with non-polynomial updates. Some approaches extend existing moment-based methods to handle more complex functional dependencies within loop iterations [SCG+20; JWW21]. Techniques that approximate or transform these non-polynomial updates into analytically tractable forms have played a crucial role in making probabilistic analysis more practical and broadly applicable.

By refining these analytical techniques, probabilistic programming has become an essential tool for reasoning about uncertainty in diverse domains. From robotic motion planning to cyber-physical systems and cryptographic security, probabilistic programs enable more accurate predictions, uncertainty quantification, and improved decision-making processes.

Despite these advances, several research gaps remain:

a) *Handling non-polynomial updates in unbounded loops*: existing methods often struggle with loops that involve trigonometric, exponential, or logarithmic transformations of state variables, requiring new ways to represent and analyze such updates efficiently.

b) *Accuracy of moment computation*: While some techniques enable moment computation, many either sacrifice accuracy for computational feasibility or lack generality when handling arbitrary non-polynomial updates [SCG+20].

c) *Distribution recovery in (unbounded) loops*: Although many approaches focus on moment computation, a significant challenge remains in reconstructing full probability distributions from computed moments, which is crucial for accurate probabilistic inference and real-world

decision-making applications. Furthermore, existing methods that address this issue primarily focus on finite loops, leaving the analysis of infinite probabilistic loops largely unexplored [GMV16; GSV20; RBI+24].

By addressing these gaps, the field of probabilistic programming can continue to evolve, providing robust solutions for increasingly complex systems that demand precise and scalable stochastic analysis.

1.2 Problem statement

The central challenge addressed in this thesis is the accurate reconstruction of probability distributions in unbounded probabilistic loops with general functional self-updates. Existing approaches either compute exact moments for a restricted class of functions (such as polynomials [BKS19]) or approximate moments using general-purpose polynomial expansions that are applicable only to a limited class of smooth functions (e.g., Taylor series [MB03; SCG+20]). Additionally, many existing techniques estimate distributions of loop variables based on discrete input perturbations or polynomial self-updates, limiting their applicability to more general functional forms. To the best of our knowledge, no prior work has systematically addressed the analysis of unbounded loops with arbitrary self-updates.

This thesis aims to establish a unified framework that facilitates both exact and approximate moment derivation for a broad class of functional forms beyond polynomials, with a particular focus on continuous input perturbations. Furthermore, it introduces a methodology for recovering probability density functions from computed moments, applicable to both finite and infinite loops, significantly broadening the scope of probabilistic program analysis.

The ability to automatically compute moments and recover probability distributions has profound implications in:

- **AI and Machine Learning**: Enhancing probabilistic inference techniques in Bayesian networks [BKS20b].
- **Cyber-Physical Systems**: Providing reliability and predictability in autonomous robotics and distributed systems over the in(finite) time horizon [SCG+20; KMS+22b; KMS+24; JWW21].
- **Financial Modeling**: Improving the accuracy of economic and risk assessment models [KMS+22b; KMS+24].

To demonstrate the translation of real-world scenarios into the probabilistic program form, let us consider an example of continuous-time differential equation. *The Ornstein-Uhlenbeck (OU)* process [UO30] is a stochastic process that models mean-reverting behavior, commonly used in finance, physics, and neuroscience. It is governed by the stochastic differential equation (SDE):

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t, \quad \theta > 0, \quad \sigma \geq 0, \tag{1.1}$$

where, X - is the state variable (e.g., interest rate, stock price deviation, or velocity of a massive Brownian particle), θ is the rate of mean reversion, μ is the long-term mean, σ is the volatility,

```
# Parameters
θ = 0.7   # Mean reversion speed
μ = 0.5   # Long-term mean level
σ = 0.2   # Volatility
τ = 0.2     # Discretization step

# Initial value
X = 0.0  # Value of X_0

# Probabilistic loop for the exact OU process
while true:
    ω = Normal(0, 1 - e^{-2 * θ * τ}) # Brownian motion increment
    α = e^{-θ * τ}
    β = σ / (2 * θ)^0.5
    X = α * X + (1 - α) * μ + β * ω
    # The last equation models the dependence of the incremented
        discretized state variable X_{τ(t+1)} on its previous value X_{τt}
end
```

Figure 1.1: A probabilistic loop modeling the original OU process

which measures the strength of randomness, and W is a Wiener process (Brownian motion). A solution to Equation 1.1 is well known and can be expressed as:

$$X_t = X_0 e^{-\theta t} + \mu(1 - e^{-\theta t}) + \frac{\sigma}{\sqrt{2\theta}} W_{1-e^{-2\theta t}}.$$

If one chooses the discretization rate $\tau > 0$ which is the size of incremental time step and sets initial parameters, one can simulate the behavior of X as a state variable in a probabilistic loop shown in Fig.1.1.

Unfortunately, most stochastic differential equations, unlike the OU process, do not have closed-form solutions. Therefore, we can apply the Euler-Maruyama scheme to discretize the process and find an approximate representation. The corresponding probabilistic loop is shown in Fig. 1.2.

The goal here is to study the statistical properties of the state variable X at any iteration, where the loop variable is implicitly defined as $n = \tau \cdot t$. This translates to computing moments of a given order for specific (or all) iterations, as well as to estimating the distribution of the state variable.

For the Ornstein-Uhlenbeck process, the first part of the problem is addressed by the method outlined in [BKS19], where the authors define a category of probabilistic programs called Prob-solvable loops. For these loops, moment-based invariants of the program's state variables are automatically derived as closed-form expressions. Essentially, a Prob-solvable loop features an initialization phase followed by a single-level loop where variables are updated by sampling from distributions defined via their moments (for example, Bernoulli or Normal) and processed using polynomial arithmetic.

In contrast, capturing more complex dynamics often demands **non-polynomial updates**, as demonstrated by the turning vehicle example in Fig. 1.3. This raises an open research question: How can the framework of Prob-solvable loops be extended to compute moment-based invariants in closed form when the loop updates involve non-polynomial, nonlinear functions?

```
# Parameters
θ = 0.7   # Mean reversion speed
μ = 0.5   # Long-term mean level
σ = 0.2   # Volatility
dt = 0.2    # Discretization step

# Initial value
X = 0.0  # Value of X_0

# Probabilistic loop for the approximated OU process
while true:
    ω = Normal(0, dt^{0.5}) # Brownian motion increment
    X = X + θ * (μ - X) * dt + σ * ω
end
```

Figure 1.2: A probabilistic loop modeling the OU process after the Euler-Maruyama discretization

The turning vehicle model was introduced in [SCG+20] and depicts the position of a moving vehicle. The state variables are (x, y, v, ψ), where (x, y) is the vehicle's position with velocity v and yaw angle ψ. The vehicle's velocity is stabilized around $v_0 = 10$ m/s. The dynamics are modeled by the equations $x(t+1) = x(t) + \tau v \cos(\psi(t))$, $y(t+1) = y(t) + \tau v \sin(\psi(t))$, $v(t+1) = v(t) + \tau(K(v(t) - v_0) + w_1(t+1))$, and $\psi(t+1) = \psi(t) + w_2(t+1)$. The mean reversion parameter $K = -0.5$. The disturbances w_1 and w_2 have distributions $w_1 \sim U[-0.1, 0.1]$, $w_2 \sim N(0, 0.01)$. Initially, the state variables are distributed as: $x(0) \sim U[-0.1, 0.1]$, $y(0) \sim U[-0.5, -0.3]$, $v(0) \sim U[6.5, 8.0]$, $\psi(0) \sim N(0, 0.01)$. We allow all normally distributed parameters take values over the entire real line, in contrast to [SCG+20] who could not accommodate distributions with infinite support and required the normal variables to take values over finite intervals.

In [SCG+20], the authors introduced a method called Polynomial forms, which employed Taylor series expansion and constructed an error bound by manipulating the Lagrange remainder. However, their approach proved to be quite unstable, limiting its use to estimating moments only for a few iterations at the beginning of the loop execution. Additionally, the method was restricted to distributions with bounded support, such as the truncated normal, meaning it could not handle distributions defined over the entire real line.

To enable more precise probabilistic inference, one needs to estimate the probability distribution of the state variables inside loops.

In previous works, exact distributions were computed. λPSI is a solver designed to compute exact distributions of probabilistic programs (PPs) as symbolic mathematical expressions, supporting features such as first-class functions, nested inference, and handling discrete, continuous, and mixed random variables [GSV20]. However, its applicability is limited in two significant ways: it is restricted to bounded loops, and the symbolic expressions it produces become extremely complex, making them difficult to compute and implement even for a small number of loop iterations.

More recently, [KBC+23] introduced Prodigy, a method that enables exact inference in loop-free PPs with discrete random states. To extend this capability to potentially infinite while-loops, they attempt to identify classes of while-loop programs that can be treated as equivalent to loop-

```
v0 = 10, τ = 0.1, K = -0.5
ψ = Normal(0, 0.01)
v = Uniform(6.5, 8.0)
x = Uniform(-0.1, 0.1)
y = Uniform(-0.5, -0.3)
while true:
    w1 = Uniform(-0.1, 0.1)
    w2 = Normal(0, 0.01)
    x = x + τ v cos(ψ)
    y = y + τ v sin(ψ)
    v = v + τ (K(v-v0)+w1)
    ψ = ψ + w2
end
```

Figure 1.3: A probabilistic loop modeling the behaviour of a turning vehicle [SCG+20] using non-polynomial (cos, sin) updates in the loop body.

free programs. This approach is applicable to probabilistic programs involving discrete random variables, where the distribution depends on parameters updated within a Bayesian framework.

While both λPSI [GSV20] and Prodigy [KBC+23] offer strong guarantees, they are limited in their scope. Prodigy only works with discrete random variables and loop-free programs (or those that can be transformed into such), while λPSI struggles to handle even modestly complex expressions, becoming practically infeasible after just a few iterations. Neither tool offers broad practical applicability.

The most recent paper, [RBI+24], proposes moment-based density approximations using a Gaussian mixture, but is limited to bounded loops and only supports polynomial updates. None of the methods mentioned above provides a probability distribution estimation for state variables within **infinite loops** with non-polynomial updates.

1.3 Research goals

This research tackles the challenge of reconstructing probability distributions in unbounded probabilistic loops with general functional self-updates. Existing methods either compute exact moments for a restricted class of functions (such as polynomials) or approximate moments using polynomial expansions with limited applicability. Additionally, current approaches focus on computing exact or estimated probability distributions only in loop-free programs or bounded loops. This work extends moment-based probabilistic program analysis to efficiently handle non-polynomial updates in infinite loops, addressing key limitations in the field. This research focuses on probabilistic loops with non-polynomial updates and continuous input distributions, addressing challenges in both exact and approximate moment computation as well as density estimation. Our analysis is static and does not rely solely on sampling-based inference techniques.

The primary objectives are:

1. Develop an exact methodology for computing moments in loops with trigonometric and exponential updates.

2. Propose an alternative to existing techniques (e.g., Taylor series expansion) for approximating wide class of non-polynomial functions, enabling their analysis in unbounded loops.

3. Introduce a moment-based approach to estimate probability distributions in probabilistic loops for any iteration.

4. Characterize theoretical properties of the proposed methods.

5. Compare and validate the proposed methods against existing moment evaluation techniques.

6. Apply the developed methodologies to benchmark models in motion planning, cyber-physical systems, and financial or economic modeling.

While some problems outside our primary scope can be transformed into a form compatible with our approach, this study does not aim to address arbitrary stochastic processes beyond loop-based structures, nor does it consider purely discrete or mixed input perturbations.

1.4 Impactful contributions

This work introduces two approaches for moment computation in probabilistic loops:

1) The *approximate* method based on general polynomial chaos expansion (gPCE) [XK02], which enables the high-precision approximation of non-polynomial updates, thereby extending the applicability of Prob-solvable loops. This approach decomposes a random function into a linear combination of orthogonal polynomial basis functions. By expressing the non-linear functionals of the updates as sums of orthogonal polynomials, we can perform an automatic closed-form computation of moments at each loop iteration (n) using the methodology proposed in [BKS19; BKS21].

2) The *exact* moment computation method for loops with trigonometric and exponential updates, utilizing advancements in algebraic and probabilistic techniques. [JWW21] developed a method that obtains the exact time evolution of the moments of random states for a class of dynamical systems that depend on trigonometric updates. We amended their approach to make it compatible with the `Polar` tool [MSB+22]. Specifically, we incorporated the approach of [JWW21] into *Prob-solvable loops* when updates involve trigonometric functions. This allows us to automatically compute the *exact* moments of any order and at all iterations. Moreover, we extended [JWW21] to include *exponential* updates.

We also proposed *K-series estimation*, a density estimation method that recovers the probability density function (pdf) of bounded support from a finite set of moments of a random vector $\mathbf{X} = (X_1, \ldots, X_k)^T$, $m_{i_1,i_2,\ldots,i_k} = \mathbb{E}\left(X_1^{i_1} X_2^{i_2} \cdot \ldots \cdot X_k^{i_k}\right)$, $0 \leq i_j \leq d_j$, $d_j \in \mathbb{N}$, $j = 1, \ldots, k$, using a basis of orthogonal polynomials that target the unknown density by choosing the reference pdf. Our approach is general in that it allows for any continuous reference distribution, whose effect is incorporated in the construction of the orthogonal polynomials via the Gram-Schmidt orthogonalization procedure and can be tailored to improve the accuracy of the estimation.

Moreover, the contributions of this thesis in probabilistic program analysis are to

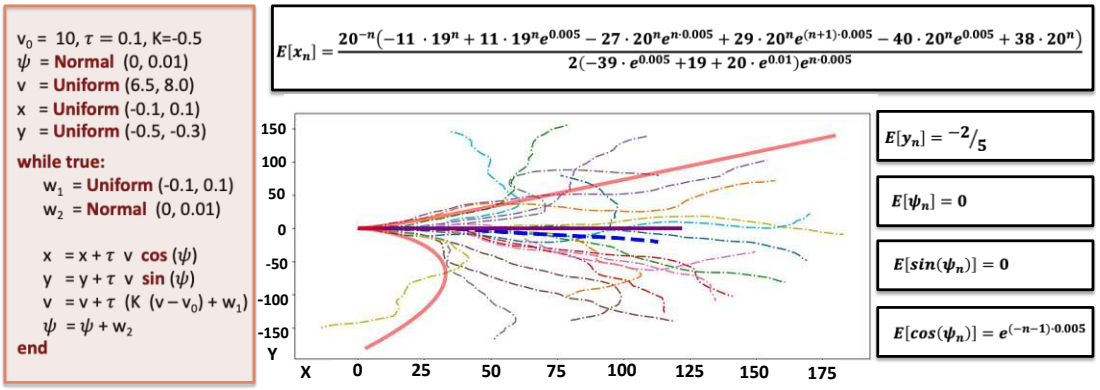


Figure 1.4: On the left is a probabilistic loop modeling the behavior of a turning vehicle [SCG+20] using non-polynomial (cos, sin) updates in the loop body. On the top right is the exact expected position (x_n, y_n) and other exact expected values computed automatically in closed-form in the number of loop iterations n. The plot in the center contains 20 sampled trajectories (x_n, y_n) up to iteration $n = 201$ (dashdot lines with different colors) and the approximated expected trajectory computed by averaging the sampled ones (dashed blue line). Moreover, we automatically computed the exact expected trajectory and standard deviation with our method. The solid purple line marks the exact expected trajectory. The two solid red lines mark the boundary of the region contained within $\pm$ two standard deviations of the expected trajectory.

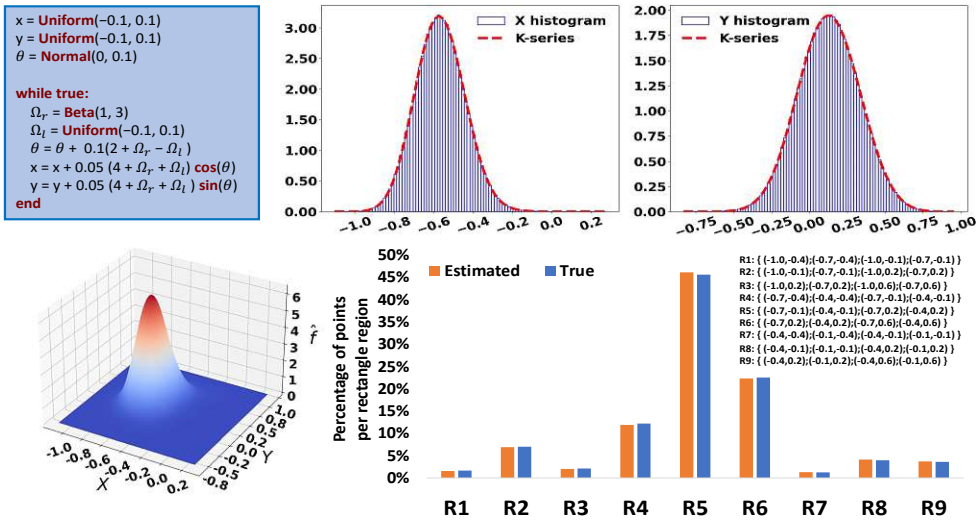


Figure 1.5: Probabilistic loop with non-polynomial assignment for the Differential-Drive Mobile Robot [JWW21] (top left), the approximations of the marginal distributions with K-series (top right), the approximation of the joint distribution with K-series (bottom left) and comparison with true histogram (bottom right).

(a) Demonstrate the use of Polynomial Chaos Expansion in moment-based inference in infinite probabilistic loops.

(b) Provide formal guarantees on convergence properties of the gPCE estimator to the underlying function in L_1 sence.

(c) Adapt the mathematical framework for estimating the distribution of a random variable, proposed by [FMS13], to the setting of probabilistic loops where multiple random variables are generated at each iteration.

(d) Based on this framework, we introduce K-series: the first method to automatically derive the distributions of multiple state variables in probabilistic programs, such as *prob-solvable* loops [BKS19; MSB+22], symbolically and for any number of iterations.

(e) Derive the theoretical foundation in Proposition 2 and Theorem 2, where we prove that other methods, such as the Gram-Charlier (GC) expansion [Cra57; KS77; Hal00] and Method of Moments [MMR17], are special cases of our general approach.

(f) Obtain the convergence rate of our density estimator to the true pdf in Theorem 3.

(g) Show that K-series is an accurate estimator of the unknown true pdf by proving the moment matching principle of K-series in Theorem 4; that is, we show that the first n moments of the true pdf and the K-series estimator are the same.

(h) Derive the approximation to the true support of the target pdf.

1.5 Thesis structure.

This thesis is based on three peer-reviewed publications and one that is submitted. The corresponding publications are listed at the beginning of each chapter. Each chapter is designed to be self-contained, introducing its notation, preliminaries, and related work. Nevertheless, the logical flow in connecting the chapters is preserved.

Chapter 2. Moment-based Invariants for Probabilistic Loops. Many stochastic dynamical systems with continuous states can be represented as probabilistic programs featuring nonlinear, non-polynomial updates in non-nested loops. In this chapter we introduce two complementary methods—one exact and one approximate—to compute moment-based invariants for such programs without relying on sampling. The exact method, embedded in the Polar tool, handles probabilistic programs with trigonometric and exponential updates. The approximate method employs polynomial chaos expansion to transform non-polynomial updates into sums of orthogonal polynomials, enabling moment computation using Polar. This allows us to compute moments of any order for state variables in a wide range of nonlinear systems. We evaluate our approach on diverse examples, including economic models and physical motion systems, demonstrating its effectiveness compared to existing techniques.

We replace Theorem 4.1 from [KMS+24] (which corresponds to Theorem 1 in [KMS+22b]) with Theorem 1, which is more general and shows the universality of our method.

Chapter 3. The K-series Approach to Moment-based Density Elicitation in Probabilistic Loops. We introduce the K-series estimation method for recovering unknown univariate and multivariate distributions using a finite set of moment constraints. This method is particularly suited for probabilistic analysis of systems modeled as probabilistic loops, encompassing applications in robotics, macroeconomics, biology, software, and cyber-physical systems. K-series

provides a static approximation of joint and marginal distributions for continuous random variables evolving within a probabilistic non-nested loop with nonlinear updates. Given a finite number of known moments, it derives symbolic expressions for the distributions as functions of the loop iteration. The approach is both computationally efficient and highly accurate in estimating probability densities. We validate its effectiveness through multiple benchmark examples from the literature.

Chapter 4. Exact Upper and Lower Bounds for the Output Distribution of Neural Networks with Random Inputs. In this chapter we establish exact upper and lower bounds for the cumulative distribution function (cdf) of a neural network's output across its entire support when subjected to stochastic (noisy) inputs. These bounds progressively converge to the true cdf as the resolution increases. Our approach applies to any feedforward neural network with continuous, monotonic, and piecewise twice continuously differentiable activation functions, such as ReLU, tanh, and softmax, as well as convolutional neural networks—extending beyond the capabilities of existing methods. A key innovation of our work is the use of ReLU neural networks to bound general neural networks, enabling the derivation of tight cdf bounds. Experimental results demonstrate that our method provides guaranteed bounds on the predictive output distribution, offering exact error guarantees that competing approaches lack.

Chapter 5. Summary and Future Perspectives. In this chapter, we summarize the thesis and explore potential directions for future research based on our findings.

1.6 Included publications.

Peer-reviewed publications.

[KMS+22b] Andrey Kofnov, Marcel Moosbrugger, Miroslav Stankovič, Ezio Bartocci, and Efstathia Bura. *Moment-based invariants for probabilistic loops with non-polynomial assignments.* In E. Ábrahám and M. Paolieri, editors, Quantitative Evaluation of Systems, pages 3–25, Cham. Springer International Publishing, 2022. DOI: 10.1007/978-3-031-16336-4_1

[KMS+24] Andrey Kofnov, Marcel Moosbrugger, Miroslav Stankovič, Ezio Bartocci, and Efstathia Bura. 2024. *Exact and Approximate Moment Derivation for Probabilistic Loops With Non-Polynomial Assignments.* *ACM Trans. Model. Comput. Simul.* 34, 3, Article 18 (July 2024), 25 pages. DOI: 10.1145/3641545

[KBB25] Andrey Kofnov, Ezio Bartocci, and Efstathia Bura. 2025. *Moment-based Density Elicitation with Applications in Probabilistic Loops.* Accepted for publication in *ACM Trans. Probab. Mach. Learn.*

Preprints under review.

[KKB+25] Andrey Kofnov, Daniel Kapla, Ezio Bartocci, and Efstathia Bura. *Exact Upper and Lower Bounds for the Output Distribution of Neural Networks with Random Inputs.* DOI: 10.48550/arXiv.2502.11672

In all four papers, Andrey Kofnov was not only the first author but also the lead contributor both in terms of ideas and motivation as well as theory and method development.

Distinctions and Recognitions. The publication [KMS+22b] was honored with the **Best Paper Award** at the 2022 International Conference on Quantitative Evaluation of Systems.

2 Moment-based Invariants for Probabilistic Loops

This chapter is based on the following publications: [KMS+22b; KMS+24]

- *Andrey Kofnov, Marcel Moosbrugger, Miroslav Stankovič, Ezio Bartocci, and Efstathia Bura. Moment-based invariants for probabilistic loops with non-polynomial assignments. In E. Ábrahám and M. Paolieri, editors, Quantitative Evaluation of Systems, pages 3–25, Cham. Springer International Publishing, 2022.*
- *Andrey Kofnov, Marcel Moosbrugger, Miroslav Stankovič, Ezio Bartocci, and Efstathia Bura. 2024. Exact and Approximate Moment Derivation for Probabilistic Loops With Non-Polynomial Assignments. *ACM Trans. Model. Comput. Simul.* 34, 3, Article 18 (July 2024), 25 pages.*

2.1 Preliminaries

We assume the reader is familiar with basic probability theory. For more details, we refer to [Dur19].

2.1.1 Prob-Solvable Loops

[BKS19] defined the class of *Prob-solvable loops* for which moments of all orders of program variables can be computed symbolically: given a Prob-solvable loop and a program variable x, their method computes a closed-form solution for $\mathbb{E}(x_n^k)$ for arbitrary $k \in \mathbb{N}$, where n denotes the nth loop iteration. Prob-solvable loops are restricted to polynomial variable updates.

Definition 1 (Prob-solvable loops [BKS19])**.** *Let $m \in \mathbb{N}$ and $x_1, \dots x_m$ denote real-valued program variables. A Prob-solvable loop with program variables $x_1, \dots x_m$ is a loop of the form*

$$I;\ \texttt{while true:}\ \ U\ \ \texttt{end},\quad \textit{such that}$$

- *I is a sequence of initial assignments over a subset of $\{x_1, \dots, x_m\}$. The initial values of x_i can be drawn from a known distribution. They can also be real constants.*
- *U is the loop body and a sequence of m random updates, each of the form,*

$$x_i = \textit{Dist} \quad \textit{or} \quad x_i = a_i x_i + P_i(x_1, \dots x_{i-1}),$$

where $a_i \in \mathbb{R}$, $P_i \in \mathbb{R}[x_1, \dots, x_{i-1}]$ is a polynomial over program variables $x_1, \dots, x_{i-1}$ and Dist is a random variable whose distribution is independent of program variables with computable moments. a_i and the coefficients in P_i can be random variables with the same constraints as for Dist.

The syntax of Prob-solvable loops as defined in Definition 1 is restrictive. For instance, an assignment for a variable x_i must not reference variables x_j with $j > i$. Hence, the structural dependencies among program variables are acyclic. Some of these syntactical restrictions were lifted in a later work [MSB+22] to support distributions depending on program variables, if-statements, and linear cyclic dependencies. The latter means that polynomial assignments can be of the form $x_i = L_i(x_1, \ldots, x_n) + P_i(x_1, \ldots x_{i-1})$, where P_i is a polynomial, and L_i is a linear function, as long as all program variables in $L_i(x_1, \ldots, x_n)$ with non-zero coefficient depend only linearly on x_i. In this work, we utilize this relaxation and allow for linear cyclic dependencies in Prob-solvable loops.

Many real-life systems exhibit non-polynomial dynamics and require more general updates, such as, for example, trigonometric or exponential functions. We develop two methods – one approximate, one exact – that allow the modeling of non-polynomial assignments in probabilistic loops by polynomial assignments. Doing so allows us to use the *Prob-solvable loop* based methods in [BKS19; MSB+22] to compute the moments of the stochastic components of a much broader class of systems. Our method for exact moment derivation for probabilistic loops with non-polynomial functions builds upon Prob-solvable loops. In contrast, our PCE-based approach, described in the following sections, is not limited to Prob-solvable and can be used in more general probabilistic loops. The only requirement is that the loops satisfy the conditions in Section 2.2.1.

2.1.2 Polynomial Chaos Expansion

Polynomial chaos expansion (PCE) [EMS+12; XK02] recovers a random variable in terms of a linear combination of functionals whose entries are known random variables, sometimes called germs, or, basic variables. Let $(\Omega, \Sigma, \mathbb{P})$ be a probability space, where Ω is the set of elementary events, Σ is a σ-algebra of subsets of Ω, and $\mathbb{P}$ is a probability measure on Σ. Suppose X is a real-valued random variable defined on $(\Omega, \Sigma, \mathbb{P})$, such that

$$\mathbb{E}(X^2) = \int_\Omega X^2(\omega) d\mathbb{P}(\omega) < \infty. \tag{2.1}$$

The space of all random variables X satisfying (2.1) is denoted by $L_2(\Omega, \Sigma, \mathbb{P})$. The elements of $L_2(\Omega, \Sigma, \mathbb{P})$ are real-valued random variables defined on $(\Omega, \Sigma, \mathbb{P})$ with finite second moments. If we define the inner product as

$$\mathbb{E}(XY) = (X, Y) = \int_\Omega X(\omega) Y(\omega) d\mathbb{P}(\omega) \tag{2.2}$$

and norm

$$||X|| = \sqrt{\mathbb{E}(X^2)} = \sqrt{\int_\Omega X^2(\omega) d\mathbb{P}(\omega)},$$

then $L_2(\Omega, \Sigma, \mathbb{P})$ is a Hilbert space; i.e., an infinite dimensional linear space of functions endowed with an inner product and a distance metric. Elements of a Hilbert space can be uniquely identified by their coordinates with respect to an orthonormal basis of functions, in analogy with Cartesian coordinates in the plane. Convergence with respect to the norm $||\cdot||$ is called *mean-square convergence*. A particularly important feature of a Hilbert space is that when the limit of a sequence of functions exists, it belongs to the space.

The elements in $L_2(\Omega, \Sigma, \mathbb{P})$ can be classified into two groups: *basic* and *generic* random variables, which we want to decompose using the elements of the first set of basic variables. [EMS+12] showed that the basic random variables that can be used in the decomposition of other functions have finite moments of all orders with continuous probability density functions (pdfs).

The σ-algebra generated by the basic random variable Z is denoted by $\sigma(Z)$. Suppose we restrict our attention to decompositions of a random variable $X = g(Z)$, where g is a function with $g(Z) \in L_2(\Omega, \sigma(Z), \mathbb{P})$, and the basic random variable Z determines the class of orthogonal polynomials $\{\phi_i(Z), i \in \mathbb{N}\}$,

$$\langle \phi_i(Z), \phi_j(Z) \rangle = \int_\Omega \phi_i(Z(\omega))\phi_j(Z(\omega))d\mathbb{P}(\omega) = \int \phi_i(x)\phi_j(x)f_Z(x)dx = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \quad (2.3)$$

where f_Z denotes the pdf of Z. The set $\{\phi_i(Z), i \in \mathbb{N}\}$ is a polynomial chaos basis.

Definition 2. *A random variable Z is said to be exponentially integrable[1], if there exists a positive $a > 0$ such that $\int_{\mathbb{R}} \exp\{a|z|\} f_Z(z)dz < \infty$. A random vector $\mathbf{Z}$ is said to be exponentially integrable if there exists a positive $a > 0$ such that $\int_{\mathbb{R}^k} \exp\{a\|\mathbf{Z}\|_{\mathbb{R}^k}\} dF_{\mathbf{Z}}(\mathbf{z})d\mathbf{z} < \infty$ (see [EMS+12; Rah18]).*

A pdf supported on an unbounded set is uniquely identifiable by its moments if and only if it is exponentially integrable [EMS+12]. [2] This encompasses a very broad class, including most widely used densities. However, notable counterexamples are the log-normal and Cauchy distributions.

When a distribution can be uniquely identified by its moments, it is also completely determined by its moments, which enables a succinct and accurate representation of the distribution. Moreover, distribution identification by moments facilitates meaningful comparisons between diverse distributions and streamlines statistical inference procedures. Further, we will consider reference distributions (distributions of basic random variables) that either have compact support or are exponentially integrable.

If Z is normal with mean zero, the Hilbert space $L_2(\Omega, \sigma(Z), \mathbb{P})$ is called *Gaussian* and the related set of polynomials is represented by the family of Hermite polynomials (see, for example, [XK02]) defined on the whole real line. Hermite polynomials form a basis of $L_2(\Omega, \sigma(Z), \mathbb{P})$. Therefore, every random variable X with finite second moment can be approximated by the truncated PCE

$$X^{(d)} = \sum_{i=0}^{d} c_i \phi_i(Z), \quad (2.4)$$

for suitable coefficients c_i that depend on the random variable X. The truncation parameter d is the highest polynomial degree in the expansion. Since the polynomials are orthogonal,

$$c_i = \frac{1}{||\phi_i||^2} \langle X, \phi_i \rangle = \frac{1}{||\phi_i||^2} \langle g, \phi_i \rangle = \frac{1}{||\phi_i||^2} \int_{\mathbb{R}} g(x)\phi_i(x)f_Z(x)dx. \quad (2.5)$$

The truncated PCE of X in (2.4) converges in mean square to X [EMS+12, Section 3.1]. The first two moments of (2.4) are determined by $\mathbb{E}(X^{(d)}) = c_0$, and $\mathbb{V}\mathrm{ar}(X^{(d)}) = \sum_{i=1}^{d} c_i^2 ||\phi_i||^2$.

[1] We use the term "exponentially integrable" interchangeably in the context of a random variable and its distribution (i.e., its pdf).

[2] See also [Bil12, Th. 30.1, p. 388] and [CB01, Th. 2.3.11].

Representing a random variable by a series of Hermite polynomials in a countable sequence of independent Gaussian random variables is known as Wiener–Hermite polynomial chaos expansion. In applications of Wiener–Hermite PCEs, the underlying Gaussian Hilbert space is often taken to be the space spanned by a sequence $\{Z_i, i \in \mathbb{N}\}$ of independent standard Gaussian basic random variables; i.e., $Z_i \sim \mathcal{N}(0,1)$. For computational purposes, the countable sequence $\{Z_i, i \in \mathbb{N}\}$ is restricted to a finite number $k \in \mathbb{N}$ of random variables. The Wiener–Hermite PCE converges for random variables with finite second moment. Specifically, for any random variable $X \in L_2(\Omega, \sigma(\{Z_i, i \in \mathbb{N}\}), \mathbb{P})$, the approximation (2.4) satisfies

$$X_k^{(d)} = \sum_{i=0}^{d} a_i \phi_i(Z_1, \ldots, Z_k) \to X \quad \text{as } d, k \to \infty \tag{2.6}$$

in mean-square convergence (see [EMS+12]). The distribution of X can be quite general; e.g., discrete, singularly continuous, absolutely continuous as well as of mixed type.

2.2 Polynomial Chaos Expansion Algorithm

2.2.1 Random Function Representation

In this section, we state the conditions under which the estimated polynomial is an unbiased and consistent estimator and has exponential convergence rate. Suppose k continuous random variables $Z_1, \ldots, Z_k$ are used to introduce stochasticity in a PP with corresponding cumulative distribution functions (cdfs) F_{Z_i}, $i = 1, \ldots, k$. Also, suppose all k distributions have probability density functions, and let $\mathbf{Z} = (Z_1, \ldots, Z_k)$ with cdf $F_{\mathbf{Z}}$. We assume that the elements of $\mathbf{Z}$ satisfy the following conditions:

(A) Z_i, $i = 1, \ldots, k$, are independent.

(B) We consider functions g such that $g(\mathbf{Z}) \in L_2(\mathcal{Q}, F_{\mathbf{Z}})$, where $\mathcal{Q}$ is the support of the joint distribution of $\mathbf{Z} = (Z_1, \ldots, Z_k)$.[3]

(C) All random variables Z_i have distributions that are uniquely characterized by their moments.[4]

Under condition (A), the joint cdf of the components of $\mathbf{Z}$ is $F_{\mathbf{Z}} = \prod_{i=1}^{k} F_{Z_i}$. To ensure the construction of unbiased estimators with optimal exponential convergence rate (see [XK02], [EMS+12]) in the context of probabilistic loops, we further introduce the following assumptions:

(D) g is a function of a fixed number of basic variables (arguments) over all loop iterations.

(E) If $\mathbf{Z}(j) = (Z_1(j), \ldots, Z_k(j))$ is the stochastic argument of g at iteration j, then $F_{Z_i(j)}(x) = F_{Z_i(l)}(x)$ for all pairs of iterations (j, l) and x in the support of F_{Z_i}.

If Conditions (D) and (E) are not met, then the polynomial coefficients in the PCE need be computed for each loop iteration individually to ensure optimal convergence rate. It is straightforward to show the following proposition.

[3] Ω is dropped from the notation as the sample space is not important in our formulation.
[4] Conditions that ascertain this are given in Theorem 3.4 of [EMS+12].

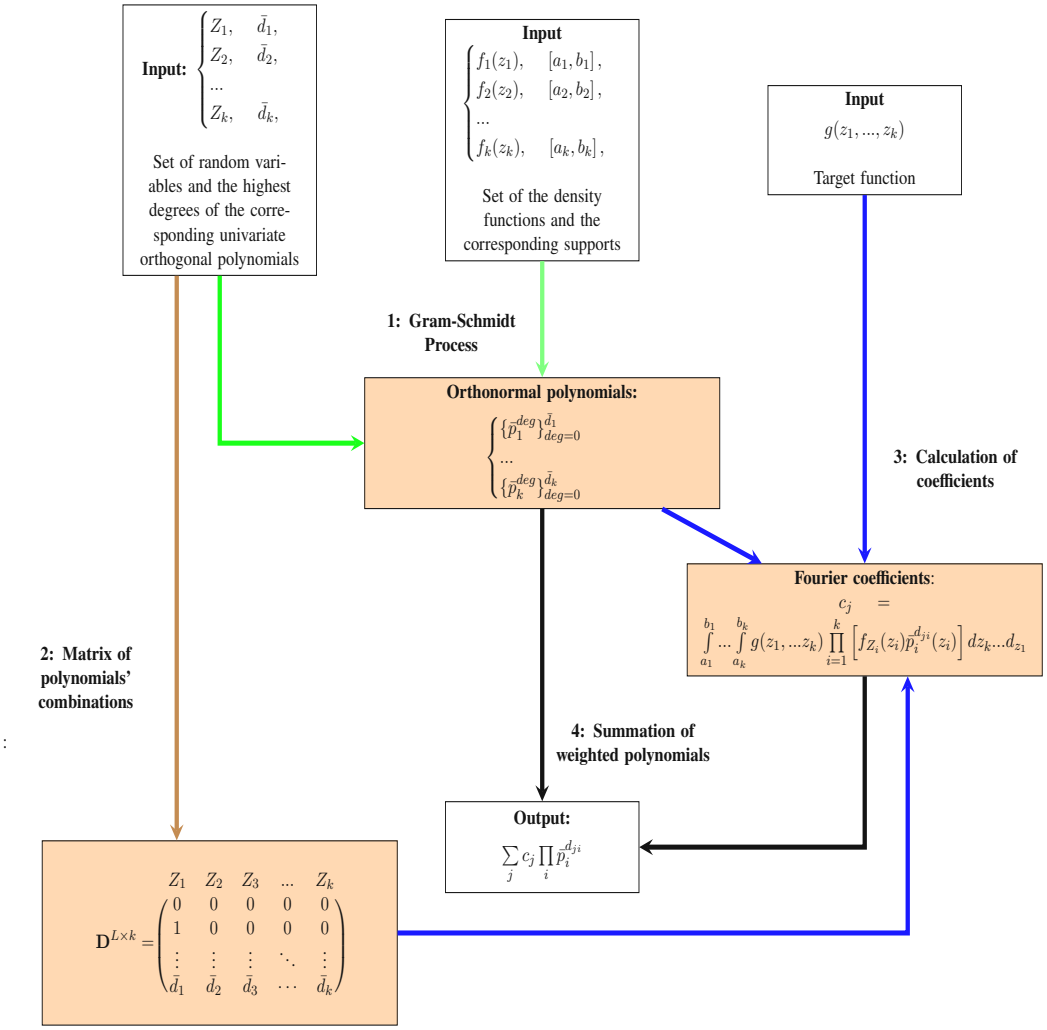


Figure 2.1: Illustration of PCE algorithm

Proposition 1. *If* $\mathbf{Z} = (Z_1, \ldots, Z_{k_1})$, $\mathbf{Y} = (Y_1, \ldots, Y_{k_2})$ *satisfy conditions (A), (C) and (E) and are mutually independent, and functions* g *and* h *satisfy conditions (B) and (D), then their sum,* $g(\mathbf{Z}) + h(\mathbf{Y})$*, and product,* $g(\mathbf{Z}) \cdot h(\mathbf{Y})$*, also satisfy conditions (B) and (D).*

2.2.2 PCE Algorithm

Let $Z_1, \ldots, Z_k$ be independent continuous random variables, with respective cdfs F_i, satisfying conditions (A), (B) and (C). Then, $\mathbf{Z} = (Z_1, \ldots, Z_k)^T$ has cdf $\mathbf{F_Z} = \prod_{i=1}^{k} F_i$. Let $\mathcal{Q}$ denote the support of $\mathbf{F_Z}$. The function $g : \mathbb{R}^k \to \mathbb{R}$, with $g \in L_2(\mathcal{Q}, \mathbf{F})$, can be approximated with the

truncated orthogonal polynomial expansion, as described in Fig. 2.1,

$$g(\mathbf{Z}) \approx \hat{g}(\mathbf{Z}) = \sum_{\substack{d_i \in \{0,\ldots,\bar{d}_i\},\\ i=1,\ldots,k}} c(d_1,\ldots,d_k) z_1^{d_1} \cdots z_k^{d_k} = \sum_{j=1}^{L} c_j \prod_{i=1}^{k} \bar{p}_i^{d_{ji}}(z_i), \tag{2.7}$$

where

- $\bar{p}_i^{d_{ji}}(z_i)$ is a polynomial of degree d_{ji}, and belongs to the set of orthogonal polynomials with respect to F_{Z_i} that are calculated with the Gram-Schmidt orthogonalization procedure[5];
- $\bar{d}_i = \max_j(d_{ji})$ is the highest degree of the univariate orthogonal polynomial, for $i = 1,\ldots,k$;
- $L = \prod_{i=1}^{k}(1+\bar{d}_i)$ is the total number of multivariate orthogonal polynomials and equals the truncation constant;
- c_j are the Fourier coefficients.

The Fourier coefficients are calculated using

$$c_j = \int_{\mathcal{Q}} g(z_1,...,z_k) p_i^{d_{ji}}(z_i) d\mathbf{F} = \int_{\mathcal{Q}} \cdots \int g(z_1,...,z_k) \left(\prod_{i=1}^{k} \bar{p}_i^{d_{ji}}(z_i) \right) dF_{Z_k} ... dF_{Z_1}, \tag{2.8}$$

by Fubini's theorem.

Example 1. *Referring to the Turning vehicle model in Fig. 1.4, the non-polynomial functions to approximate are $g_1 = cos$ and $g_2 = sin$ in the updates of program variables y, x, respectively. In both cases, we only need to consider a single basic random variable, $Z \sim \mathcal{N}(0, 0.01)$ (ψ in Fig. 1.4).*

Using polynomials of degree up to 5, (2.7) *has the following form for the two functions,*

$$\hat{g}_1(z) = cos(\psi) = a_0 + a_1\psi + ... + a_5\psi^5 \tag{2.9}$$

and

$$\hat{g}_2(z) = sin(\psi) = b_0 + b_1\psi + ... + b_5\psi^5. \tag{2.10}$$

We compute the coefficients a_i, b_i in equations (2.9)-(2.10) *using* (2.8) *to obtain the values shown in Fig. 1.4.*

Example 2. *Consider the function $g(x,y) = \sqrt{x^2+y^2}$, where $x \sim Uniform(-1,1)$ and y has pdf $f_y = 0.75(1-y^2)$ supported on $[-1,1]$. Up to degree 2, the basis elements in the PC expansion are element-wise products of the univariate orthogonal polynomials*

$$p_x^0(x) = 1, \quad p_y^0(y) = 1, p_x^1(x) = \sqrt{3}x, \quad p_y^1(y) = \sqrt{5}y,$$
$$p_x^2(x) = 1.5\sqrt{5}x^2 - 0.5\sqrt{5},\ p_y^2(y) = 1.25\sqrt{14}y^2 - 0.25\sqrt{14}.$$

[5] Generalized PCE typically entails using orthogonal basis polynomials specific to the distribution of the basic variables, according to the Askey scheme of [XK02; Xiu10]. We opted for the most general procedure that can be used for any basic variable distribution.

The corresponding PCE polynomial basis elements are

$$p_{xy}^{00}(x,y) = 1, \quad p_{xy}^{02}(x,y) = 1.25\sqrt{14}y^2 - 0.25\sqrt{14}, \quad p_{xy}^{20}(x,y) = 1.5\sqrt{5}x^2 - 0.5\sqrt{5},$$
$$p_{xy}^{22}(x,y) = 1.875\sqrt{70}x^2y^2 - 0.375\sqrt{70}x^2 - 0.625\sqrt{70}y^2 + 0.125\sqrt{70},$$

with corresponding non-zero Fourier coefficients $c_{00} = 0.677408, c_{02} = 0.154109, c_{20} = 0.216390,$ *and* $c_{22} = -0.040153$. *The resulting estimator is*

$$\hat{g}(x,y) = \sum_{(i,j)=(0,0)}^{(2,2)} c_{i,j} p_{xy}^{ij}(x,y) = -0.629900x^2y^2 + 0.851774x^2 + 0.930747y^2 + 0.249327.$$

Complexity. Assuming the expansion is carried out up to the same polynomial degree d for each basic variable, $\bar{d}_i = d, \forall i = 1, ..., k$. This implies $d = \sqrt[k]{L} - 1$. The complexity of the scheme is $\mathcal{O}(sd^2k + s^kd^k)$, where $\mathcal{O}(s)$ is the complexity of computing univariate integrals.

The complexity of our approximation scheme consists of of two parts: (1) the orthogonalization process and (2) the calculation of coefficients. Regarding (1), we orthogonalize and normalize k sets of d basic linearly independent polynomials via the Gram-Schmidt process. For degree d=1, we need to calculate one integral, the inner product with the previous polynomial. Additionally, we need to compute one more integral, the norm of itself (for normalization). For each subsequent degree d', we must calculate d' additional new integrals. The computation of each integral has complexity $\mathcal{O}(s)$. Regarding (2), the computation of the coefficients requires calculating L=$(d$+1$)^k$ integrals with k-variate functions as integrands.

We define the approximation error to be

$$se(\hat{g}) = \sqrt{\int_{\mathcal{Q}} \left(g(z_1, ..., z_k) - \hat{g}(z_1, ..., z_k)\right)^2 dF_{Z_1} \dots dF_{Z_k}} \tag{2.11}$$

since $\mathbb{E}(\hat{g}(Z_1, ..., Z_k)) = g(Z_1, \cdots, Z_k)$ by construction.

The implementation of this algorithm may become challenging when the random functions have complicated forms and the number of parametric uncertainties is large. In this case, the calculation of the PCE coefficients involves high dimensional integration, which may prove difficult and time prohibitive for real-time applications [SD20].

2.3 Prob-Solvable Loops for General Non-Polynomial Functions

PCE allows incorporating non-polynomial updates into Prob-solvable loop programs and use the algorithm in [BKS19] and exact tools, such as Polar [MSB+22], for moment (invariant) computation. We identify two classes of programs based on how the distributions of the generated random variables vary.

2.3.1 Iteration-Stable Distributions of Random Arguments

Let $\mathcal{P}$ be an arbitrary Prob-solvable loop and suppose that a (non-basic) state variable $x \in \mathcal{P}$ has a non-polynomial L_2-type update $g(\mathbf{Z})$, where $\mathbf{Z} = (Z_1, \ldots, Z_k)^T$ is a vector of (basic) continuous, independent, and identically distributed random variables *across iterations*. That is, if $f_{Z_j(n)}$ is the pdf of the random variable Z_j in iteration n, then $f_{Z_j(n)} \equiv f_{Z_j(n')}$, for all iterations n, n' and $j = 1, \ldots, k$. The basic random variables $Z_1, \ldots, Z_k$ and the update function g satisfy conditions (A)–(E) in Section 2.2.1. For the class of Prob-solvable loops where all variables with non-polynomial updates satisfy these conditions, the computation of the Fourier coefficients in the PCE approximation (2.7) can be carried out as explained in Section 2.2.2. In this case, the convergence rate is optimal.

2.3.2 Iteration Non-Stable Distribution of Random Arguments

Let $\mathcal{P}$ be an arbitrary Prob-solvable loop and suppose that a state variable $x \in \mathcal{P}$ has a non-polynomial L_2-type update $g(\mathbf{Z})$, where $\mathbf{Z} = (Z_1, \ldots, Z_k)^T$ is a vector of continuous independent but *not necessarily identically* distributed random variables across iterations. For this class of Prob-solvable loops, conditions (A)–(C) in Section 2.2.1 hold, but (D) and/or (E) may not be satisfied. In this case, we can ensure optimal exponential convergence by fixing the number of loop iterations. For unbounded loops, we describe an approach converging in mean-square and establish its convergence rate next.

Conditional estimator given number of iterations. Let N be an a priori fixed finite integer, representing the maximum iteration number. The set $\{1, \ldots, N\}$ is a finite sequence of iterations for the Prob-solvable loop $\mathcal{P}$.

Iterations are executed sequentially for $n = 1, \ldots, N$, which allows the estimation of the final functional that determines the target state variable at each iteration $n \in \{1, \ldots, N\}$ and its set of supports. Knowing these features, we can carry out N successive expansions. Let $P(n)$ be a PCE of $g(\mathbf{Z})$ for iteration n. We introduce an additional program variable c that counts the loop iterations. The variable c is initialized to 0 and incremented by 1 at the beginning of every loop iteration. The final estimator of $g(\mathbf{Z})$ can be represented as

$$\hat{g}(\mathbf{Z}) = \sum_{n=1}^{N} P(n) \left[\prod_{j=1, j\neq n}^{N} \frac{(c-j)}{n-j} \right]. \tag{2.12}$$

Replacing non-polynomial functions with (2.12) results in a program with only polynomial-type updates and *constant* polynomial structure; that is, polynomials with coefficients that remain constant across iterations. Moreover, the estimator is unbiased with optimal exponential convergence on the set of iterations $\{1, \ldots, N\}$ [XK02].

Unconditional estimator. Here the iteration number is unbounded. Without loss of generality, we consider a single basic random variable Z; that is, k=1. The function $g(Z)$ is scalar-valued and can be represented as a polynomial of *nested* L_2 functions, which depend on polynomials of the argument variable. Each nested functional argument is expressed as a sum of orthogonal polynomials yielding the final estimator, which is itself a polynomial.

Since PCE converges to the function it approximates in mean-square (see [EMS+12]) on the whole interval (argument's support), PCE converges on any sub-interval of the support of the argument in the same sense.

Let us consider a function g with a sufficiently large domain and a random variable Z with known distribution and support. For example, $g(Z) = e^Z$, with $Z \sim N(\mu, \sigma^2)$. The domain of g and the support of Z are the real line. We can expand g into a PCE with respect to the distribution of Z as

$$g(Z) = \sum_{i=0}^{\infty} c_i p_i(Z). \tag{2.13}$$

The distribution of Z is reflected in the polynomials in (2.13). Specifically, p_i, for $i = 0, 1, \ldots$, are Hermite polynomials of special type in that they are orthogonal (orthonormal) with respect to $N(\mu, \sigma^2)$. They also form an orthogonal basis of the space of L_2 functions. Consequently, any function in L_2 can be estimated arbitrarily closely by these polynomials. In general, any continuous distribution with finite moments of all orders and sufficiently large support can also be used as a model for basic variables in order to construct a basis for L_2 (see [EMS+12]).

Table 2.1 lists examples of functions of up to three random arguments approximated by PCE's of different degrees and, correspondingly, number of coefficients. We use $TruncNormal\left(\mu, \sigma^2, [a, b]\right)$ and $TruncGamma\left(\theta, k, [a, b]\right)$ to denote the truncated normal distribution with expectation μ and standard deviation σ on the (finite or infinite) interval $[a, b]$, and the truncated gamma distribution on the (finite or infinite) interval $[a, b]$, $a, b > 0$, with shape parameter k and scale parameter θ, respectively. The approximation error in (2.11) is reported in the last column. The results confirm (2.6) in practice: the error decreases as the degree or, equivalently, the number of components in the approximation of the polynomial increases.

Now suppose that the distribution of the underlying variable Z is unknown with pdf $f_Z(z)$ that is continuous on its support $[a, b]$. Then, there exists another basis of polynomials, $\{q_i\}_{i=0}^{\infty}$, which are orthogonal on the support $[a, b]$ with respect to the pdf f_Z. Then, on the interval $[a, b]$, $g(Z) = \sum_{i=0}^{\infty} k_i q_i(Z)$, and $\mathbb{E}_f\left[g(Z)\right] = \mathbb{E}_f\left[\sum_{i=0}^{M} k_i q_i(Z)\right], \forall M \geq 0$.

Since $[a, b] \subset \mathbb{R}$, the expansion $\sum_{i=0}^{n} c_i p_i(Z)$, where p_i are Hermite polynomials, converges in mean-square to $g(Z)$ on $[a, b]$ with $n \to \infty$. Also, $\mathbb{E}_f\left(g(Z)\right) = \mathbb{E}_f\left(\sum_{i=0}^{\infty} c_i p_i(Z)\right)$ for the true pdf f on $[a, b]$. In general, though, it is not true that $\mathbb{E}_f\left(g(Z)\right) = \mathbb{E}_f\left(\sum_{i=0}^{M} c_i p_i(Z)\right)$ for any arbitrary $M \geq 0$ and any pdf $f_Z(z)$ on $[a, b]$, as the estimator is biased.

Let us consider a general case where f_Z is exponentially integrable on a support Ω, and τ is exponentially integrable positive everywhere on a support $\mathcal{Q} \subset \mathbb{R}$ such that $\Omega \subseteq \mathcal{Q}$. We notice that any continuous pdf on a bounded support is exponentially integrable. Let $\{p_i^\tau\}$ be a set of orthogonal polynomials on $\mathcal{Q}$ with respect to density τ.

Assumption 1. *The ratio $f_Z(x)/\tau(x)$ is in $L_1(\Omega, f_Z)$.*

We define $\widetilde{f}_Z(x)$ on $\mathcal{Q}$ to be

$$\widetilde{f}_Z(x) = \begin{cases} f_Z(x), & x \in \Omega, \\ 0, & x \in \mathcal{Q} \setminus \Omega. \end{cases}$$

Since $\{p_i^\tau\}$ is an orthonormal system on $\mathcal{Q}$ with respect to pdf τ, any function in $L_2(\mathcal{Q}, \tau)$ can be expanded into a Fourier series (see, e.g., [KF76] or [Rud76]) along the $\{p_i^\tau\}$ basis elements. Under

Function	Random Variables	Degree / #coefficients	Error
$f(x_1,x_2) = \xi e^{-x_1} + (\xi - \frac{\xi^2}{2})e^{x_2-x_1}$ $\xi = 0.3$	$x_1 \sim Normal(0,1)$, $x_2 \sim Normal(2, 0.01)$	1 / 4	3.076846
		2 / 9	1.696078
		3 / 16	0.825399
		4 / 25	0.363869
		5 / 36	0.270419
$f(x_1,x_2) = 0.3e^{x_1-x_2} + 0.6e^{-x_2}$	$x_1 \sim TruncNormal(4,1,[3,5])$, $x_2 \sim TruncNormal(2, 0.01, [0,4])$	1 / 4	0.343870
		2 / 9	0.057076
		3 / 16	0.007112
		4 / 25	0.000709
		5 / 36	0.000059
$f(x_1,x_2) = e^{x_1 x_2}$	$x_1 \sim TruncNormal(4,1,[3,5])$ $x_2 \sim TruncGamma(3,1,[0.5,1])$	1 / 4	5.745048
		2 / 9	1.035060
		3 / 16	0.142816
		4 / 25	0.016118
		5 / 36	0.001543
$f(x_1,x_2,x_3) = 0.3e^{x_1-x_2} +$ $0.6e^{x_2-x_3} + 0.1e^{x_3-x_1}$	$x_1 \sim TruncNormal(4,1,[3,5])$ $x_2 \sim TruncGamma(3,1,[0.5,1])$ $x_3 \sim U[4,8]$	1 / 8	1.637981
		2 / 27	0.303096
		3 / 64	0.066869
$f(x_1) = \psi cos(x_1) + (1-\psi)sin(x_1)$ $\psi = 0.3$	$x_1 \sim Normal(0,1)$	1 / 2	0.222627
		2 / 3	0.181681
		3 / 4	0.054450
		4 / 5	0.039815
		5 / 6	0.009115

Table 2.1: Approximations of 5 non-linear functions using PCE.

Assumption 1, $r(x) = \widetilde{f}(x)/\phi(x)$ satisfies

$$\int_{\Theta} r^2(x)\tau(x)dx = \int_{\mathcal{Q}\setminus\Omega} \frac{\widetilde{f}_Z(x)}{\tau(x)}\widetilde{f}_Z(x)dx + \int_{\Omega} \frac{f_Z(x)}{\tau(x)}dF_Z(x) < \infty, \tag{2.14}$$

so that $r(x) \in L_2(\mathcal{Q},\tau)$. Moreover, $r(x) \in L_2(\mathcal{Q}',\tau)$ for any $\mathcal{Q}' \subseteq \mathcal{Q}$.

Theorem 1. *Suppose the density f_Z of Z is supported on Ω and exponentially integrable. Assume Assumption 1 holds and let τ denote the exponentially integrable pdf that is positive everywhere on $\mathcal{Q}$, such that $\Omega \subseteq \mathcal{Q}$. Additionally, suppose $g : \mathbb{R} \to \mathbb{R}$ with $g \in L_2(\mathcal{Q},\tau)$ and $g \in L_2(\Omega, f_Z)$, and that $g_n^{f_Z} = \sum_{i=0}^{n} c_i^{f_Z} p_i^{f_Z}(x)$ is the PCE of g on Ω with respect to f_Z and $g_n^{\tau} = \sum_{i=0}^{n} c_i^{\tau} p_i^{\tau}(x)$ is the PCE of g on $\mathcal{Q}$ with respect to τ. Then*

(a) $\|g - g_n^{\tau}\|_{L_1(\Omega,f_Z)} \xrightarrow[n\to\infty]{} 0$;

(b) $\|g_n^{\tau} - g_n^{f_Z}\|_{L_1(\Omega,f_Z)} \xrightarrow[n\to\infty]{} 0$.

Proof.

$$\|g_n^{\tau} - g_n^{f_Z}\|_{L_1(\Omega,f_Z)} \leq \|g - g_n^{f_Z}\|_{L_1(\Omega,f_Z)} + \|g - g_n^{\tau}\|_{L_1(\Omega,f_Z)}$$

The first norm in the right-hand side converges to zero due to the convergence of $g_n^{f_Z}$ to g in L_2 which implies convergence in L_1. Taking a closer look at the second term,

$$\begin{aligned}\|g - g_n^\tau\|_{L_1(\Omega,f_Z)} &= \int_\Omega |g(x) - g_n^\tau(x)| f_Z(x)\, dx = \int_\Omega |g(x) - g_n^\tau(x)| \tau(x) \frac{f_Z(x)}{\tau(x)}\, dx \\ &= \left\langle |g - g_n^\tau|, \frac{f_Z}{\tau} \right\rangle_{L_2(\Omega,\tau)} \overset{\text{Cauchy-Schwarz}}{\leq} \|g - g_n^\tau\|_{L_2(\Omega,\tau)} \cdot \underbrace{\left\| \frac{f_Z}{\tau} \right\|_{L_2(\Omega,\tau)}}_{< \infty \text{ due to Assumption 1}} \\ &\leq C \cdot \|g - g_n^\tau\|_{L_2(\mathcal{Q},\tau)} \xrightarrow[n\to\infty]{} 0\end{aligned}$$

□

2.4 Exact Moment Derivation

In Sections 2.2 and 2.3, we combined PCE with the *Prob-solvable loop* algorithm of [MSB+22; BKS19] to compute PCE *approximations* of the moments of the distributions of the random variables generated in a probabilistic loop. In this section, we develop a method for the derivation of the *exact* moments of probabilistic loops of specified loop structure and functional assignments. [BKS19], and later [MSB+22], introduce a technique for exact moment computation for *Prob-solvable loops* without non-polynomial functions. *Prob-solvable loops* support common probability distributions F with constant parameters and program variables x with specific polynomial assignments (cf. Section 2.1). In Section 2.4.1, we first show how to compute $\mathbb{E}(h(F)^k)$ where h is the exponential or a trigonometric function. In Section 2.4.2, we describe how to incorporate trigonometric and exponential updates of program variables x into the *Prob-solvable loop* setting.

2.4.1 Trigonometric and Exponential Functions of Random Variables

The first step in supporting trigonometric and exponential functions in Prob-solvable loops is to understand how to compute the expected values of random variables that are trigonometric and exponential functions of random variables with known distributions. Due to the polynomial arithmetic supported in Prob-solvable loops, non-polynomial functions of random variables can be mixed via multiplication in the resulting program. We adopt the results from [JWW21] providing a formula for the expected value of mixed trigonometric polynomials of distributions, given the distributions' characteristic functions.

Definition 3. *We call* $p(x)$ *a* standard polynomial of order $d \in \mathbb{N}$ *if*

$$p(x) = \sum_{k=0}^{d} \alpha_k x^k,$$

with coefficients $\alpha_k \in \mathbb{R}$. *Further,* $p(x)$ *is defined to be a* mixed trigonometric polynomial *if it is a mixture of a standard polynomial with trigonometric functions of the form*

$$p(x) = \sum_{k=0}^{d} \alpha_k x^{b_k} \cdot cos^{c_k}(x) \cdot sin^{s_k}(x),$$

with $b_k, c_k, s_k \in \mathbb{N}$ and coefficients $\alpha_k \in \mathbb{R}$ [JWW21].

Following [JWW21], we define the mixed-trigonometric-polynomial moment of order α for a random variable X as

$$m_{X^{\alpha_1} c_X^{\alpha_2} s_X^{\alpha_3}} = \mathbb{E}\left[X^{\alpha_1} \cos^{\alpha_2}(X) \sin^{\alpha_3}(X)\right], \tag{2.15}$$

where $(\alpha_1, \alpha_2, \alpha_3) \in \mathbb{N}^3$ such that $\alpha = \sum_{k=1}^{3} \alpha_k$. When the characteristic function of the random variable X is known, Lemma 4 in [JWW21], together with the linearity of the expectation operator, provides the computation rule for (2.15):

$$m_{X^{\alpha_1} c_X^{\alpha_2} s_X^{\alpha_3}} = \frac{1}{i^{\alpha_1+\alpha_3} 2^{\alpha_2+\alpha_3}} \times \sum_{(k_1,k_2)=(0,0)}^{(\alpha_2,\alpha_3)} \binom{\alpha_2}{k_1} \binom{\alpha_3}{k_2} (-1)^{\alpha_3-k_2} \frac{d^{\alpha_1}}{dt^{\alpha_1}} \Phi_X(t)\Big|_{t=2(k_1+k_2)-\alpha_2-\alpha_3}. \tag{2.16}$$

Example 3. *Let $\mathbf{X} \sim \mathcal{N}(0,1)$. Its characteristic function is $\Phi_X(t) = \exp(-0.5t^2)$ and $\Phi'_X(t) = -t\exp(-0.5t^2)$. Then,*

$$\mathbb{E}\left[X \sin(X) \cos(X)\right] = \frac{1}{i^2 2^2}\left(-\Phi'_X(-2) + \Phi'_X(0) - \Phi'_X(0) + \Phi'_X(2)\right) = \exp(-2).$$

To support exponential functions in Prob-solvable loops, we define *mixed exponential polynomials* as

$$p(x) = \sum_{k=0}^{d} \alpha_k x^{b_k} \exp^{e_k}(x),$$

with $b_k, e_k \in \mathbb{N}$ and coefficients $\alpha_k \in \mathbb{R}$. Lemma 1 obtains a computational rule for moments of mixed exponential polynomials, provided they exist, in terms of the moment-generating function of the random variable X.

Lemma 1. *Let X be a random variable with moment-generating function $\mathcal{M}_X(t) = \mathbb{E}\left[\exp(tX)\right]$. Suppose $\alpha_1, \alpha_2 \in \mathbb{N}$, and let $\alpha = \alpha_1 + \alpha_2$. If the mixed-exponential-polynomial moment of order α exists, it can be computed using the following formula*

$$m_{X^{\alpha_1} e_X^{\alpha_2}} = \mathbb{E}\left[X^{\alpha_1} \exp^{\alpha_2}(X)\right] = \frac{d^{\alpha_1}}{dt^{\alpha_1}} \mathcal{M}_X(t)\Big|_{t=\alpha_2}. \tag{2.17}$$

Proof.

$$\begin{aligned} m_{X^{\alpha_1} e_X^{\alpha_2}} = \mathbb{E}\left[X^{\alpha_1} \exp^{\alpha_2}(X)\right] &= \mathbb{E}\left[\frac{d^{\alpha_1}}{dt^{\alpha_1}} \exp(tX)\Big|_{t=\alpha_2}\right] \\ &= \frac{d^{\alpha_1}}{dt^{\alpha_1}} \mathbb{E}\left[\exp(tX)\right]\Big|_{t=\alpha_2} = \frac{d^{\alpha_1}}{dt^{\alpha_1}} \mathcal{M}_X(t)\Big|_{t=\alpha_2}. \end{aligned}$$

□

Example 4. *Suppose $X \sim \mathcal{N}(0,1)$ with moment generating function $\mathcal{M}_X(t) = \exp(0.5t^2)$. Since $X\exp^2(X)$ is integrable with respect to the normal pdf,*

$$\mathbb{E}\left[X \exp^2(X)\right] = \frac{d}{dt} \exp(0.5t^2)\Big|_{t=2} = t \cdot \exp(0.5t^2)\Big|_{t=2} = 2 \cdot \exp(2).$$

Listing 2.1: PP loop prototype

$z = 0;\ x = 0;\ y = 0$
`while true:`
    $z = Normal(0, 1)$
    $x = x + z$
    $y = y + h(x)$
`end`

2.4.2 Trigonometric and Exponential Functions in Variable Updates

We now examine the presence of trigonometric and exponential functions of program variables, specifically of *accumulator* variables, in Prob-solvable loops.

Definition 4 (Accumulator)**.** *We call a program variable x an* accumulator *if the update of x in the loop body has the form $x = x + z$, such that z_i and z_{i+1} are independent and identically distributed for all $i \geq 1$.*

Consider a loop $\mathcal{P}$ with an accumulator variable x, updated as $x = x + z$, and a trigonometric or exponential function $h(x)$. Further, assume that the characteristic function (if $h = \sin$ or $h = \cos$) or the moment-generating function (if $h = \exp$) of z is known. Note that the distribution of the variable x is, in general, different in every iteration. Listing 2.1 gives an example of such a loop.

The idea now is to transform $\mathcal{P}$ into an equivalent Prob-solvable loop $\mathcal{P}'$ such that the term $h(x)$ does not appear in $\mathcal{P}'$. In the following, we assume, for simplicity, that first z is updated in $\mathcal{P}$, then x, and only then h is used. The following arguments are analogous if the updates are ordered differently (only the indices change). Note that we can rewrite $h(x_{t+1})$ as $h(x_t + z_{t+1})$.

Transforming $\exp(x)$. In the case of $h = \exp$, we have

$$\exp(x_{t+1}) = \exp(x_t + z_{t+1}) = \exp(x_t)\exp(z_{t+1}). \tag{2.18}$$

We utilize this property and transform the program $\mathcal{P}$ into a program $\mathcal{P}'$ by introducing an auxiliary variable $\hat{x}$ that models the value of $\exp(x)$. The update of $\hat{x}$ in the loop body succeeds the update of x and is

$$\hat{x} = \hat{x} \cdot \exp(z). \tag{2.19}$$

The auxiliary variable is initialized as $\hat{x}_0 = \exp(x_0)$. We then replace $h(x)$ by $\hat{x}$ in $\mathcal{P}$ to arrive at our transformed program $\mathcal{P}'$. Because z is identically distributed in every iteration and its moment-generating function is known, we can use the results from Section 2.4.1 to compute any moment of $\exp(z)$. Thus, the update in (2.19) is supported by Prob-solvable loops.

Transforming sine and cosine. In the case of $h = \sin$ or $h = \cos$, applying standard trigonometric identities obtains

$$\begin{aligned} \cos(x_{t+1}) &= \cos(x_t + z_{t+1}) = \cos(x_t)\cos(z_{t+1}) - \sin(x_t)\sin(z_{t+1}), \\ \sin(x_{t+1}) &= \sin(x_t + z_{t+1}) = \sin(x_t)\cos(z_{t+1}) + \cos(x_t)\sin(z_{t+1}). \end{aligned} \tag{2.20}$$

We introduce two auxiliary variables, s and c, modeling the values of $\sin(x)$ and $\cos(x)$, simultaneously updated[6] in the loop body as

$$c,\ s = c \cdot \cos(z) - s \cdot \sin(z),\ s \cdot \cos(z) + c \cdot \sin(z), \tag{2.21}$$

with the initial values $s_0 = \sin(x_0)$ and $c_0 = \cos(x_0)$. We then replace $h(x)$ with s or c in $\mathcal{P}'$. Again, because z is identically distributed in every iteration and its characteristic function is known, we can use the results from Section 2.4.1 to compute any moment of $\sin(z)$ and $\cos(z)$. Thus, the update in (2.21) is supported in Prob-solvable loops.

Example 5. *Listings 2.2 and 2.3 show the program from Listing 2.1 with $h = \exp$ and $h = \cos$, respectively, rewritten as equivalent Prob-solvable loops. The program in Listing 2.2 has linear circular variable dependencies due to the variables c and s.*

Listing 2.2: PP prototype equivalent – cos

$z = 0;\ x = 0;\ c = cos(z)$
$s = sin(z);\ y = 0$
while `true:`
  $z = Normal(0, 1)$
  $x = x + z$
  $c,\ \ s = c \cdot \cos(z) - s \cdot \sin(z),\ \ s \cdot \cos(z) + c \cdot \sin(z)$
  $y = y + c$
end

Listing 2.3: PP prototype equivalent – exp

$z = 0;\ x = 0;\ \hat{x}_i = exp(z);\ y = 0$
while `true:`
  $z = Normal(0, 1)$
  $x = x + z$
  $\hat{x}_i = \hat{x}_i \cdot \exp(z)$
  $y = y + \hat{x}_i$
end

The following properties of $\sin$ and $\cos$ functions are also supported in Prob-solvable loops:

$$\sin(n\alpha) = \sum_{\substack{r=0,\\ 2r+1\leq n}} (-1)^r \binom{n}{2r+1} \cos^{n-2r-1}(\alpha) \sin^{2r+1}(\alpha), n \in \mathbb{N}$$

$$\cos(n\alpha) = \sum_{\substack{r=0,\\ 2r\leq n}} (-1)^r \binom{n}{2r} \cos^{n-2r}(\alpha) \sin^{2r}(\alpha), n \in \mathbb{N}$$

2.5 Evaluation

We evaluate our PCE-based method for moment approximation and our exact moment derivation approach on eleven benchmarks. We apply our PCE-based method to approximate non-polynomial functions. This transforms all benchmark programs into Prob-solvable loops, which allows using the static analysis tool Polar [MSB+22] to compute the moments of the program variables as a function of the loop iteration n.

We implemented the techniques for exact moment derivation for loops containing trigonometric or exponential polynomials, presented in Section 2.4, in the tool Polar. We evaluate the technique for

[6] Simultaneous updates $c, s = \text{expr}_1, \text{expr}_2$ can always be expressed as sequentially: $t_1 = \text{expr}_1; t_2 = \text{expr}_2; c = t_1; s = t_2$.

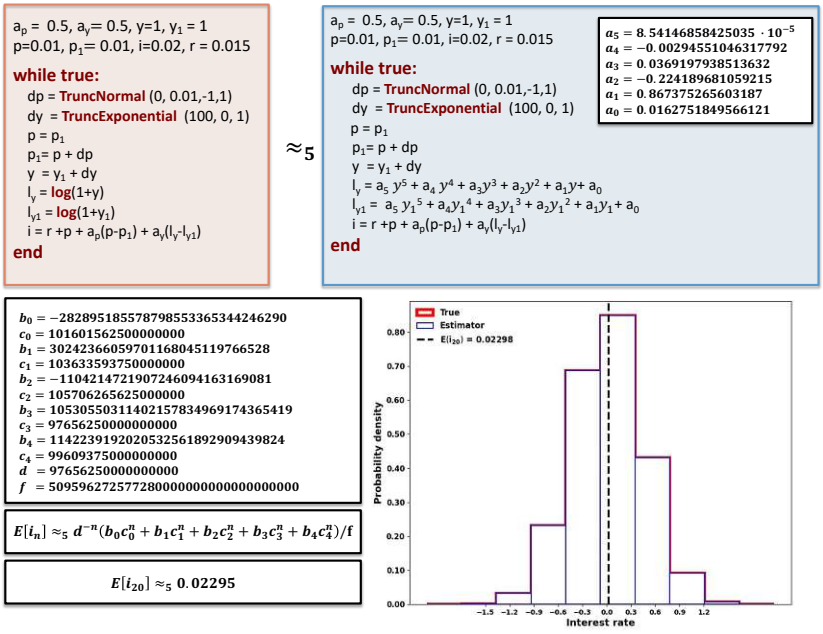


Figure 2.2: The probabilistic loop in the top left panel encodes the Taylor rule [Tay93], an equation that prescribes a value for the short-term interest rate based on a target inflation rate and the gross domestic product. The program uses a non-polynomial function (log) in the loop body to update the continuous-state variable (i). The top right panel contains the *Prob-Solvable loop* (with polynomial updates) obtained by approximating the log function using polynomial chaos expansion (up to 5th degree). In the bottom left, we compute the expected interest rate ($\mathbb{E}[i_n]$) as a closed-form expression in loop iteration n using the Prob-solvable loop and evaluate it at $n = 20$. In the bottom right panel, we compare the true and estimated distributions for a fixed iteration (we sample the loop 10^6 times at iteration $n = 20$).

exact moment derivation using Polar on all benchmarks satisfying the general program structure of Listing 2.1 in Section 2.4. We also compare our approximate and exact methods with the technique based on *polynomial forms* of [SCG+20]. When appropriate, we applied our methods, as well as the polynomial form, on the eleven benchmark models. All experiments were run on a machine with 32 GB of RAM and a 2.6 GHz Intel i7 (Gen 10) processor.

Taylor rule model. Central banks set monetary policy by raising or lowering their target for the federal funds rate. The Taylor rule[7] is an equation intended to describe the interest rate decisions of central banks. The rule relates the target of the federal funds rate to the current state of the economy through the formula

$$i_t = r_t^* + \pi_t + a_\pi(\pi_t - \pi_t^*) + a_y(y_t - \bar{y}_t), \tag{2.22}$$

where i_t is the nominal interest rate, r_t^* is the equilibrium real interest rate, $r_t^* = r$, π_t is inflation rate at t, π_t^* is the short-term target inflation rate at t, $y_t = \log(1 + Y_t)$, with Y_t the real GDP, and $\bar{y}_t = \log(1 + \bar{Y}_t)$, with $\bar{Y}_t$ denoting the potential real output.

[7] It was proposed by the American economist John B. Taylor as a technique to stabilize economic activity by setting an interest rate [Tay93].

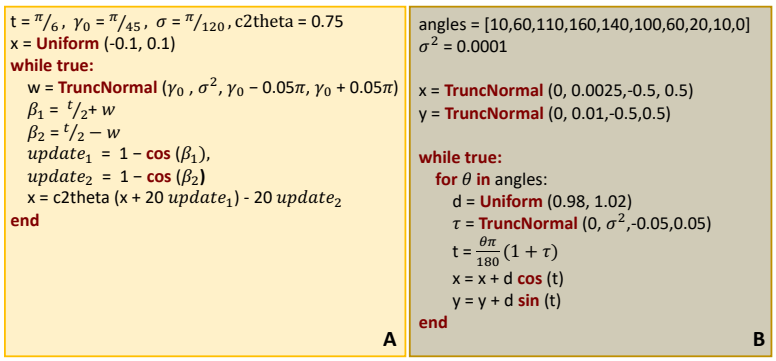


Figure 2.3: Probabilistic loops: (A) Rimless wheel walker [ST12] and (B) 2D Robotic arm [BGP+16] (in the figure we use the inner loop as syntax sugar to keep the program compact).

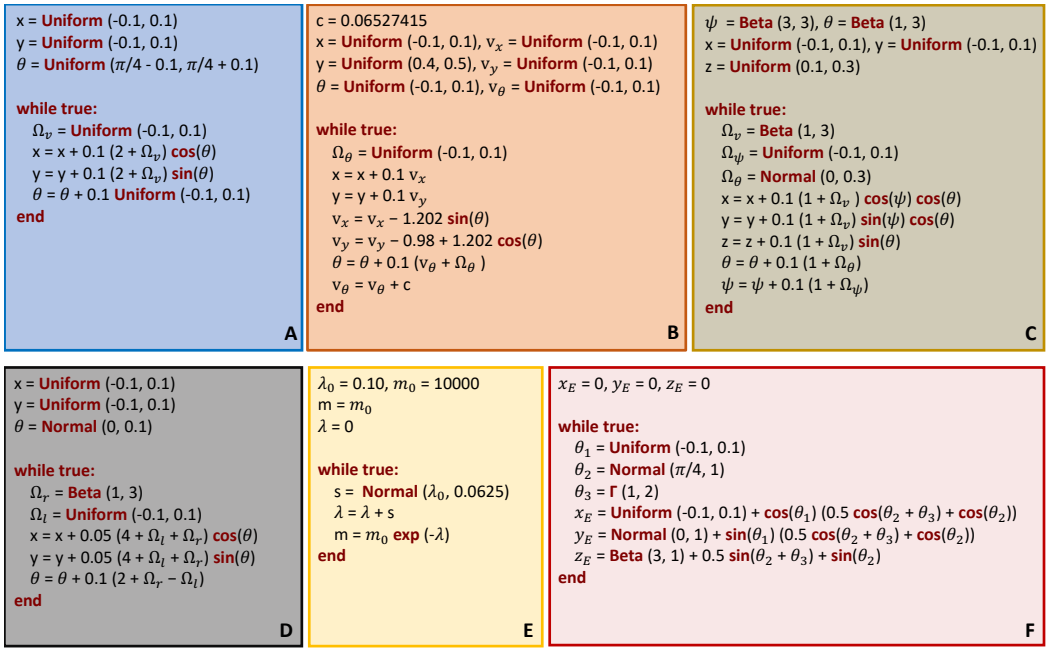


Figure 2.4: Probabilistic loops: (A) Uncertain underwater vehicle [PHC+22; JWW21], (B) Planar aerial vehicle [ST12; JWW21], (C) 3D aerial vehicle [PHC+22; JWW21], (D) Differential-drive mobile robot [vdBAG11; JWW21], (E) Stochastic decay, (F) 3D (Mobile) Robotic arm [JF14; JWW21]

Highly-developed economies grow exponentially with a sufficiently small rate (e.g., according to the World Bank,[8] the average growth rate of the GDP in the USA in 2001-2020 equaled 1.73%). Accordingly, we set the growth rate of the potential output to 2%. We model inflation as a martingale process; that is, $\mathbb{E}_t[\pi_{t+1}] = \pi_t$, following [AO01]. The Taylor rule model is described by the program in Fig. 2.2.

[8] `https://data.worldbank.org/indicator/NY.GDP.MKTP.KD.ZG?locations=US`

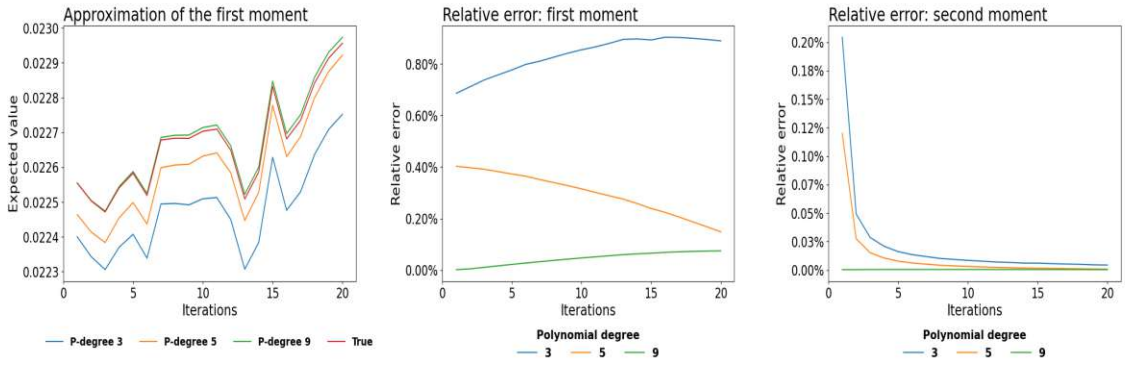


Figure 2.5: The approximations and their relative errors for the Taylor rule model.

Turning vehicle model. This model is described by the probabilistic program in Fig. 1.3. It was introduced in [SCG+20] and depicts the position of a vehicle, as follows. The state variables are (x, y, v, ψ), where (x, y) is the vehicle's position with velocity v and yaw angle ψ. The vehicle's velocity is stabilized around $v_0 = 10$ m/s. The dynamics are modelled by the equations $x(t{+}1) = x(t) + \tau v \cos(\psi(t))$, $y(t{+}1) = y(t) + \tau v \sin(\psi(t))$, $v(t{+}1) = v(t) + \tau(K(v(t) - v_0) + w_1(t{+}1))$, and $\psi(t{+}1) = \psi(t) + w_2(t{+}1)$. The disturbances w_1 and w_2 have distributions $w_1 \sim U[-0.1, 0.1]$, $w_2 \sim N(0, 0.1)$. We set $K = -0.5$, as in [SCG+20]. Initially, the state variables are distributed as follows: $x(0) \sim U[-0.1, 0.1]$, $y(0) \sim U[-0.5, -0.3]$, $v(0) \sim U[6.5, 8.0]$, $\psi(0) \sim N(0, 0.01)$. We allow all normally distributed parameters to take values over the entire real line, in contrast to [SCG+20] which could not accommodate distributions with infinite support and required the normal variables to be truncated.

Rimless wheel walker. The *Rimless wheel walker* [SCG+20; ST12] is a system that describes a walking human. The system models a rotating wheel consisting of n_s spokes, each of length L, connected at a single point. The angle between consecutive spokes is $\theta = 2\pi/n_s$. We set $L = 1$ and $\theta = \pi/6$. This system is modeled by the program in Fig. 2.3 (A). For more details, we refer to [SCG+20].

Robotic arm model. Proposed and studied in [BGP+16; San20; SCG+20], this system models the position of a 2D robotic arm. The arm moves through translations and rotations. At every step, errors in movement are modeled with probabilistic noise. The *robotic arm model* is described by the program in Fig. 2.3 (B).

Uncertain underwater vehicle. This benchmark models the movement of an underwater vehicle subject to external disturbances [JWW21; PHC+22] and is encoded by the program in Fig. 2.4 (A). The program variables x and y represent the position of the vehicle in a 2D plane and θ its orientation. The external disturbances are modeled by probabilistic shocks to the velocity and the orientation of the vehicle.

Planar aerial vehicle. This benchmark was studied in [JWW21; ST12] and models the vertical and horizontal movement of an aerial vehicle subject to wind disturbances. It can be written as the program in Fig. 2.4 (B), where the variables x and y represent the horizontal and vertical positions. The variable θ models the rotation around the x axis. The linear velocities are captured by v_x

and v_y, and v_θ represents the angular velocity. The wind disturbance is modeled by the random variable Ω_θ. For more details, we refer to [JWW21].

3D aerial vehicle. This system, studied in [JWW21; PHC+22], models the movement of an aerial vehicle in three-dimensional space subject to wind disturbances. The system can be written as a program as illustrated in Fig. 2.4 (C). The program variables x, y, and z represent the position of the vehicle. The orientations around the y and z axis are captured by the variables θ and ψ, respectively. The linear and angular velocities are constant 1. Wind disturbances are modeled by the random variables Ω_ν, Ω_ψ, and Ω_θ. For more details, we refer to [JWW21].

Differential-drive mobile robot. This system models the movement of a differential-drive mobile robot with two wheels subject to external disturbances and was studied in [JWW21; vdBAG11]. In Fig. 2.4 (D) we express the *Differential-drive mobile robot* system as a program. The program variables x and y represent the robot's position. Its orientation is captured by the variable θ. The velocities are constant 1 for the left wheel and constant 3 for the right wheel. The random variables Ω_r and Ω_l model external disturbances. For more details, we refer to [JWW21].

Mobile robotic arm. The system, studied in [JWW21; JF14; JF10], models the uncertain position of the end-effector of a mobile robotic arm as a function of the uncertain base position and uncertain joint angles. Fig. 2.4 (F) shows the system as a program. The program variables θ_1, θ_2, and θ_3 represent the uncertain angles of three joints. The distributions of θ_1 and θ_2 are uniform and normal, respectively, while θ_3 is gamma distributed with shape parameter 1 and scale parameter 2. The position of the end-effector in 3D space is given by the variables x_E, y_E, and z_E. The uncertain position of the base in 3D space is modeled by three different distributions (uniform, normal, beta) in the assignments of x_E, y_E, and z_E. For more details, we refer to [JWW21].

Stochastic decay. The program in Fig. 2.4 (E) models exponential decay with a non-constant stochastic decay rate. Variable m represents the total quantity subject to decay, where m_0 is the initial quantity. The decay rate λ starts off at 0 and changes according to a normal distribution at every time step.

Fig. 2.5 illustrates the performance of our PCE-based approach as a function of the polynomial degree of our approximation on the *Taylor rule*. The approximations to the true first moment (in red) are plotted in the left panel and the relative errors, calculated as $rel.err = |est - true|/true$, for the first and second moments in the middle and right panels, respectively, over iteration number. All plots show that the approximation error is low and deteriorates as the polynomial degree increases from 3 to 9, across iterations. For this benchmark, the drop is sharper for the second moment.

The *Rimless wheel walker* and the *Robotic arm* models are the only two benchmarks from [SCG+20] with nonlinear non-polynomial updates. Polynomial forms of degree 2 were used to compute bounding intervals for $\mathbb{E}(x_n)$ (for fixed n) for these two models. The [SCG+20] tool supports neither the approximation of logarithms (required for the *Taylor rule model*) nor distributions with unbounded support (required for all benchmarks except for the *Taylor rule model* on which the tool fails). To facilitate comparison with polynomial forms, our set of benchmarks is augmented with a version of the *Turning vehicle model* using truncated normal distributions (ψ and $w_2 \sim TruncNormal(0, 0.01, [-1, 1])$ in Fig. 1.4), which is called *Turning vehicle model (trunc.)* in Table 2.2), instead of normal distributions with unbounded support.

Among the eleven benchmark models in Table 2.2, the polynomial form tool of [SCG+20] can be used to approximate moments only in five, namely the *Turning vehicle model (trunc.)*, *Rimless*

Benchmark	Target	Poly form	Sim.	Exact	PCE estimate		
					Deg.	Result	Runtime
Taylor rule model	$\mathbb{E}(i_n)$ n=20	×	0.022998	×	3	0.02278	$0.4s$+$0.5s$
					5	0.02295	$0.5s$+$5.0s$
					9	0.02300	$5.9s$+$34.6s$
Turning vehicle model	$\mathbb{E}(x_n)$ n=20	×	15.60666	15.60760 1.9s	3	14.44342	$0.6s$+$3.6s$
					5	15.43985	$1.4s$+$9.2s$
					9	15.60595	$15.6s$+$16.1s$
Turning vehicle model (trunc.)	$\mathbb{E}(x_n)$ n=20	for deg. 2 $[-3 \cdot 10^5, 3 \cdot 10^5]$ 1057s	15.60818	15.60760 89.2s	3	14.44342	$0.6s$+$3.6s$
					5	15.43985	$1.4s$+$9.1s$
					9	15.60595	$15.6s$+$15.8s$
Rimless wheel walker	$\mathbb{E}(x_n)$ n=2000	for deg. 2 $[1.791, 1.792]$ 5.42s	1.79173	1.79159 8.0s	1	1.79159	$0.2s$+$0.5s$
					2	1.79159	$0.3s$+$0.4s$
					3	1.79159	$0.6s$+$0.6s$
Robotic arm model	$\mathbb{E}(x_n)$ n=100	for deg. 2 $[268.87, 268.88]$ 2.74s	268.852	268.85236 5.6s	1	268.85236	$1.3s$+$0.3s$
					2	268.85236	$2.5s$+$0.6s$
					3	268.85236	$4.8s$+$0.7s$
Uncertain underwater vehicle	$\mathbb{E}(x_n^2)$ n=10	for deg. 2 $[1.9817, 2.0252]$ 2.9s	2.00332	2.00339 0.6s	3	2.08986	$0.1s$+$0.9s$
					5	2.04514	$0.1s$+$2.8s$
					8	2.00432	$0.6s$+$8.6s$
Planar aerial vehicle	$\mathbb{E}(y_n)$ n=10	for deg. 2 $[1.4306, 1.4315]$ 4.1s	1.43111	×	6	1.42184	$0.2s$+$5.9s$
					8	1.43016	$0.6s$+$13.7s$
					10	1.43099	$2.1s$+$28.0s$
3D aerial vehicle	$\mathbb{E}(x_n)$ n=20	×	0.67736	0.67770 4.9s	3	0.47805	$0.1s$+$1.5s$
					5	0.65280	$0.1s$+$5.7s$
					8	0.67245	$0.6s$+$30.5s$
Differential-drive mobile robot	$\mathbb{E}(x_n^2)$ n=25	×	0.29175	0.29151 12.0s	8	0.19919	$0.6s$+$9.5s$
					10	0.29310	$2.1s$+$13.8s$
					12	0.29215	$8.3s$+$22.4s$
Mobile Robotic Arm	$\mathbb{E}(x_n)$ n=2000	×	0.38413	0.38535 0.2s	2	0.38535	$0.8s$+$0.2s$
					3	0.38535	$1.3s$+$0.3s$
					4	0.38535	$2.0s$+$0.5s$
Stochastic decay	$\mathbb{E}(m_n)$ n=10	×	5031.8404	5028.3158 0.3s	6	5035.7468	$1.9s$+$1.0s$
					8	5028.0312	$4.7s$+$1.6s$
					10	5028.3222	$15.6s$+$2.0s$

Table 2.2: Evaluation of our approach on 11 benchmarks. Poly form = the interval for the target as reported by [SCG+20]; Sim = target approximated through 10^6 samples; Exact = the target result computed by our technique for exact moment derivation; Deg. = maximum degrees used for the approximation of the non-linear functions; Result = result of our approximate method per degree; Runtime = execution time of our method in seconds (time of PCE + time of Polar); × = the respective method is not applicable.

wheel walker, *Robotic arm*, *Uncertain underwater vehicle*, and *Planar aerial vehicle*. Our method for exact moment derivation supports trigonometric functions and the exponential function but no logarithms. Hence, it is not applicable to the *Taylor rule model*. Moreover, our exact method cannot be applied to the *Planar aerial vehicle* benchmark because the perturbation of its program variable θ is not iteration-stable and θ is used as an argument to a trigonometric function. Our PCE-based moment estimation approach applies to all.

The *Robotic arm*, *Rimless wheel walker*, and *Mobile robotic arm* models contain no stochastic accumulation: each basic random variable is iteration-stable and can be estimated using the scheme in Section 2.3.1. Therefore, for these benchmarks, our estimates converge exponentially fast to the true values. In fact, our estimates coincide with the true values for first moments, because the estimators are unbiased. The other benchmarks contain stochasticity accumulation, which leads to the instability of the distributions of basic random variables. For these benchmarks, we apply the scheme in Section 2.3.2.

Table 2.2 contains the evaluation results of our approximate and exact approaches, and of the technique based on *polynomial forms* of [SCG+20] on the eleven benchmarks. In consecutive order, the table columns are: the name of the benchmark model; the target moment and iteration; the *polynomial form* results (estimation interval and runtime), if applicable; the sampling-based value of the target moment; the exact moment and the runtime of its calculation, if applicable; the truncation parameter (polynomial degree) in PCE; the PCE estimate value; and the PCE estimate calculation runtime.

Our results illustrate that our method based on PCE is able to accurately approximate general non-linear dynamics for challenging programs. Specifically, for the *Rimless wheel walker* model, our first moment estimate coincides with the exact result and falls in the interval estimate of the polynomial forms technique. For the *Robotic arm model*, our results are equal to the exact result and closer to the sampling one based on 10^6 samples. They lie outside the interval predicted by the polynomial forms technique, pointing to the latter's lack of accuracy in this model.

Our method for exact moment derivation can be faster than the polynomial form technique and our PCE-based approximation approach, for instance, for the *Turning vehicle model*. Nevertheless, if all basic random variables are iteration-stable, our approximation approach will provide an unbiased estimation and hence the exact result for the first moments. This is the case, for example, for the *Rimless wheel walker* benchmark for which our approximation method provides the true result in under 0.7s, compared to our exact moment derivation method which needs 8s.

Our experiments also demonstrate that our PCE-based method provides accurate approximations in a fraction of the time required by the polynomial form based technique. While polynomial forms compute an error interval, they need to be computed on an iteration-by-iteration basis. In contrast, our method based on PCE and Prob-solvable loops computes an expression for the target parameterized by the loop iteration $n \in \mathbb{N}$ (cf. Fig. 1.4). As a result, increasing the target iteration n does *not* increase the runtime of our approach. To see this, consider the *Uncertain underwater vehicle* benchmark: the runtimes of polynomial forms and of our approach using the PCE estimate of order 5 are comparable (2.9s). However, increasing the target iteration n from 10 to 20 escalates the runtime of polynomial forms to 237s while the runtimes of both our approaches (approximate and exact) remain the same.

2.6 Conclusion

We present two methods, one exact and one approximate, to compute in closed form the state variable moments in probabilistic loops with non-polynomial updates. The latter employs polynomial chaos expansion to estimate non-polynomial, general functional relationships. The approximations produced by our technique have optimal exponential convergence when the parameters of the general non-polynomial functions have distributions that are stable across all iterations. We demonstrate the convergence of the PCE estimator to the true function in the L_1 sense for the case of unstable parameter distributions.

Our exact method applies to probabilistic loops with trigonometric and exponential assignments if the random perturbations of the arguments of the non-linear functions are independent across iterations.

Our methods can accommodate non-linear, non-polynomial updates in classes of probabilistic loops amenable to automated moment computation, such as the class of Prob-solvable loops. We emphasize that our PCE-based approximation is not limited to Prob-solvable loops and can be applied to approximate non-linear dynamics in more general probabilistic loops.

Our experiments demonstrate the ability of our methods to characterize non-polynomial behavior in stochastic models from various domains via their moments, with high accuracy and in a fraction of the time required by other state-of-the-art tools. In future work, we plan to investigate how to use these solutions to automatically compute stability properties (e.g. Lyapunov stability and asymptotic stability) in stochastic dynamical systems.

3 K-series for Moment-based Density Elicitation in Probabilistic Loops

This chapter is based on the following accepted paper: [KBB25]

- *Andrey Kofnov, Ezio Bartocci, and Efstathia Bura. 2025. Moment-based Density Elicitation with Applications in Probabilistic Loops. Accepted for publication in *ACM Trans. Probab. Mach. Learn.**

3.1 K-series

We develop the *K-series* estimation method to recover the joint and marginal distributions of a vector of random variables given a finite number of their moments. Our proposal generalizes the Gram-Charlier (GC) series to estimate an unknown pdf with bounded support. Both K-series and GC require a known reference distribution in order to derive the unknown continuous pdf. The normal reference pdf is instrumental in GC series as it dictates the choice of Hermite polynomials. Our approach allows using *any* continuous pdf provided its support covers the support of the target pdf we want to estimate. We present the univariate and its multivariate extension in Sections 3.1.1 and 3.1.6, respectively.

3.1.1 Univariate K-series

Let X be a continuous random variable, supported on an arbitrary interval $\Omega \subseteq \mathbb{R}$, with cumulative distribution function (cdf) $F_X(x)$ that is continuously differentiable on Ω and the corresponding pdf $f(x) = dF_X(x)/dx$ is non-negative upper bounded everywhere on Ω with countable zeros. Let $M = \{m_1, m_2, \ldots, m_n, \ldots\}$ be the set of all moments of the random variable X and suppose only the first n are known. We denote this finite subset of M by $M_n = \{m_1, \ldots, m_n\}, n \in \mathbb{N}$ and the vector with elements the moments in M_n by $\mathbf{m}_n = (1, m_1, \ldots, m_n)^T$. Boldface symbols denote vectors and matrices throughout the chapter.

From definition 2, a probability density function is said to be exponentially integrable, if there exists a positive $a > 0$ such that

$$\int_{\mathbb{R}} \exp\{a|x|\} f(x) dx < \infty.$$

For such pdfs, as well as distributions with bounded support, the probability distribution is fully characterizable by its moments.

Let $\phi(x)$ be an arbitrary continuous pdf that is positive everywhere on its support Θ, where $\Omega \subseteq \Theta$. We require either Θ be unbounded and $\phi(x)$ uniquely identifiable by its moments, or Θ be finite (bounded). Let $H = \{h_0(x), h_1(x), \ldots, h_n(x)\}$, $h_0(x) \equiv 1$ be a sequence of orthonormal polynomials on Θ with respect to $\phi(x)$; i.e.,

$$\langle h_i, h_j \rangle_\phi = \int_\Theta h_i(x) h_j(x) \phi(x) dx = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

A function $l(x)$ is said to belong to $L_p(\Sigma, \rho)$ if $\int_\Sigma |l(x)|^p \rho(x) dx < \infty$ (see [Rud86]). Throughout this chapter, f is used to denote the target and ϕ the reference pdf, respectively. Also, at least one of the following two assumptions is assumed to hold.

Assumption 2. *The support Ω of the pdf of X is a bounded set.*

Assumption 3. *The ratio $f(x)/\phi(x)$ is in $L_1(\Omega, f)$.*

We define $\widetilde{f}(x)$ on Θ to be

$$\widetilde{f}(x) = \begin{cases} f(x), & x \in \Omega, \\ 0, & x \in \Theta \setminus \Omega. \end{cases} \tag{3.1}$$

Since H is an orthonormal system on Θ with respect to pdf ϕ, any function in $L_2(\Theta, \phi)$ can be expanded into a Fourier series (see, e.g., [KF76] or [Rud76]) along the H basis elements. Under Assumption 2 or 3, $g(x) = \widetilde{f}(x)/\phi(x)$ satisfies

$$\int_\Theta g^2(x) \phi(x) dx = \int_{\Theta \setminus \Omega} \frac{\widetilde{f}(x)}{\phi(x)} \widetilde{f}(x) dx + \int_\Omega \frac{f(x)}{\phi(x)} dF_X(x) < \infty, \tag{3.2}$$

so that $g(x) \in L_2(\Theta, \phi)$. In consequence, g has a Fourier series representation

$$g(x) = \sum_{i=0}^{\infty} \alpha_i h_i(x), \tag{3.3}$$

with

$$\begin{aligned} \alpha_i = \langle g, h_i \rangle_\phi &= \int_\Theta g(x) h_i(x) \phi(x) dx = \int_\Theta \frac{\widetilde{f}(x)}{\phi(x)} h_i(x) \phi(x) dx \\ &= \int_\Theta \widetilde{f}(x) h_i(x) dx = \int_\Omega f(x) h_i(x) dx + \int_{\Theta/\Omega} \widetilde{f}(x) h_i(x) dx = \langle 1, h_i \rangle_f . \end{aligned}$$

The series in (3.3) converges in $L_2(\Theta, \phi)$. From (3.1) and (3.3), an estimator of f is

$$\hat{f}(x) = \phi(x) \sum_{i=0}^{n} \langle 1, h_i \rangle_f h_i(x). \tag{3.4}$$

Each polynomial $h_i(x)$ is a sum of monomials, $h_i(x) = \sum_{j=0}^{i} a_{ij}x^j, i = 0, \ldots, n$. Since the first n moments of $f(x)$ are known,

$$\langle 1, h_i \rangle_f = \sum_{j=0}^{i} a_{ij} \langle 1, x^j \rangle_f = \sum_{j=0}^{i} a_{ij} m_j, \tag{3.5}$$

where m_j is the jth *raw* moment of X for $j = 0, \ldots, i$, $i = 0, \ldots, n$.

Definition 5. *The series-based estimator* (3.4) *of the pdf f of X is called a K-series estimator with reference ϕ, or simply K-series.*

Let $\mathbf{A} = \{a_{ij}\}_{i,j=0}^{n}$ be a lower triangular matrix with entries the coefficients of the *ordered* vector of polynomials $h_i(x)$, $\mathbf{h}_n(x) = (h_0(x), \ldots, h_n(x))^T$ from H. Then, (3.4) can also be computed by

$$\hat{f}(x) = \phi(x) \left(\mathbf{A} \cdot \mathbf{m}_n\right)^T \cdot \mathbf{h}_n(x). \tag{3.6}$$

The only requirements for the *K-series* estimator are (a) the unknown target pdf f have bounded support and (b) the support of the reference ϕ be large enough to cover it. The only constraint for the choice of the reference distribution is to be continuous with support larger than that of the target pdf. Any such pdf can serve as a reference and thus polynomials h_i of any order can be computed using the Gram-Schmidt orthogonalization procedure in (3.4).

There is no technical necessity to strictly adhere to the ordered sequence of the sequence of moments. It is possible to use moments of any order. What is necessary is to generate orthogonal polynomials in equation (3.4) in a specific sequence. For example, if moments of order 1, 5, and 12 are available, we can construct a system of orthogonal polynomials from the monomial set $\{1, x, x^5, x^{12}\}$ by the Gram-Schmidt process.

3.1.2 K-series estimation in practice

We illustrate K-series estimation with two examples. For the first, we let the target pdf be truncated exponential with known parameters and support and derive its first two moments and its K-series estimate. In the second (Irwin-Hall Distribution), we express the distribution generating algorithm as a prob-solvable loop, compute its *exact* moments using the POLAR tool [MSB+22] and then its K-series estimate.

Truncated Exponential. Suppose $X \sim Trunc\ Exp(1, [0, 1])$ with support $\Omega = [0, 1]$. We assume the first two moments are known, specifically, we let $M_2 = \{m_1 = 0.418023, m_2 = 0.254070\}$, and the reference distribution is uniform with the same support as the target unknown distribution; i.e., $\phi(x) = 1$ for $x \in [0, 1]$.

Legendre polynomials $l_n(\tau)$ are a standard basis of orthogonal polynomials on the interval [-1,1] with a weight function of 1. Consequently, for any uniform pdf on an arbitrary bounded interval, a corresponding set of orthonormal polynomials can be derived from the standard Legendre polynomials through the substitution $\tau \to (\tau - \mu)/\sigma$ and subsequent normalization.

Since ϕ is uniform, we use the shifted and scaled Legendre polynomials as the orthonormal basis in the series (see [XK02]); $\bar{l}_0 = 1$, $\bar{l}_1 = \sqrt{3}(2x - 1)$, $\bar{l}_2 = \sqrt{5}(6x^2 - 6x + 1)$. To compute the unknown pdf estimator in (3.4), we need to compute the α_i coefficients in (3.3). By (3.5), this requires the substitution of x^i with the corresponding moment m_i in M_2, for $i = 1, 2$. Doing so

yields $\alpha_0 = 1, \alpha_1 = \sqrt{3}(2{\cdot}0.418023{-}1) = -0.283976, \alpha_2 = \sqrt{5}(6{\cdot}0.25407{-}6{\cdot}0.418023{+}1) = 0.036407$. The K-series estimator is

$$\hat{f}(x) = 1 - 0.283976 \cdot \bar{l}_1(x) + 0.036407 \cdot \bar{l}_2(x),$$

and almost fully coincides with the true truncated exponential pdf in panel (a) of Figure 3.1.

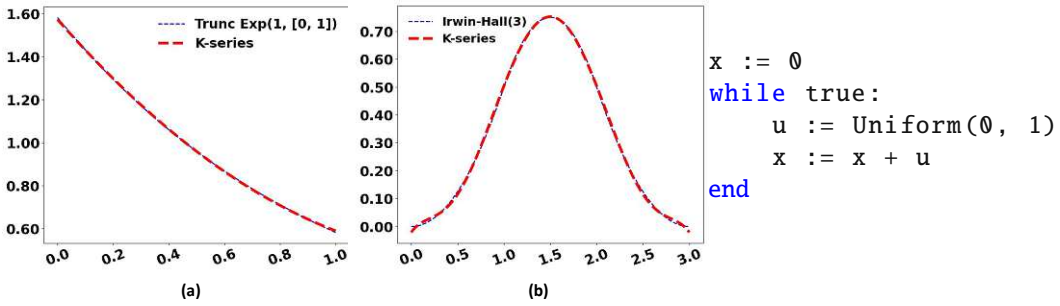


Figure 3.1: K-series approximation of a truncated exponential distribution (panel (a)) and the Irwin-Hall distribution (panel (b)).

The Irwin-Hall Distribution. Irwin–Hall is the probability distribution of a sum of independent uniform random variables on the unit interval (uniform sum distribution). That is, $X \sim$ Irwin–Hall(t) if $X = \sum_{i=1}^{t} U_i$, for U_i independent and identically distributed (i.i.d.) as Uniform(0,1). This distribution, parameterized by the number of its summands, is encodable as the prob-solvable loop in the right panel of Fig. 3.1.

At each iteration t, the support of x is $(0, t)$. Since the Irwin-Hall distribution is equivalent to a prob-solvable loop, its exact n first moments can be computed with the algorithm in [BKS19]:

$$M(t) = \left\{ \frac{t}{2}, \frac{t(3t+1)}{12}, \frac{t^2(t+1)}{8}, \frac{t(15t^3+30t^2+5t-2)}{240}, \right.$$
$$\left. \frac{t^2(3t^3+10t^2+5t-2)}{96}, \frac{t(63t^5+315t^4+315t^3-91t^2-42t+16)}{4032}, \dots \right\}.$$

The first 6 moments of Irwin-Hall (3) are $M_6(3) = \left\{\frac{3}{2}, \frac{5}{2}, \frac{9}{2}, \frac{43}{5}, \frac{69}{4}, \frac{3025}{84}\right\}$. We use the Uniform$[0, 3]$ as a reference and construct the K-series estimator of the pdf of x at iteration $t = 3$ with the 6 first moments and the first 7 shifted and scaled Legendre polynomials. The true pdf and its K-series estimate are plotted in panel (b) of Fig. 3.1, where we can see their almost perfect agreement.

While iteration $t = 3$ is used for illustration purposes, iteration number is not important for our method. One only needs to specify an appropriate reference and support (for the uniform reference the support is $[0, t]$). Alternatively, we can use a reference that has the appropriate support for any iteration; normal, truncated normal, gamma, etc.

3.1.3 Special cases of K-series

The K-series density estimator generalizes the widely used Gram-Charlier (GC) series density estimator. GC represents the pdf f of a random variable X as a series in terms of its cumulants and a normal reference distribution ϕ by using Hermite polynomials (see, e.g., [Cra57; KS77]). The GC (type-)A estimate of the pdf f of X is given by

$$f_{\mathrm{GC}}(x) = \phi(x) \sum_{n=0}^{\infty} (-1)^n c_n He_n(x), \tag{3.7}$$

where $c_n = (-1)^n \int_{-\infty}^{\infty} f(t) He_n(t) dt / n!$, ϕ is the standard normal pdf and

$$He_n(x) = n! \sum_{k=0}^{[n/2]} \left((-1)^k x^{n-2k} \right) / \left(k!(n-2k)! 2^k \right)$$

Proposition 2 shows that the GC series A estimator in (3.7) is a special case of the K-series estimator.

Proposition 2. *Suppose the reference pdf ϕ is normal with mean and variance corresponding to the first and second moments of the target pdf f. Then, the K-series estimator* (3.4) *equals the Gram-Charlier estimator* (3.7)*.*

Proposition 2 is easy to obtain using the standard normal as reference pdf and replacing the polynomials h_i in (3.4) by $He_i / \sqrt{i!}$.

[MMR17] developed the *Method of Moments (MM)* estimation algorithm for parameters of a target distribution f by equating sample moments with the corresponding moments of the distribution. The approximation is carried out on the interval where they wish to maximize accuracy. In practice, this is the same as assuming finite or bounded support. [MMR17] showed that MM beats the GC expansion for several distributions, such as the Weibull on a positive finite support, in simulation experiments.

The MM algorithm starts by choosing an interval $[a, b]$ that is thought to contain most of the mass of the target unknown distribution. Using a finite set of moments $\{m_1, \ldots, m_n\}$ and $m_0 = 1$, MM constructs a polynomial estimator $\hat{f}(x)$ by solving a linear system of equations,

$$m_i = \int_a^b x^i \hat{f}(x) dx, \; i = 0, \ldots, n, \tag{3.8}$$

which yields the coefficients p_i of the series representation $\hat{f}_{\mathrm{MM}}(x) = \sum_{i=0}^{n} p_i x^i$.

Let $\mathbf{m}_n = (1, m_1, \ldots, m_n)^T$, $\mathbf{p}_n = (p_0, p_1, \ldots, p_n)^T$, and $\mathbf{x}_n = (1, x, \ldots, x^n)^T$. The linear system (3.8) can be expressed in matrix form as $\mathbf{m}_n = \mathbf{M}_{ab} \cdot \mathbf{p}_n$, where $\mathbf{M}_{ab}$ is the matrix with elements the integrals of powers of x over the interval $[a, b]$. Theorem 2 shows that MM is a special case of the K-series estimator.

Theorem 2. *Suppose the reference pdf ϕ is the uniform with the same support as the target pdf f. Then, the MM estimator coincides with the K-series estimator* (3.4)*.*

Proof. Suppose f is supported on (a, b). Then, $\phi(x) = \phi = 1/(b-a)$. Let $\{l_j(x) = \sum_{i=0}^{j} \lambda_{ji} x^i\}_{j=0}^{n}$ be the set of the first n shifted scaled Legendre polynomials that are orthonormal on $[a, b]$, so that $\mathbf{\Lambda} = (\lambda_{ji})_{j,i=0}^{n}$ is a lower triangular matrix.

Every polynomial of degree n can be expressed as a weighted sum of polynomials of degree up to n. In such a case, we can represent the MM estimator $\hat{f}_{\text{MM}} = \sum_{i=0}^{n} p_i x^i$ as a weighted sum of Legendre polynomials $1, l_1(x), \ldots, l_n(x)$ with weight coefficients $\phi \cdot a_j$, $j = 0, \ldots, n$. Then,

$$\hat{f}_{\text{MM}}(x) = \sum_{i=0}^{n} p_i x^i = \phi \sum_{j=0}^{n} a_j l_j(x) = \phi \sum_{j=0}^{n} a_j \sum_{i=0}^{j} \lambda_{ji} x^i,$$

or, equivalently, $\mathbf{p}_n^T \mathbf{x}_n = \phi \cdot \mathbf{a}_n^T \mathbf{l}_n = \phi \cdot \mathbf{a}_n^T \mathbf{\Lambda} \mathbf{x}_n$, where $\mathbf{l}_n = (1, l_1(x), \ldots, l_n(x))^T$ and $\mathbf{a}_n = (1, a_1, \ldots, a_n)^T$. Thus, $\mathbf{p}_n = \phi \cdot \mathbf{\Lambda}^T \mathbf{a}_n$. Now,

$$\mathbf{m}_n = \mathbf{M}_{ab} \cdot \mathbf{p}_n, \tag{3.9}$$

implies $\mathbf{m}_n = \mathbf{M}_{ab} \cdot \mathbf{p}_n = \phi \cdot \mathbf{M}_{ab} \cdot \mathbf{\Lambda}^T \mathbf{a}_n$, where $\mathbf{M}_{ab}$ is the matrix with elements the integrals of powers of x over the interval $[a, b]$,

$$\mathbf{M}_{ab} = \begin{pmatrix} b-a & \frac{b^2-a^2}{2} & \cdots & \frac{b^{n+1}-a^{n+1}}{n+1} \\ \frac{b^2-a^2}{2} & \frac{b^3-a^3}{3} & \cdots & \frac{b^{n+2}-a^{n+2}}{n+2} \\ \vdots & & \ddots & \\ \frac{b^{n+1}-a^{n+1}}{n+1} & \frac{b^{n+2}-a^{n+2}}{n+2} & \cdots & \frac{b^{2n+1}-a^{2n+1}}{2n+1} \end{pmatrix}. \tag{3.10}$$

In the matrix form of the K-series estimator (3.6), $\mathbf{\Lambda} \cdot \mathbf{m}_n = \mathbf{a}_n$. It suffices to show that

$$\phi \cdot \mathbf{M}_{ab} \cdot \mathbf{\Lambda}^T = \mathbf{\Lambda}^{-1}. \tag{3.11}$$

The matrix $\phi \cdot \mathbf{M}_{ab}$ contains the moments of the uniform distribution. Therefore, $\mathbf{\Lambda}$ is a matrix with entries the coefficients of orthonormal polynomials and the left lower triangular factor of the Cholesky decomposition of the moment matrix $\phi \cdot \mathbf{M}_{ab}$. Thus, (3.11) follows from [Sza15, Prop. 2(i)]. □

MM and GC are special cases of K-series estimation. As such, they also enjoy the theoretical properties of K-series in the constrained setting in which they apply. We next show in Theorem 3 that the general K-series estimator (3.4) converges to the true target pdf.

Theorem 3. *Let $\phi(x)$ be continuous, positive everywhere on Θ: $\Omega \subseteq \Theta$ and either (a) Θ is unbounded and $\phi(x)$ is uniquely identifiable by its moments, or (b) Θ is finite (bounded). Under Assumption 2 or 3, the K-series estimator* (3.4) *converges to the true pdf* (3.1)*, $\widetilde{f}(x)$, in $L_1(\Theta, 1)$. Moreover, if $\phi(x)$ is a uniform pdf, it converges in $L_2(\Theta, 1)$.*

Proof.

$$\begin{aligned}
\left\| \frac{\widetilde{f}(x)}{\phi(x)} - \sum_{i=0}^{n} \alpha_i h_i(x) \right\|_{\phi}^2 &= \int_{\Theta} \left[\frac{\widetilde{f}(x)}{\phi(x)} - \sum_{i=0}^{n} \alpha_i h_i(x) \right]^2 \phi(x) dx \\
&= \int_{\Theta} \left[\widetilde{f}(x) - \phi(x) \sum_{i=0}^{n} \alpha_i h_i(x) \right]^2 \frac{1}{\phi(x)} dx \\
&= \int_{\Theta} \phi(x) dx \int_{\Theta} \left[\widetilde{f}(x) - \phi(x) \sum_{i=0}^{n} \alpha_i h_i(x) \right]^2 \frac{1}{\phi(x)} dx \\
&= ||\sqrt{\phi(x)}||_1^2 \cdot \left\| \left(\widetilde{f}(x) - \phi(x) \sum_{i=0}^{n} \alpha_i h_i(x) \right) \frac{1}{\sqrt{\phi(x)}} \right\|_1^2 \\
&\geq \left(\int_{\Theta} \left| \widetilde{f}(x) - \phi(x) \sum_{i=0}^{n} \alpha_i h_i(x) \right| dx \right)^2 , \qquad (3.12)
\end{aligned}$$

where the last inequality is due to Cauchy-Schwarz inequality. The function $\widetilde{f}(x)$ in (3.1) is a density. In the case where Θ is bounded, $\phi(x)$ is uniquely identifiable by its moments. When Θ is unbounded, $\phi(x)$ is exponentially integrable by the assumption (a) of the theorem. Hence, for all $n \geq 1$, $\phi(x) \left| \sum_{i=0}^{n} \alpha_i h_i(x) \right|$ is integrable.

Since the truncated series $\sum_{i=0}^{n} \alpha_i h_i(x)$ converges to $g(x) = \widetilde{f}(x)/\phi(x)$ in $L_2(\Theta, \phi)$, as $n \to \infty$, from (3.12) we obtain that the K-series estimator (3.4) converges to the extended true pdf $\widetilde{f}(x)$ in $L_1(\Theta, 1)$.

Next, suppose $\phi(x)$ is the pdf of the uniform distribution, so that Θ is bounded, and $\phi(x) = c$. Then,

$$\left\| \frac{\widetilde{f}}{\phi} - \sum_{i=0}^{n} \alpha_i h_i(x) \right\|_{\phi}^2 = \int_{\Theta} \left[\frac{\widetilde{f}(x)}{\phi(x)} - \sum_{i=0}^{n} \alpha_i h_i(x) \right]^2 \phi(x) dx = \frac{1}{c} \int_{\Theta} \left[\widetilde{f}(x) - c \sum_{i=0}^{n} \alpha_i h_i(x) \right]^2 dx$$

Hence, $\widetilde{f}(x) = c \cdot g(x)$ is in $L_2(\Theta, 1)$, and $\int_{\Theta} c^2 \left[\sum_{i=0}^{n} \alpha_i h_i(x) \right]^2 dx$ is an integral of a polynomial over a bounded interval, so that the K-series estimator (3.4) converges to the true pdf $\widetilde{f}(x)$ in $L_2(\Theta, 1)$. □

The following theorem provides formal guarantees that the moments of the obtained estimate coincide with the corresponding moments of the target random variable based on which the estimate is constructed.

Theorem 4 (Moment matching)**.** *Suppose the K-series estimator* (3.4) *is constructed using the first* n *moments* $\{m_1, \ldots, m_n\}$ *of the random variable* X *with pdf* $f(x)$ *and set* $m_0 = 1$*. Then,*

$$\int_{\Theta} x^i \hat{f}(x) dx = \int_{\Omega} x^i f(x) dx = m_i,$$

for all $0 \leq i \leq n$.

Proof. Let $h_i(x)$ be the ith orthonormal polynomial with respect to the reference pdf $\phi(x)$ in (3.4). Then, $\int_\Omega h_i(x)f(x)dx = \alpha_i$. Also, by the orthogonality of h_is, the following holds

$$\int_\Theta h_i(x)\hat{f}(x)dx = \int_\Theta h_i(x)\phi(x)\sum_{j=0}^{n}\alpha_j h_j(x)dx = \sum_{j=0}^{n}\alpha_j \int_\Theta h_i(x)\phi(x)h_j(x)dx = \alpha_i.$$

It remains to observe that any monomial x^i, $0 \leq i \leq n$, can be expressed as a linear combination of the orthogonal polynomials h_j, $0 \leq j \leq i$. □

3.1.4 Approximation of the support

The space spanned by $\lfloor (n+1)/2 \rfloor$ orthogonal polynomials with respect to the target density $f(x)$ can be constructed using the sequence of its first n moments (see [Sze39]). The determinant

$$D_j(x) = \begin{vmatrix} m_0 & m_1 & \dots & m_j \\ m_1 & m_2 & \dots & m_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{j-1} & m_n & \dots & m_{2j-1} \\ 1 & x & \dots & x^j \end{vmatrix} \tag{3.13}$$

defines the corresponding orthogonal polynomial (non-normalized) of degree j. That is, if the first n moments of a random variable X are known, then we can construct the first $\lfloor (n+1)/2 \rfloor$ orthogonal polynomials.

We let $e_j(x)$ denote an orthogonal polynomial of degree j of the random variable X. Theorems 5, 6 and 7 (see [Sze39; Chi78]) state elementary properties of zeros of orthogonal polynomials.

Theorem 5. *Let Ω be an interval which is a supporting set for the distribution of X. The zeros of $e_j(x)$ are all real, simple and are located in Ω.*

Theorem 6. *Between two zeros of $e_j(x)$ there is at least one zero of $e_i(x)$, $i > j$.*

Theorem 7. *The zeros $\{x_{j,\nu}\}_{\nu=1}^{j}$ and $\{x_{j+1,\nu}\}_{\nu=1}^{j+1}$ of $e_j(x)$ and $e_{j+1}(x)$ respectively, mutually separate each other. That is,*

$$x_{j+1,\nu} < x_{j,\nu} < x_{j+1,\nu+1}, \qquad \nu = 1, \dots, j$$

From Theorems 5, 6 and 7, we can conclude that all zeros of orthogonal polynomials are simple and located precisely within the interior of the support. Moreover, as the polynomials' degree increases, the distance between the two outermost zeros also increases, resulting in a more accurate inner approximation of the random variable's support. The higher number of moments available, the higher the degree of polynomials that can be obtained, and the more accurate the estimation of the support becomes. One only needs to calculate the polynomial of the highest possible degree using formula (3.13), determine its zeros, and identify the lowest and highest values.

We demonstrate this method using the example of the Irwin-Hall distribution in Section 3.1.2. Let us suppose, that the first 6 moments of the random variable X are available:

$$M_6 = \left\{ \frac{3}{2}, \frac{5}{2}, \frac{9}{2}, \frac{43}{5}, \frac{69}{4}, \frac{3025}{84} \right\}.$$

We are interested in the minimum possible support of the pdf of X. Since the first 6 moments are known, we can construct orthogonal polynomials of the random variable X up to degree $\lfloor (n+1)/2 \rfloor = 3$. Applying (3.13) yields the highest degree computable polynomial,

$$D_3(x) = \begin{vmatrix} 1.00 & 1.50 & 2.50 & 4.50 \\ 1.50 & 2.50 & 4.50 & 8.60 \\ 2.50 & 4.50 & 8.60 & 17.25 \\ 1 & x & x^2 & x^3 \end{vmatrix} = 0.025x^3 - 0.1125x^2 + 0.1525x - 0.06. \tag{3.14}$$

The polynomial in (3.14) has 3 distinct roots: $\{0.693774, 1.5, 2.306226\}$. Since all the roots belong to the interior of the support, the inner approximation of the support is $[0.693774, 2.306226]$.

3.1.5 Validity of the input

Not every sequence of real values can form a valid set of moments for any probability distribution. This issue is known as the Hamburger moment problem (see [Chi78]). Given a sequence of real numbers $\{m_i\}_{i=0}^{\infty}$, the question is whether there exists a positive Borel measure F such that

$$\int_{-\infty}^{\infty} x^i dF(x) = m_i, \qquad i = 0, 1, 2, \ldots$$

We introduce a procedure to examine whether the input set of values can be moments of a distribution. We require the input sequence of moments to be consecutive and without gaps. Since we are dealing with a truncated set of moments, we refer to it as the *truncated moment problem*. Let

$$\Delta_r = \det(m_{i+j})_{i,j=0}^{r} = \begin{vmatrix} m_0 & m_1 & \ldots & m_r \\ m_1 & m_2 & \ldots & m_{r+1} \\ \vdots & \vdots & \ddots & \vdots \\ m_r & m_{r+1} & \ddots & m_{2r} \end{vmatrix} \tag{3.15}$$

be a sequence of determinants.

Theorem 8. *[Chi78] The Hamburger moment problem has a solution if and only if the determinants* Δ_r *in* (3.15) *are all positive.*

By Theorem 8, the truncated moment problem admits a solution only if all the determinants Δ_r, $r = 0, \ldots, \lfloor n/2 \rfloor$, are positive. The complete process of univariate K-series estimation is described in Algorithm 1.

3.1.6 Multivariate K-series

K-series density estimation is easily generalizable to multivariate distributions by considering the product of independent univariate distributions as the reference joint pdf. The coefficients of the corresponding multivariate orthogonal polynomials recover the multivariate dependence structure via their joint moments.

Let $\mathbf{X} = (X_1, \ldots, X_k)^T$ be a vector of continuous random variables with joint non-negative pdf $f(\mathbf{x})$, upper bounded and supported on Ω with countable zeros. Suppose that there exists a k-dimensional compact cube $\mathcal{Q}$, such that $\Omega \subseteq \mathcal{Q}$. We assume that a finite number of moments, not necessarily an equal number for all, is known for each X_j, $j = 1, \ldots, k$, and all cross-product moments are also known. That is, we assume the set

$$M_{d_1,\ldots,d_k} = \left\{ m_{i_1,\ldots,i_k} = \mathbb{E}\left(X_1^{i_1} \ldots X_k^{i_k} \right) : i_j = 0, \ldots, d_j, d_j \in \mathbb{N}, j = 1, \ldots, k \right\} \tag{3.16}$$

is known. Let $\mathbf{Z} = (Z_1, \ldots, Z_k)^T$ be a vector of continuous independent random variables and $\widetilde{\phi}(\mathbf{z}) = \prod_{j=0}^{k} \phi_j(z_j)$ be its pdf that is positive everywhere on its support Θ, where $\Omega \subseteq \Theta$. We require either Θ be unbounded and $\widetilde{\phi}(\mathbf{z})$ uniquely identifiable by its moments, or Θ be bounded (see [Rah18]).

Let

$$\tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) = \prod_{j=1}^{k} h_{i_j}^j(z_j), \tag{3.17}$$

where $h_{i_j}^j(z_j)$ is a polynomial of degree i_j that belongs to the set of orthogonal polynomials with respect to $\phi_j(z_j)$, $i_j = 0, \ldots, d_j$, $j = 1, \ldots, k$, that are calculated with the Gram-Schmidt orthogonalization procedure. The set $H = \{\tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}), i_j = 0, \ldots, d_j, d_j \in \mathbb{N}, j = 1, \ldots, k\}$ contains the k-variate orthonormal polynomials on Θ with respect to $\widetilde{\phi}(\mathbf{z})$. As in the univariate case, we require Assumption 2 hold and let

$$\widetilde{f}(\mathbf{z}) = \begin{cases} f(\mathbf{z}), & \mathbf{z} \in \Omega, \\ 0, & \mathbf{z} \in \Theta \setminus \Omega. \end{cases} \tag{3.18}$$

Then, $\widetilde{f}(\mathbf{z})/\widetilde{\phi}(\mathbf{z}) = g(\mathbf{z})$ is approximated by

$$\hat{g}(\mathbf{z}) = \sum_{\substack{i_j \in \{0,\ldots,d_j\}, \\ j=1,\ldots,k}} \alpha(i_1, \ldots, i_k) \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) = \sum_{\substack{i_j \in \{0,\ldots,d_j\}, \\ j=1,\ldots,k}} \alpha(i_1, \ldots, i_k) \prod_{j=1}^{k} h_{i_j}^j(z_j),$$

Algorithm 1: Univariate K-series procedure

Input:

- $\{m_i\}_{i=0}^n$ - sequence of n moments, $m_0 = 1$
- $\phi(x)$ - reference pdf
- Θ - support of the reference

Output:

- *True* / *False* - is the sequence $\{m_i\}_{i=0}^n$ feasible?
- $[root_{min}, root_{max}]$- inner approximation of the support
- $\hat{f}(x) = \phi(x) \sum_{i=0}^{n} \langle 1, h_i \rangle_f \, h_i(x)$- K-series estimator

Compute: Determinants Δ_r according to (3.15), $0 \leq r \leq \lfloor n/2 \rfloor$.
if $\exists r$: $\Delta_r \leq 0$ **then**
  return: *False*
end
`/* Approximation of the support */`
Compute: Orthogonal polynomial $e_{\lfloor (n+1) \rfloor /2}$ of the highest degree using (3.13).
Search for: The lowest and the highest roots of $e_{\lfloor (n+1) \rfloor /2}$: $\{root_{min}, root_{max}\}$
`/* Orthogonal Polynomials Construction: */`
$h_0(x) = 1$
forall i *in* $\{1, 2, \ldots, n\}$ **do**
  `/* Gram-Schmidt Orthogonalization */`
  $\widetilde{h}_i(x) = x^i - \sum_{j=0}^{i-1} \frac{\langle x^i, h_j(x) \rangle_\phi}{\langle h_j(x), h_j(x) \rangle_\phi}$;
  $h_i(x) = \widetilde{h}_i(x) / \parallel \widetilde{h}_i(x) \parallel_\phi$;
end
forall *polynomial* h_i *in* $\{h_1, \ldots, h_n\}$ **do**
  forall *monomial* x^j *in* $h_i(x) = \sum_{j=0}^{i} a_{ij} x^j$ **do**
    Substitute: $x^j \leftarrow m_j$
  end
  Compute: Fourier coefficients $\alpha_i = \langle h_i(x), 1 \rangle_f = \sum_{j=0}^{i} a_{ij} m_j$
end
Compute: $\hat{f}(x) = \phi(x) \sum_{i=0}^{n} \langle 1, h_i \rangle_f \, h_i(x)$
return: *True*, $[root_{min}, root_{max}]$, $\hat{f}(x)$

where the Fourier coefficients $\alpha(i_1, \ldots, i_k)$ are calculated as

$$\begin{aligned}\alpha(i_1, \ldots, i_k) &= \left\langle g, \tilde{h}_{i_1,\ldots,i_k} \right\rangle_{\widetilde{\phi}} = \int_{\Theta} g(\mathbf{z}) \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) \widetilde{\phi}(\mathbf{z}) d\mathbf{z} \\ &= \int_{\Omega} f(\mathbf{z}) \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) d\mathbf{z} + \int_{\Theta/\Omega} \widetilde{f}(\mathbf{z}) \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) d\mathbf{z} \\ &= \left\langle 1, \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) \right\rangle_f . \end{aligned} \tag{3.19}$$

Since for all $i_j = 0, \ldots, d_j, j = 1, \ldots, k$, $h^j_{i_j}(z_j) = \sum_{l=0}^{i_j} a^j_{i_j l} z^l_j$, their product is

$$\tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) = \prod_{j=1}^{k} h^j_{i_j}(z_j) = \prod_{j=1}^{k} \sum_{l=0}^{i_j} a^j_{i_j l} z^l_j = \sum_{\substack{l_j \in \{0,\ldots,i_j\}, \\ j=1,\ldots,k}} z_1^{l_1} \cdots z_k^{l_k} \prod_{j=1}^{k} a^j_{i_j l_j}.$$

Assuming all first cross-moments of $f(\mathbf{z})$, $m_{l_1,\ldots,l_k} = \mathbb{E}_f\left(Z_1^{l_1} \cdots Z_k^{l_k}\right)$ are known, we can compute (3.19) as

$$\left\langle 1, \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) \right\rangle_f = \sum_{\substack{l_j \in \{0,\ldots,i_j\}, \\ j=1,\ldots,k}} \left\langle 1, z_1^{l_1} \cdots z_k^{l_k} \right\rangle_f \prod_{j=1}^{k} a^j_{i_j l_j} = \sum_{\substack{l_j \in \{0,\ldots,i_j\}, \\ j=1,\ldots,k}} m_{l_1,\ldots,l_k} \prod_{j=1}^{k} a^j_{i_j l_j}. \tag{3.20}$$

The *multivariate K-series* estimator of f is

$$\hat{f}(\mathbf{x}) = \widetilde{\phi}(\mathbf{x}) \sum_{\substack{i_j \in \{0,\ldots,d_j\}, \\ j=1,\ldots,k}} \left\langle 1, \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}) \right\rangle_{\mathbf{f}} \tilde{h}_{i_1,\ldots,i_k}(\mathbf{z}). \tag{3.21}$$

A probabilistic loop application of K-series estimation is shown in Fig. 1.5, where the pdf of the location (X, Y) of the *Differential-Drive Mobile Robot* is estimated, assuming it is characterizable by its moments. The joint and marginal distributions of the location variables X and Y are derived from a finite set of moments at iteration $t = 25$. The value of 25 was chosen to provide a non-trivial example of the capabilities of our approach. Moreover, it serves as a juxtaposition to competing tools (such as λPSI [GSV20]), which fail to generate a meaningful expression even by iteration 5. We use the first 6 moments for the marginals and the first 48 moments for the joint distribution. The middle and right top panels of Fig. 1.5) plot the marginal pdfs of X and Y, respectively. The histograms are based on 10^6 draws from the true marginals. Our K-series estimates are in dashed red and agree almost perfectly with the true marginal pdfs. The left bottom panel plots the estimated joint pdf of (X, Y). The right panel draws comparative frequency bar plots of 10^6 true and estimated values of the bivariate random variable (X, Y) over 2-dimensional grids of the support of the bivariate distribution. Our estimate (red bars) practically coincides with the true joint pdf (blue bars) over the grid.

We provide another illustration of this algorithm on the truncated bivariate normal.

Suppose we want to recover the joint pdf of two random variables X and Y on a set $\Omega = [-2,2] \times [-4,5]$ using their first eight cross-moments,

$$\left(m_{x^j y^i} = \mathbb{E}(X^j Y^i)\right)_{i,j=0,\dots,2} = \begin{pmatrix} m_{x^0y^0} & m_{x^1y^0} & m_{x^2y^0} \\ m_{x^0y^1} & m_{x^1y^1} & m_{x^2y^1} \\ m_{x^0y^2} & m_{x^1y^2} & m_{x^2y^2} \end{pmatrix}$$

$$= \begin{pmatrix} 1.00000 & 0.71721 & 1.13054 \\ 1.99556 & 1.43124 & 2.25606 \\ 4.96894 & 3.56379 & 5.61757 \end{pmatrix}. \tag{3.22}$$

We choose the reference marginal pdfs be both truncated normal $\phi_x(z_x)$ and $\phi_y(z_y)$ with $Z_x \sim Trunc\,\mathcal{N}(m_x, m_{x^2} - m_x^2, [-2,2]) = Trunc\,\mathcal{N}(0.71721, 0.61614, [-2,2])$, and $Z_y \sim Trunc\,\mathcal{N}(m_y, m_{y^2} - m_y^2, [-4,5]) = Trunc\,\mathcal{N}(1.99556, 0.98667, [-4,5])$, respectively.

We construct sets of univariate orthonormal polynomials using, for example, the Gram-Schmidt orthogonalization procedure, and obtain

$$\begin{aligned} h_0^x(z_x) &= 1, & h_0^y(z_y) &= 1, \\ h_1^x(z_x) &= 1.42119 z_x - 0.89705, & h_1^y(z_y) &= 1.01307 z_y - 2.01751, \\ h_2^x(z_x) &= 1.58907 z_x^2 - 1.63885 z_x - 0.38542, & h_2^y(z_y) &= 0.74083 z_y^2 - 2.92557 z_y + 2.16624 \end{aligned}$$

Hence, starting from a reference joint pdf that is the product of the pdfs of the independent random variables Z_x and Z_y, $\widetilde{\phi}(z_x, z_y) = \phi_x(z_x)\phi_y(z_y)$, the multivariate orthogonal polynomials are simply all the pairwise products of univariate polynomials:

$$\begin{aligned} \tilde{h}_{0,0}(z_x, z_y) &= 1 \\ \tilde{h}_{0,1}(z_x, z_y) &= 1.01307 z_y - 2.01751 \\ \tilde{h}_{0,2}(z_x, z_y) &= 0.74083 z_y^2 - 2.92557 z_y + 2.16624 \\ \tilde{h}_{1,0}(z_x, z_y) &= 1.42119 z_x - 0.89705 \\ \tilde{h}_{1,1}(z_x, z_y) &= 1.43976 z_x z_y - 2.86727 z_x - 0.90877 z_y + 1.80981 \\ \tilde{h}_{1,2}(z_x, z_y) &= 1.05286 z_x z_y^2 - 4.15779 z_x z_y + 3.07864 z_x - 0.66456 z_y^2 + 2.62438 z_y \\ &\quad - 1.94323 \\ \tilde{h}_{2,0}(z_x, z_y) &= 1.58907 z_x^2 - 1.63885 z_x - 0.38542 \\ \tilde{h}_{2,1}(z_x, z_y) &= 1.60984 z_x^2 z_y - 3.20596 z_x^2 - 1.66027 z_x z_y + 3.30634 z_x - 0.39046 z_y \\ &\quad + 0.77759 \\ \tilde{h}_{2,2}(z_x, z_y) &= 1.17723 z_x^2 z_y^2 - 4.64894 z_x^2 z_y + 3.44231 z_x^2 - 1.21411 z_x z_y^2 \\ &\quad + 4.79457 z_x z_y - 3.55014 z_x - 0.28553 z_y^2 + 1.12757 z_y - 0.83491 \end{aligned}$$

In order to compute the coefficients $\alpha(i_1, i_2)$ of the PCE along the reference pdf $\widetilde{\phi}(z_x, z_y)$ for each polynomial $\tilde{h}_{i_1,i_2}(z_x, z_y)$, we need to substitute every monomial factor $z_x^j z_y^i$ by the corresponding moment $m_{x^j y^i}$ from (3.22) in each polynomial. For example, the coefficient of $\tilde{h}_{1,1}(z_x, z_y)$ is

$1.43976m_{xy} - 2.86727m_x - 0.90877m_y + 1.80981 = 1.43976 \cdot 1.43124 - 2.86727 \cdot 0.71721 - 0.90877 \cdot 1.99556 + 1.80981 = 0.00051$. The resulting estimator is

$$\begin{aligned}\hat{f}(z_x, z_y) &= \phi_1(z_x)\phi_2(z_y) \sum_{i_1,i_2=(0,0)}^{(2,2)} \alpha(i_1, i_2)\tilde{h}_{i_1,i_2}(z_x, z_y) \\ &= \phi_1(z_x)\phi_2(z_y) \times \Big[1 + 0.00415 \cdot \tilde{h}_{0,1}(z_x, z_y) + 0.00924 \cdot \tilde{h}_{0,2}(z_x, z_y) \\ &\quad + 0.12224 \cdot \tilde{h}_{1,0}(z_x, z_y) + 0.00051 \cdot \tilde{h}_{1,1}(z_x, z_y) + 0.00113 \cdot \tilde{h}_{1,2}(z_x, z_y) \\ &\quad + 0.23568 \cdot \tilde{h}_{2,0}(z_x, z_y) + 0.00098 \cdot \tilde{h}_{2,1}(z_x, z_y) + 0.00218 \cdot \tilde{h}_{2,2}(z_x, z_y)\Big]\end{aligned}$$

The estimated bivariate density is plotted in Fig. 3.2 (a). In panel (b), we plot the frequencies of X and Y under the true $f(x, y)$ (blue bars) and its K-series (red bars) pdf estimate over a 2D grid comprising of eight parallelograms, where we can see their close agreement.

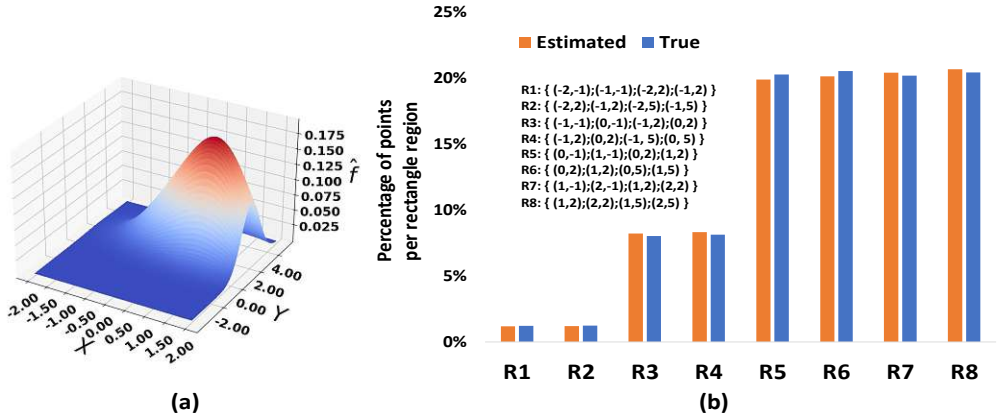


Figure 3.2: K-series estimates of the truncated bivariate normal distribution $f(x, y) = Trunc\ Normal((1, 2), (1, 1), -0.3, [-2, 2], [-4, 5])$.

Example	Var	$\lvert M\rvert$	Orthogonalization Runtime (in seconds)	K-series Runtime (in seconds)
Truncated exponential	X	2	0.00246	0.04379
The Irwin-Hall Distribution	X	6	0.05029	0.06922
Probabilistic loop with non-polynomial assignment	r	4	0.03568	0.04936
Truncated Bivariate Normal	(X, Y)	8	0.07077	0.03613

Table 3.1: Runtimes of orthogonalization procedure and K-series estimation for the illustrative benchmarks. $\lvert M\rvert$ denotes number of used moments and Var the variable(s) whose density is estimated.

3.2 Symbolic K-series representation along iterations

In this section, we demonstrate the unique ability of our method to express the distribution of one or multiple state variables as a function of the iteration number in closed form.

We introduce the semantics of *prob-solvable loops*, introduced by [BKS19], as we are considering infinite probabilistic loops and the properties of state variables at each iteration. For the class of *prob-solvable loops* (see Definition 1), moments of all orders of program variables can be symbolically computed. Given a prob-solvable loop and a program variable x, [BKS19] calculate a closed-form solution for $\mathbb{E}(x_t^k)$ for any arbitrary $k \in \mathbb{N}$, with t representing the t-th loop iteration. Prob-solvable loops were initially restricted to polynomial variable updates. [KMS+22b] relaxed the restriction to allow square-integrable function updates (see Section 2.1.1).

The K-series estimator can be expressed as a quantitative invariant in the sense that its formula is a function of loop iteration. In the univariate case, the K-series estimator (3.4) of the unknown pdf of the random variable X is $\hat{f}(x) = \phi(x) \sum_{i=0}^{n} \left(\sum_{j=0}^{i} a_{ij} m_j \right) h_i(x)$, where $m_j = \mathbb{E}(X^j)$. The estimator is a function of the moments of X, which in turn, vary along iterations in a probabilistic loop. That is, the K-series estimator can be equivalently expressed as

$$\hat{f}_t(x) = \phi(x) \sum_{i=0}^{n} \left(\sum_{j=0}^{i} a_{ij} m_j(t) \right) h_i(x), \tag{3.23}$$

where $m_j(t) = \mathbb{E}(X^j(t))$ is the moment of the random variable X at iteration t. Formula (3.23) is the symbolic representation of the K-series pdf estimator as a function of iteration number. Similarly, the multivariate K-series estimator (3.21) can be written as

$$\hat{f}(\mathbf{x}) = \widetilde{\phi}(\mathbf{x}) \sum_{\substack{i_j \in \{0,\dots,d_j\}, \\ j=1,\dots,k}} \left(\sum_{\substack{l_j \in \{0,\dots,i_j\}, \\ j=1,\dots,k}} m_{l_1,\dots,l_k}(t) \prod_{j=1}^{k} a_{i_j l_j}^{j} \right) \tilde{h}_{i_1,\dots,i_k}(\mathbf{z}) \tag{3.24}$$

where $m_{l_1,\dots,l_k}(t) = \mathbb{E}\left(X_1^{l_1}(t) \cdot \ldots \cdot X_k^{l_k}(t) \right)$ at iteration t, since the moments of the random vector depend on the iteration in a probabilistic loop.

We illustrate (3.23) by considering the probabilistic loop in Fig. 3.3(A): the target random variable r is modeled as the minimum of random variables x and y. Variable y is uniformly distributed on $(0, 20)$, while x follows a mean-reverting process and is affected by the stochastic shock $\theta \sim$Uniform$(-8, 8)$ at each iteration. We can now use the approach from [BKS19] to estimate moments for arbitrary iterations and use them to receive the symbolic expression for the pdf of r for the corresponding iteration. Since $\min(\cdot, \cdot)$ is a non-polynomial function, we apply the approach in [KMS+22b] to represent $\min(\cdot, \cdot)$ as an expansion in orthogonal polynomials. The transformation is given in the bottom panel of Fig. 3.3.

Once this is computed, the program in Fig. 3.3 (B) can be handled using the algorithm in [BKS19]. The equations estimating the first four moments for each iteration are in the left panel of Fig. 3.4. We choose the uniform distribution on $(0, 20)$ as the reference pdf. We compute the shifted and scaled Legendre polynomials and substitute the moment equations as functions of iteration t. Similarly, we can derive the symbolic expression of the pdf estimate for any arbitrary iteration t. The right

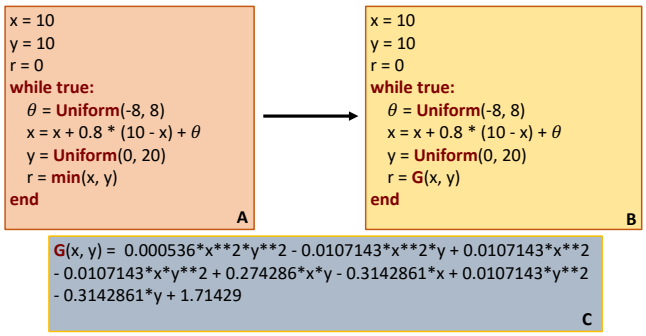


Figure 3.3: (A) Probabilistic loop with non-polynomial assignment, (B) Transformation of the program A using Polynomial Chaos Expansion [KMS+22b], by replacing the function $\min(\cdot,\cdot)$ with the polynomial $G(x, y)$.

panel of Fig. 3.4 plots the pdf estimate of the random variable r at iteration $t = 30$ given by

$$\hat{f}_{30}(r) = 5.165866e-7 * r^4 + 2.561246e-5 * r^3 - 0.001472 * r^2 + 0.012320 * r + 0.055246.$$

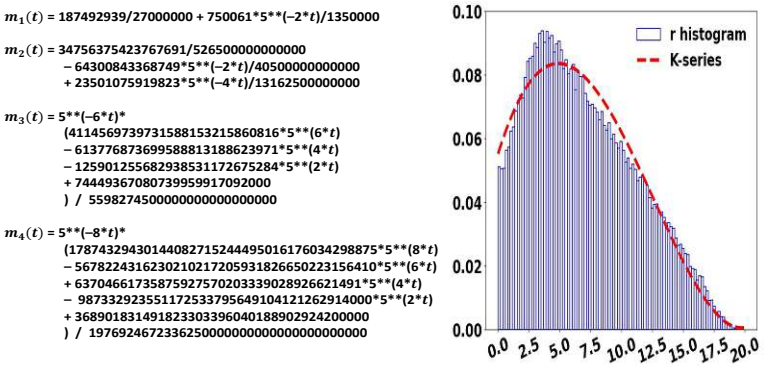


Figure 3.4: Left panel: First four moments expressed symbolically in the number of iterations. Right panel: Comparison between the histogram of the sampling pdf and the symbolic K-series estimation at $t = 30$.

We report the runtimes of the illustrative examples in Table 3.1.

3.3 Experiments

We carried out K-series estimation of the distributions of the random variables generated in the execution of several probabilistic loops. The implementation code is available upon request. The first application is the *Differential-Drive Mobile Robot* in Fig. 1.5, where we observe a practically perfect approximation of both marginal and joint pdfs of the location of the robot. All experiments were conducted on a machine equipped with 16 GB of RAM and an Apple M1 Pro processor. The

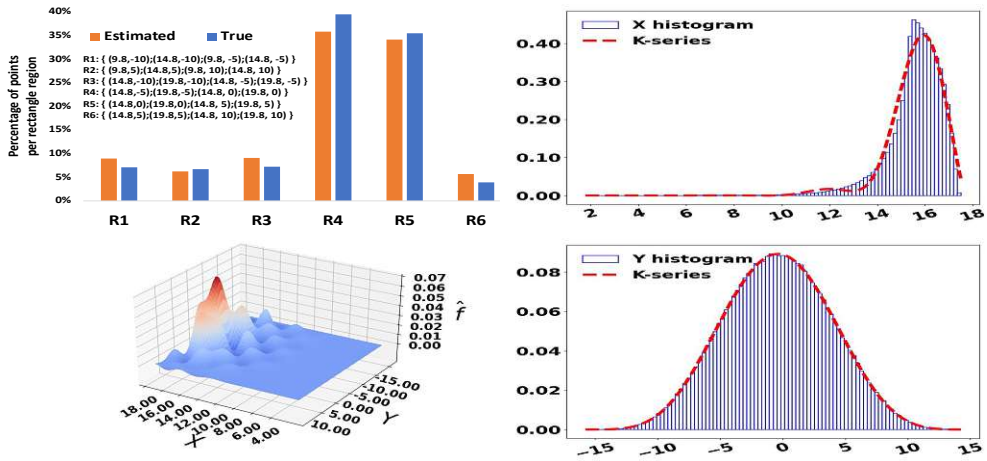


Figure 3.5: Turning Vehicle Model (Fig. 1.3, Panel A1 of Fig. 3.7): K-series estimates of the marginal pdfs of X and Y (right upper and lower panels), the joint (lower left panel) and comparison bar plot (upper left panel) at iteration $t = 20$.

runtimes of all experiments in this section are displayed in Table 3.3. The Python code for the experiments in this chapter can be found at GitHub. We distinguish between the time required for the Gram-Schmidt process and the time for the estimator construction. Our approach is highly time-efficient. Additionally, users can leverage precomputed standard sets of orthogonal polynomials to avoid recomputing them using the Gram-Schmidt process. Formal statistical tests for the goodness-of-fit of our estimates and the true (sampling) pdfs are carried out in Appendix 3.3.1 and the results, which overwhelmingly support our estimation procedure, are reported in Table 3.2.

The program in Panel A1 of Fig. 3.7 encodes the *turning vehicle model* in [SCG⁺20; KMS⁺22b]. We use the truncated normal on $(1, 18) \times (-15, 15)$ with mean the sample mean and variance 4 for X, and the sample variance of the Y distribution as reference pdfs. While the support of X is not important, the accuracy of the estimation depends on the variance for X. When the variance is very small, the estimation becomes numerically unstable. This effect on the estimation is reflected in the K-series detecting, possibly artificially, two modes in Fig. 3.5.

The program in Panel A of Fig. 3.7 is the same as the *turning vehicle model* [SCG⁺20; KMS⁺22b] in Panel A1 of Fig. 3.7, with the difference that the variance of the basic random variables ψ and w_2 is about 3 times larger. The effect of this on the joint and marginal distributions of X, Y can be seen in Fig. 3.8. In this case, the reference is truncated normal on $(-18, 18) \times (-20, 20)$ with mean the sample mean and variance the sample variance of the marginal distributions of X and Y, respectively. While the support of X is not important, the accuracy of the estimation depends on the variance for X. The K-series estimator is a sum of weighted orthonormal polynomials whose Gram-Schmidt orthogonalization with respect to the reference distribution involves the variance of the generated variables in the denominator. Thus, when the variance is very small, the fraction explodes and the estimation becomes numerically unstable. This can be managed by increasing the variance of the reference, as done in Fig. 3.8.

In Fig. 3.7, Panel B encodes the *Taylor rule* , a model for monetary policy [Tay93; KMS⁺22b], D the *rimless wheel walker* [ST12], and E the *Vasicek* [Vas77; KMS⁺22a] model. The *Taylor rule* (B), *rimless wheel* (D) and *Vasicek* model (E) generate a single random variable at each iteration.

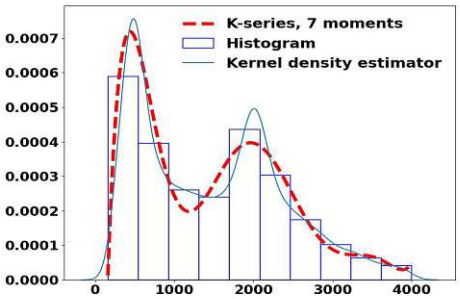


Figure 3.6: K-series estimate of pdf of household electricity

We plot the histograms from sampling the probabilistic loop programs (blue) and the overlaid pdf K-series estimates of these models in Fig. 3.10. The *2D robotic arm* model [BGP+16] in panel C of Fig. 3.7 generates a bivariate random variable. We plot the marginal K-series pdf estimates in the right panels, the joint pdf approximation in the bottom left panel, and the comparison of the true (blue bars) with our estimate (red bars) over a 2D parallelogram grid in the top left panel of Fig. 3.9. The moments of the true distribution were computed with the method in [KMS+22b] for the Taylor rule, and in [BKS19] for the rimless wheel, Vasicek and 2D robotic arm models. We used the following reference pdfs: truncated normal on $(-30, 30)$ for the Taylor rule, truncated normal on $(0, 10)$ for rimless wheel, normal distribution for the Vasicek model, and truncated normal on $(260, 280) \times (525, 540)$ for the 2D robotic arm model. For all univariate and bivariate models, our K-series estimator exhibits excellent estimation accuracy.

In a real data application, we use K-series to estimate the density of "household electricity use with a ten-minute resolution for a detached house over one year" [MMR17]. The data were analyzed by [MMR17], who estimated the unknown pdf by using sample-based estimates for the true unknown moments of the target distribution. Histograms of the data indicate that the pdf is bimodal. In real data examples, the true moments are unknown, so we also use the sample-based moment estimates to compute our K-series estimate, which is drawn in Fig. 3.6. We juxtapose our sample-moment-informed estimate with a nonparametric kernel density estimate, a standard data-driven approach for density estimation, for visual comparison. The K-series estimate fits the data better, especially at both endpoints of the support, than [MMR17]'s MM estimate, which can be viewed at the PLOS One site.

Regarding the time efficiency of K-series vis-à-vis other methods, Gram-Charlier is a special case of K-series for a normal reference distribution (Proposition 2), and the Method of Moments [MMR17] is a special case of K-series for a uniform reference distribution (Theorem 2). As such, the computational time for their implementation is the same as for K-series. Kernel density estimation (KDE) is not based on moments but requires a large number of samples from the population at hand. That is, the probabilistic program would have to be run many times to compute the kernel density estimator over its realized range of values to achieve comparative accuracy if even possible. Theoretically, K-series cannot be beaten in accuracy when true moments are available. As an example, in Fig. 3.11 we plot the true pdf of a mixture of an equal-weighted mixture of four beta distributions with parameters (1.3, 5), (5, 1.3), (6, 7) and (7, 6), respectively, in green. The K-series estimator is the red dashed curve and the KDE estimate, based on the Gaussian kernel, is

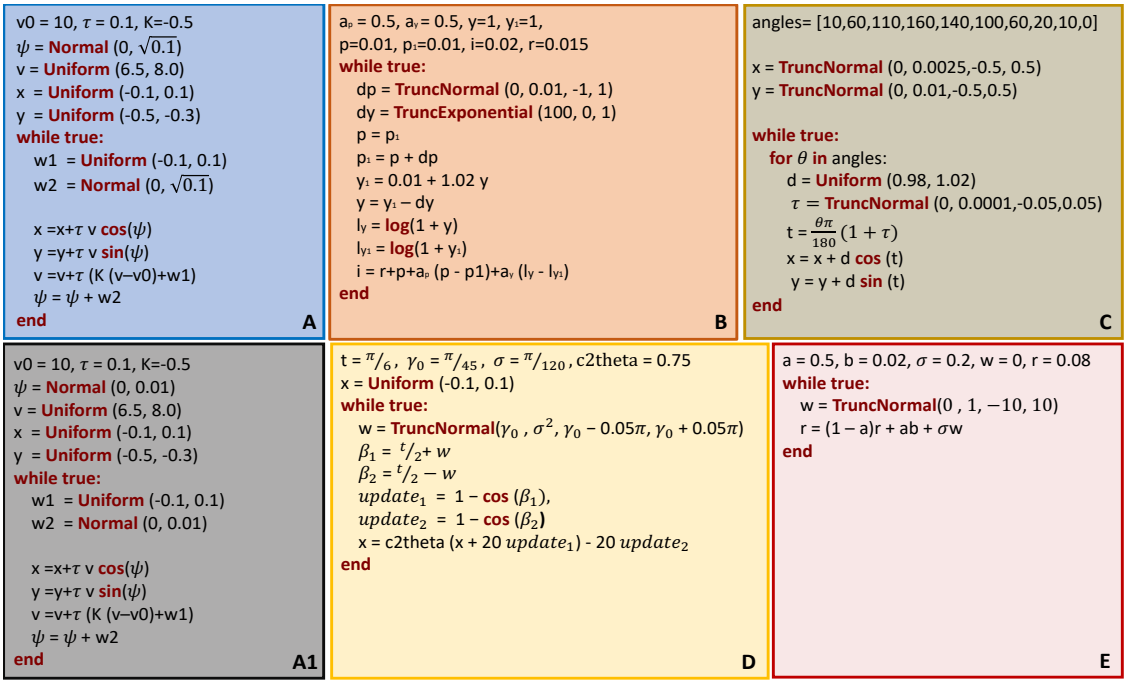


Figure 3.7: Probabilistic loops: (A) Turning vehicle model [SCG⁺20; KMS⁺22b], (A1) Small variance Turning vehicle model [SCG⁺20; KMS⁺22b], (B) Taylor rule [Tay93; KMS⁺22b], (C) 2D Robotic Arm [BGP⁺16], (D) Rimless Wheel Walker [ST12], (E) Vasicek model (truncated version) [Vas77; KMS⁺22a].

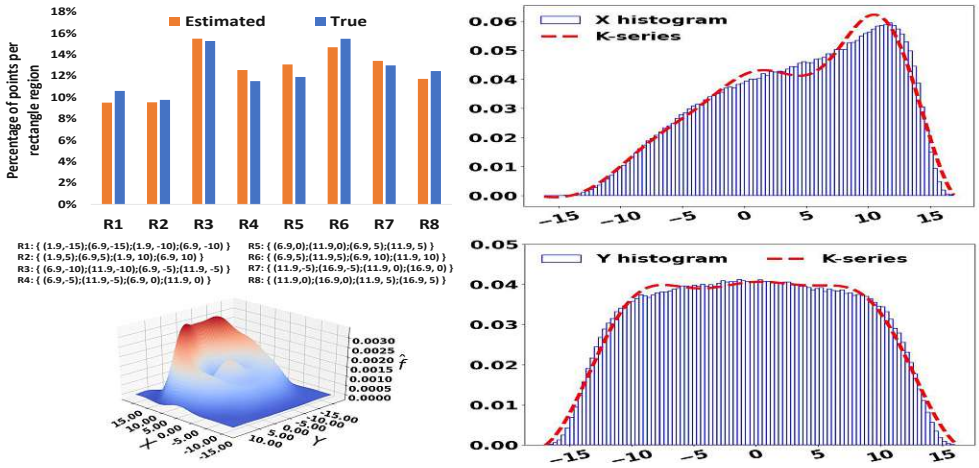


Figure 3.8: Turning vehicle model Fig. 3.7 (A): K-series estimates of the marginal pdfs of X and Y (right upper and lower panels), the joint (lower left panel) and comparison bar plot (upper left panel) at iteration $t = 20$.

the blue curve. We sampled 10000 observations from the true pdf and plotted their histogram in gray. The time to produce the KDE estimate is $\{0.00644s + 0.0138s\}$ (sample and compute pdf, resp.). The time to compute the K-series estimate is longer, $\{0.73s + 0.662s + 4.58s\}$ (compute

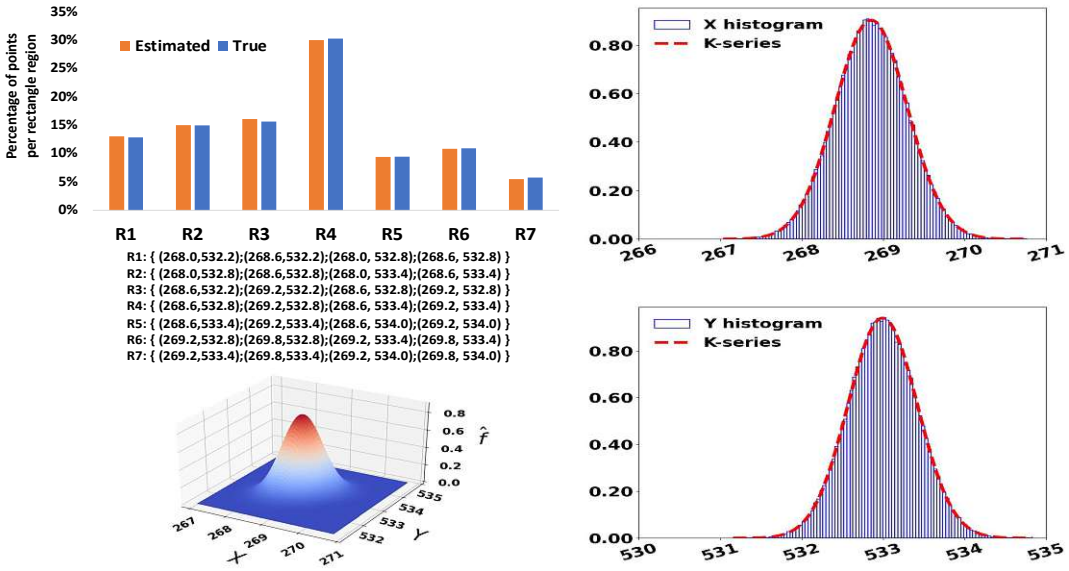


Figure 3.9: Robotic arm model Fig. 3.7 (C): K-series estimates of the marginal pdfs of X and Y (right upper and lower panels), the joint (lower left panel) and comparison bar plot (upper left panel) at iteration $t = 100$.

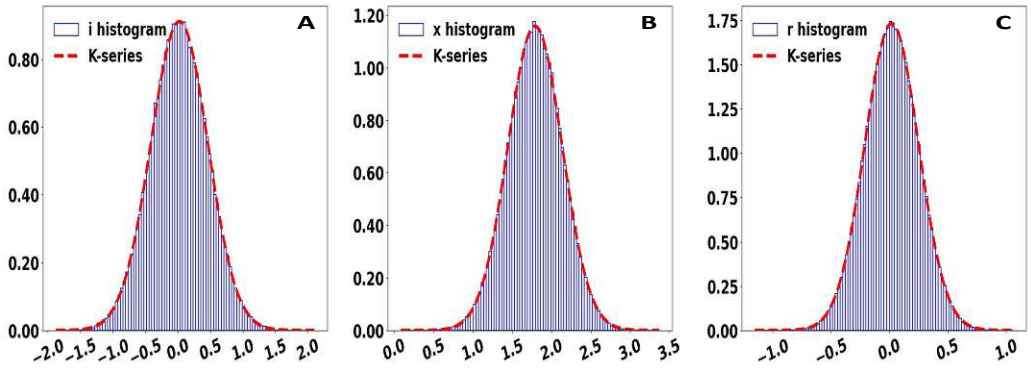


Figure 3.10: K-series estimates of pdf for variable **A)** i at iterations $t = 20$ in Fig. 3.7 (B): Taylor rule model, **B)** x at iteration $t = 2000$ in Fig. 3.7 (D): rimless wheel model, **C)** r at iteration $t = 100$ in Fig. 3.7 (E): Vasicek model.

moments, construct a system of orthogonal polynomials and compute K-series, respectively). But Fig. 3.11 reveals that K-series tracks the true pdf much more accurately than the KDE, which is also subject to boundary effects, a well-known problem in nonparametric estimation. In Fig. 3.12, we visually compare the cdf estimates of the two approaches using 50000 samples. Again, the K-series cdf is closer to the true cdf, especially at the endpoints. Also, the Kolmogorov-Smirnov distance between the K-series and the true cdf is 0.0012478, much smaller than 0.0226002, the value of the Kolmogorov-Smirnov distance between the true cdf and the cdf of the KDE estimate. As an aside comment, we note that we used a grid of 1000 points to compute all the pdfs for all other examples

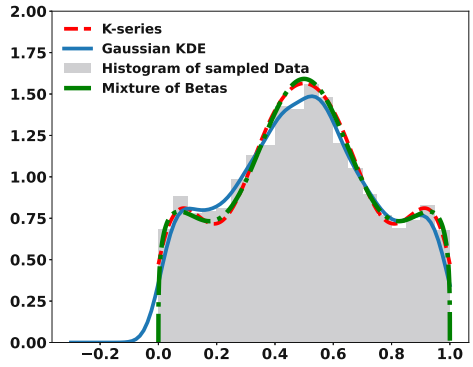


Figure 3.11: Comparison of K-series and KDE with Gaussian kernel performance in estimating a mixture of four Beta pdfs. We used 9 moments for K-series.

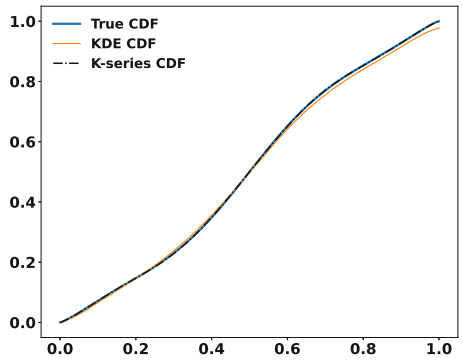


Figure 3.12: Comparison of K-series and KDE with Gaussian kernel performance in estimating a mixture of four Beta pdfs. We used a grid of 50.000 points.

in the chapter. Here, we had to use a much larger number of points to receive a sample of reliable size for KDE.

Panel A in Fig. 3.13 describes the *1D Random Walk*, and panel B the *2D Random Walk* [KUH19]. For the former, we used a truncated normal distribution on $(-98, 102)$ as reference. For the 2D Random Walk, we used two independent truncated normal distributions on $(-100, 100) \times (-100, 100)$ with true means and variances of corresponding marginal pdfs obtained with the algorithm in [BKS19]. The K-series estimator exhibits excellent performance for both 1D and 2D random walks, as can be seen in Fig. 3.13.

We explore the robustness of our method to violations of the assumption of continuity of random variables in Fig. 3.14 , where we estimate the distributions of random variables generated in Prob-solvable loops with discrete random components. Panel A in Fig. 3.14 encodes the *Stuttering P model* in [BKS19] and panel B the piece-wise deterministic process, or *PDP model*, modeling gene circuits that can be used to estimate the bivariate distribution of protein x and the mRNA levels y in a gene [IHR18].

For the *Stuttering P* model, we used a truncated normal distribution on $(0, 50)$ with true mean and variance as the reference pdf. For the *PDP* model, we used the truncated normal on $(100, 1800)$ for X and uniform on $(8, 80)$ for Y as reference pdfs to estimate marginal pdfs of X and Y and joint pdf (X, Y). The parameters of the truncated normal distribution are the exact mean and variance of the marginal pdf of the corresponding variable computed using [BKS19].

3.3.1 Kolmogorov-Smirnov and Energy Tests for Equality of Distributions

The Kolmogorov-Smirnov (K-S) test [HWC13] compares two cumulative distribution functions (cdfs). We compute the cdf $\hat{F}_{\mathrm{KS}}$ of the estimated pdf $\hat{f}_{\mathrm{KS}}$. We also compute the (empirical) cdf F_{Sample} of the data resulting from sampling the probabilistic program variables. The 2-sample

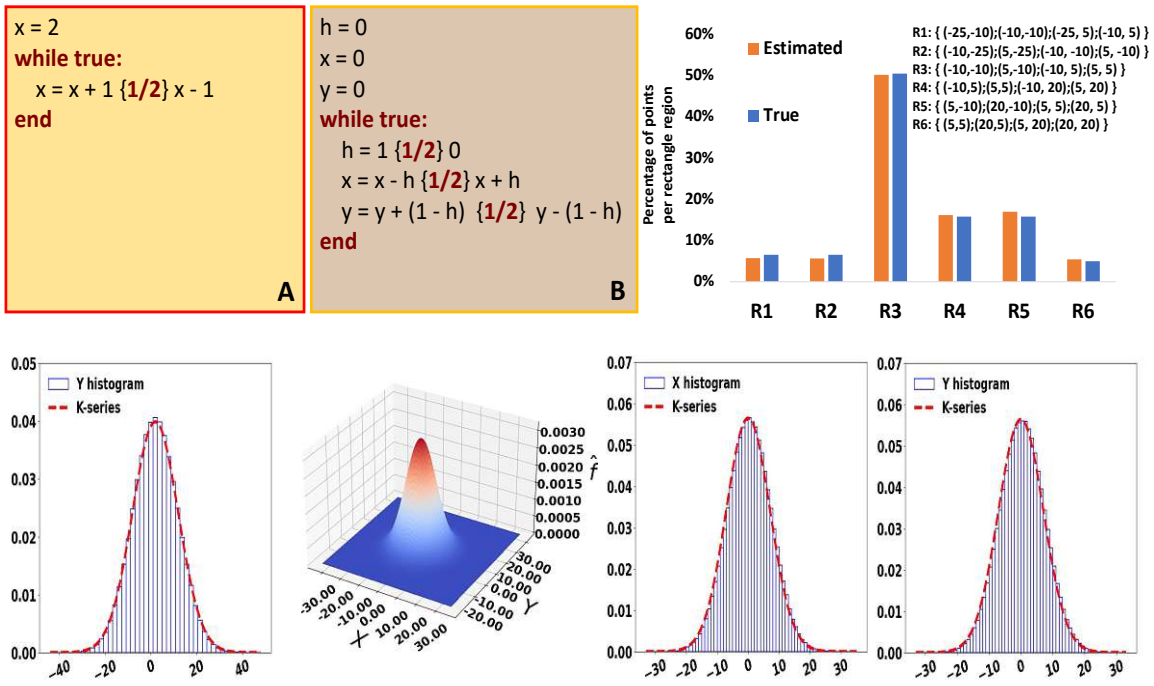


Figure 3.13: K-series estimates of the pdf of X in 1D Random Walk (A) [KUH19] at iteration $t = 100$, marginal pdfs for variables X, Y and joint distribution of (X, Y) in 2D Random Walk (B) [KUH19] at the iterations $t = 100$.

Kolmogorov-Smirnov (K-S) test statistic for testing equality of the population (true) cdfs is

$$D_{\mathrm{KS}} = \max_x \left(|F_{\mathrm{KS}}(x) - F_{Sample}(x)| \right), \tag{3.25}$$

where N_1 and N_2 are the sample sizes from the K-series and empirical cdf, respectively. We reject the equality of the two distributions if

$$D_{\mathrm{KS}} > c(\alpha)\sqrt{\frac{N_1 + N_2}{N_1 \cdot N_2}} = \sqrt{-\frac{1}{2}\ln\frac{\alpha}{2}}\sqrt{\frac{N_1 + N_2}{N_1 \cdot N_2}}$$

at significance level α.

The two-sample E-statistic for testing for equality of multivariate distributions proposed by [SR04] is the *energy* distance $e(S_1, S_2)$, which is defined by

$$e(S_1, S_2) = N_1 N_2 \left(2D_{12} - D_{11} - D_{22}\right) / (N_1 + N_2),$$

for two samples S_1, S_2 of respective sizes N_1, N_2, where

$$D_{ij} = \sum_{p=1}^{N_i} \sum_{q=1}^{N_j} ||\mathbf{X}_{ip} - \mathbf{X}_{jq}|| / (N_i N_j), \; i, j = 1, 2,$$

$||\cdot||$ denotes the Euclidean norm, and $\mathbf{X}_{1p}$ denotes the p-th and $\mathbf{X}_{2q}$ the q-th (vector-valued) observations in the first and second sample, respectively. The test is implemented by nonparametric bootstrap, an approximate permutation test in the R-package `energy` [RS22].

We used the Kolmorov-Smirnov test to compare univariate distributions and the *energy* test for multivariate distributions [SR04]. We draw 1000 observations from the sampling ("true") and estimated distributions. The critical values are 0.0607 and 0.0479 for significance levels 0.05 and 0.2, respectively. Except for very few instances, when a small number of moments is used in the K-series estimation, our estimate is statistically the same as the true distribution. We also test the agreement of the K-series with the GC estimates. When the true distribution is similar to normal, K-series is statistically indistinguishable from Gram-Charlier. But when the true distribution is not close to normal, K-series provides a far more accurate estimate than Gram-Charlier.

Problem	Var	$\|M\|$	KS Distance	KS Distance (GC)	Energy test (p-value)
Differential-Drive Robot					
	X	6	0.00069 ✔ !	0.00072 ✔ !	
	Y	6	0.00059 ✔ !	0.00059 ✔ !	
	(X,Y)	48			0.4700
PDP					
	X	2	0.00664 ✔ !	0.00680 ✔ !	
	Y	6	0.00033 ✔ !	0.05190 ✔	
	(X,Y)	8			0.4250
Turning vehicle					
	X	8	0.00807 ✔ !	0.02109 ✔ !	
	Y	8	0.00494 ✔ !	0.01030 ✔ !	
	(X,Y)	80			0.4150
Turning vehicle (small variance)					
	X	8	0.02614 ✔ !	0.11054 ✗	
	Y	8	0.00070 ✔ !	0.00169 ✔ !	
	(X,Y)	80			0.5000
Taylor rule model					
	i	6	0.00037 ✔ !	0.00037 ✔ !	
2D Robotic Arm					
	X	2	0.00037 ✔ !	0.00037 ✔ !	
	Y	2	0.00048 ✔ !	0.00048 ✔ !	
	(X,Y)	8			0.9650
Rimless Wheel Walker					
	X	2	0.00180 ✔ !	0.00180 ✔ !	
Vasicek model					
	r	2	0.00074 ✔ !	0.00074 ✔ !	
1D Random Walk					
	X	2	0.03834 ✔ !	0.03834 ✔ !	
2D Random Walk					
	X	2	0.02743 ✔ !	0.02743 ✔ !	
	Y	2	0.02714 ✔ !	0.02714 ✔ !	
	(X,Y)	8			0.4902
Stuttering P					
	S	2	0.00351 ✔ !	0.00354 ✔ !	

✔ Null hypothesis is not rejected at significance level 0.05.
✔ ! Null hypothesis is not rejected at significance level 0.2.
✗ Null hypothesis is rejected at significance level 0.05.

Table 3.2: Kolmogorov-Smirnov distances for univariate distributions and testing for equality of multivariate distributions.

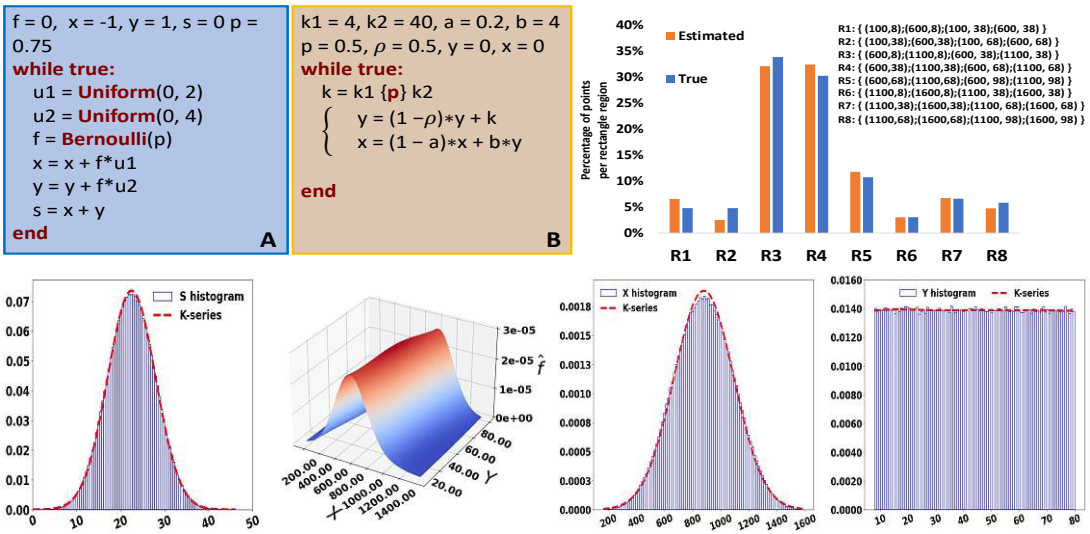


Figure 3.14: K-series estimates of the pdf of variable S in Stuttering P model [BKS19] (A) at iteration $t = 10$, marginal pdfs for variables X, Y and joint distribution for variables (X, Y) in PDP model [IHR18] (B) at the iterations $t = 100$.

3.4 Effect of Reference Distribution

We study the effect of the choice of the reference distribution in K-series on estimation accuracy. We consider reference distributions with the same support as the target unknown pdf f, with bounded support that contains the support of f and with unbounded support in absence of any knowledge about the possible values of the target distribution.

Table 3.4 lists the combinations of target and reference distributions we consider in our experiments. We plot the true target pdfs (red) and the K-series estimates for different numbers of moments using reference pdfs with the same support as the target in Figure 3.15. Our method does not suffer from the numerical instability associated with closeness to zero. In most cases, the uniform reference pdf works better on exact support.

In Figure 3.16, we plot the true four pdfs in Table 3.4 and their K-series estimates using different number of moments and the uniform reference supported on an interval that contains the support of the target pdf. Specifically, the reference pdf is supported on the interval that extends by 2 units the true support in either side. The estimation improves significantly as the number of moments increases. The left panels of Figure 3.17 plot the true pdfs and their K-series estimates using different numbers of moments and a truncated normal reference supported on the interval that extends by 2 units the true support in both ends. The right panels of Figure 3.17 plot the true pdfs and their K-series estimates using different numbers of moments and a normal reference pdf supported on the entire real line.

Visual inspection of these plots indicates that the estimation is better if the support of all reference pdfs is close to the support of the target pdf. The uniform reference distribution results in accurate estimates provided its support is close to the support of the true pdf. On the other hand, both truncated and regular normal reference pdfs lead to accurate K-series estimates the closer the target pdf is to a normal. Moreover, the truncated normal distribution tends to work better on a support

Model	Var	$\lvert M\rvert$	Orthogonalization Runtime (in seconds)	K-series Runtime (in seconds)
Differential-Drive Robot				
	X	6	0.67484	0.15971
	Y	6	0.57628	0.15921
	(X,Y)	48	1.23404	0.18318
PDP				
	X	2	0.03708	0.13438
	Y	6	0.24701	0.07646
	(X,Y)	8	0.03880	0.27987
Turning vehicle				
	X	8	0.46561	0.17895
	Y	8	0.48054	0.17901
	(X,Y)	80	0.84230	0.99251
Turning vehicle (small variance)				
	X	8	0.68676	0.17829
	Y	8	0.62172	0.17739
	(X,Y)	80	1.19591	0.98794
Taylor rule model				
	i	6	2.66375	0.16011
2D Robotic Arm				
	X	2	0.15185	0.13439
	Y	2	0.13663	0.13528
	(X,Y)	8	0.28913	0.36801
Rimless Wheel Walker				
	X	2	0.10627	0.10915
Vasicek model				
	r	2	0.16654	0.09937
1D Random Walk				
	X	2	0.09753	0.15714
2D Random Walk				
	X	2	0.13076	0.15916
	Y	2	0.13033	0.15678
	(X,Y)	8	0.25956	0.40327
Stuttering P				
	S	2	0.06936	0.15530

Table 3.3: Runtimes of orthogonalization procedure and K-series estimation for the benchmarks in Section 3.3. $\lvert M\rvert$ denotes number of used moments and Var the variable(s) whose density is estimated.

wider than the true in comparison with the uniform.

Formal assessment of the estimation accuracy is carried out with Kolmogorov-Smirnov tests. Tables 3.5, 3.6 and 3.7 report the values of the Kolmogorov-Smirnov test statistic comparing the K-series estimates with the true pdfs and whether the null of their equality is rejected for different

Target pdf f	Reference pdf ϕ
TruncExp($\lambda = 2/3, [0, 4]$)	
	$Uniform(0, 4)$
	$TruncNormal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [0, 4])$
	$Uniform(-2, 6)$
	$TruncNormal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [-2, 6])$
	$Normal(\mathbb{E}(f), \mathbb{V}\text{ar}(f))$
TruncGamma($\alpha = 2, \beta = 0.5, [0, 5]$)	
	$\phi \sim Uniform(0, 5)$
	$TruncNormal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [0, 5])$
	$Uniform(-2, 7)$
	$Trunc\ Normal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [-2, 7])$
	$Normal(\mathbb{E}(f), \mathbb{V}\text{ar}(f))$
Continuous Bernoulli($\pi = 0.3$)	
	$\phi \sim Uniform(0, 1)$
	$TruncNormal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [0, 1])$
	$Uniform(-2, 3)$
	$TruncNormal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [-2, 3])$
	$Normal(\mathbb{E}(f), \mathbb{V}\text{ar}(f))$
TruncNormal($1.5, 5.76, [-6, 6]$)	
	$\phi \sim Uniform(-6, 6)$
	$TruncNormal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [-6, 6])$
	$Uniform(-8, 8)$
	$TruncNormal(\mathbb{E}(f), \mathbb{V}\text{ar}(f), [-8, 8])$
	$Normal(\mathbb{E}(f), \mathbb{V}\text{ar}(f))$

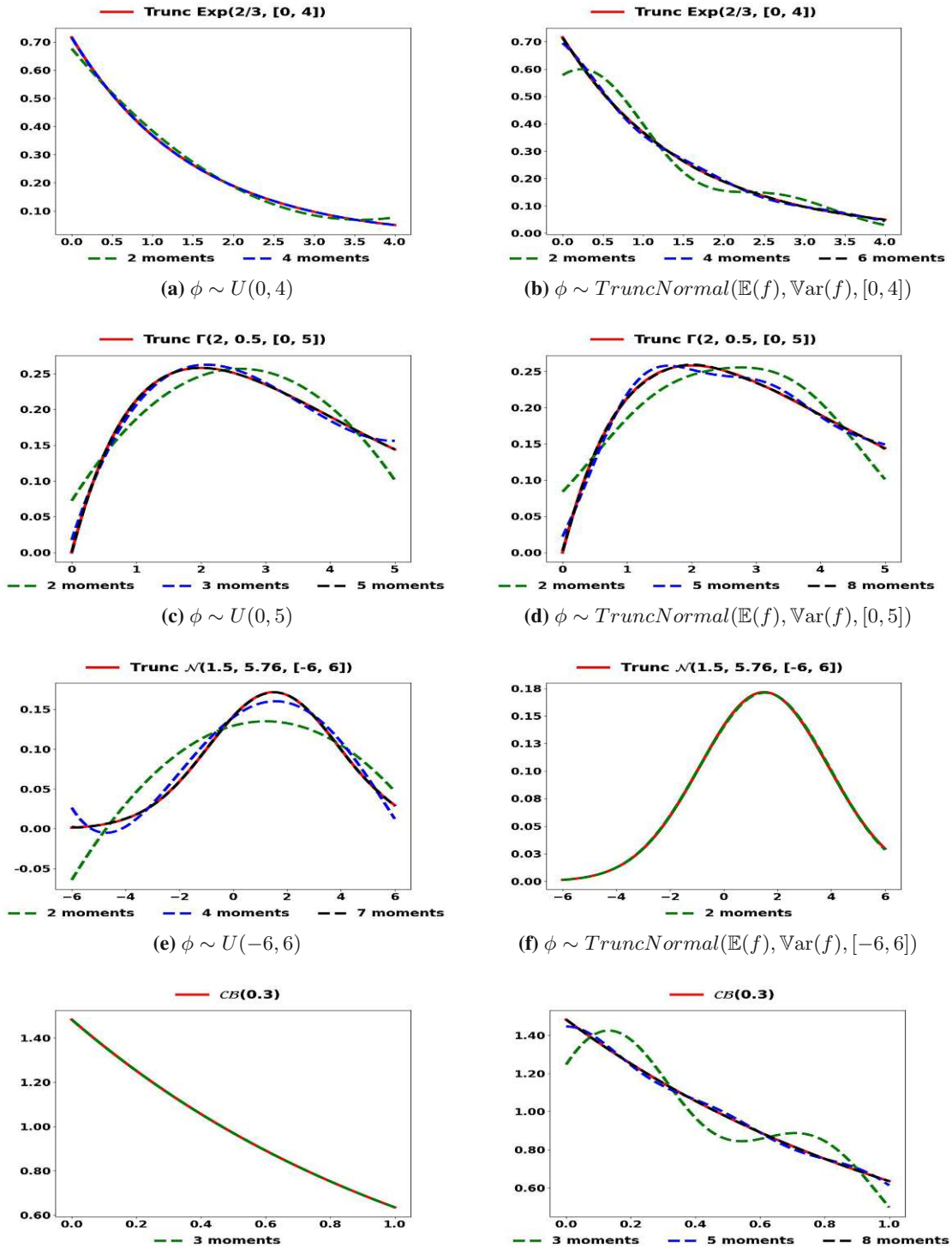
Table 3.4: Target and reference distributions.

numbers of moments and reference distributions. The sample size for both the estimated and true distribution is 1000. The critical values are 0.0607 and 0.0479 for significance levels 0.05 and 0.2, respectively.

Target pdf f	$\lvert M\rvert$	Uniform (Same support)	TruncNormal (Same support)
TruncGamma$(\alpha = 2, \beta = 0.5, [0, 5])$			
	2	0.0172 ✔ !	0.0188 ✔ !
	3	0.0031 ✔ !	0.0093 ✔ !
	5	$< 1e-4$ ✔ !	0.0033 ✔ !
	8	$< 1e-4$ ✔ !	0.0002 ✔ !
TruncNormal$(1.5, 5.76, [-6, 6])$			
	2	0.0617 ✗	0.0011 ✔ !
	4	0.0122 ✔ !	$< 1e-4$ ✔ !
	7	0.0002 ✔ !	$< 1e-4$ ✔ !
Continuous Bernoulli$(\pi = 0.3)$			
	3	$< 1e-4$ ✔ !	0.0124 ✔ !
	5	$< 1e-4$ ✔ !	0.0012 ✔ !
	8	$< 1e-4$ ✔ !	$< 1e-4$ ✔ !
TruncExp$(\lambda = 2/3, [0, 4])$			
	2	0.0082 ✔ !	0.0212 ✔ !
	4	0.0001 ✔ !	0.0025 ✔ !
	6	$< 1e-4$ ✔ !	0.0003 ✔ !

✔ Null hypothesis is not rejected at significance level 0.05.
✔ ! Null hypothesis is not rejected at significance level 0.2.
✗ Null hypothesis is rejected at significance level 0.05.

Table 3.5: Kolmogorov-Smirnov distances and significance test results for reference distributions on the same support as the true pdf.



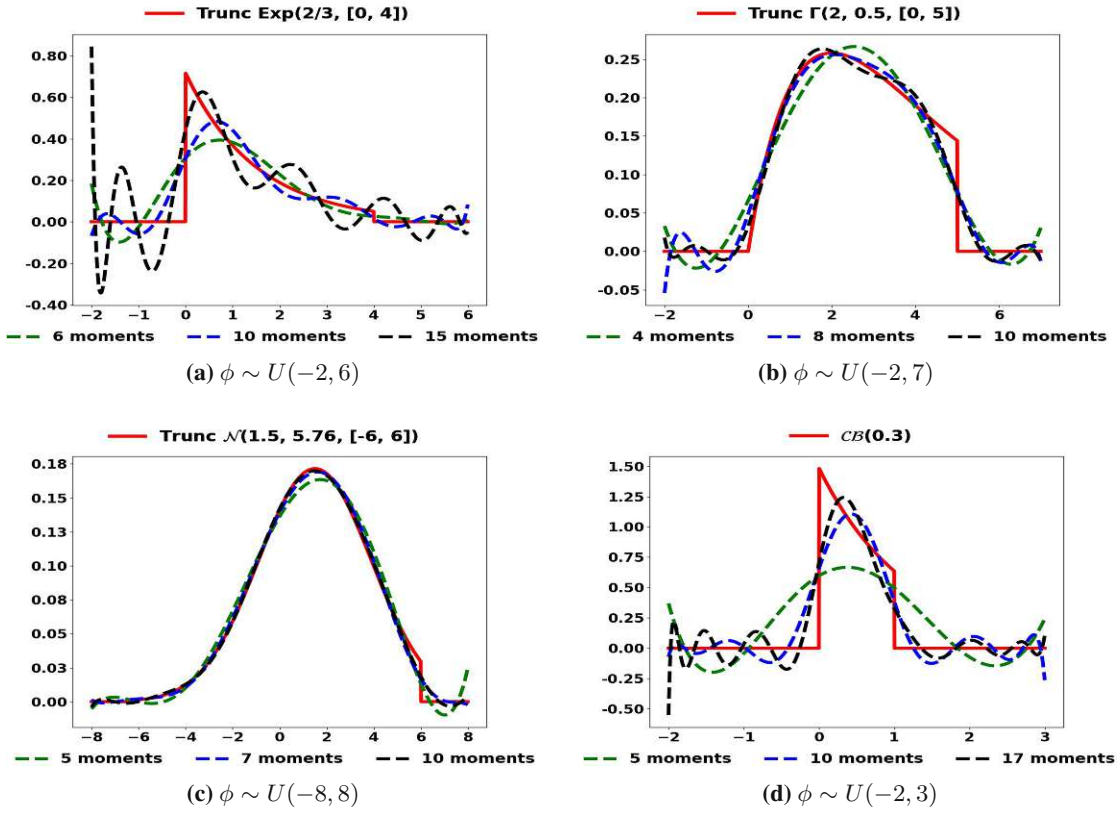
(a) $\phi \sim U(0, 4)$ **(b)** $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [0, 4])$

(c) $\phi \sim U(0, 5)$ **(d)** $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [0, 5])$

(e) $\phi \sim U(-6, 6)$ **(f)** $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [-6, 6])$

(g) $\phi \sim U(0, 1)$ **(h)** $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [0, 1])$

Figure 3.15: K-series estimates of the truncated exponential pdf, the truncated gamma pdf, the truncated normal pdf and the continuous Bernoulli with uniform reference (Method of Moments [MMR17]), left panels) and truncated normal (right panels) on exact support.



(a) $\phi \sim U(-2, 6)$ **(b)** $\phi \sim U(-2, 7)$

(c) $\phi \sim U(-8, 8)$ **(d)** $\phi \sim U(-2, 3)$

Figure 3.16: Approximations of the truncated exponential pdf, the truncated gamma pdf, the truncated normal pdf and the continuous Bernoulli using K-series with uniform reference on the extended support.

Target pdf f	$\lvert M\rvert$	Uniform (Extended support)
TruncGamma$(\alpha = 2, \beta = 0.5, [0, 5])$		
	4	0.0213 ✔ !
	8	0.0186 ✔ !
	10	0.0152 ✔ !
TruncNormal$(1.5, 5.76, [-6, 6])$		
	5	0.0099 ✔ !
	7	0.0061 ✔ !
	10	0.0048 ✔ !
Continuous Bernoulli$(\pi = 0.3)$		
	5	0.2285 ✗
	10	0.0939 ✗
	17	0.0579 ✔
TruncExp$(\lambda = 2/3, [0, 4])$		
	6	0.1099 ✗
	10	0.0713 ✗
	15	0.0546 ✔

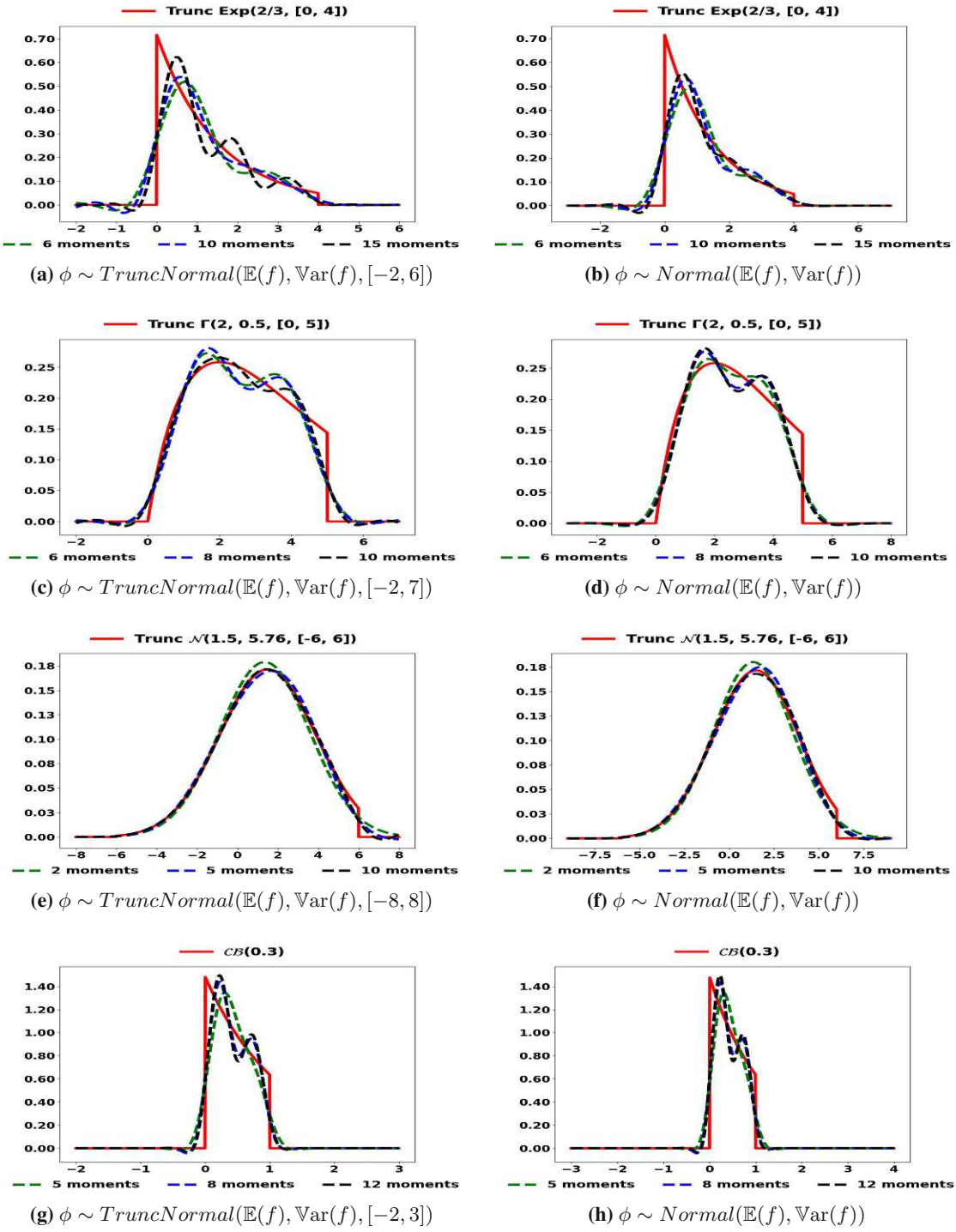
✔ Null hypothesis is not rejected at significance level 0.05.
✔ ! Null hypothesis is not rejected at significance level 0.2.
✗ Null hypothesis is rejected at significance level 0.05.

Table 3.6: Kolmogorov-Smirnov distances and significance test results for the uniform reference distribution on extended support.

Target pdf f	$\lvert M\rvert$	Trunc Normal (Extended support)	Normal (real line)
TruncGamma$(\alpha = 2, \beta = 0.5, [0, 5])$			
	6	0.0172 ✔ !	0.0202 ✔ !
	8	0.0158 ✔ !	0.0169 ✔ !
	10	0.0132 ✔ !	0.0033 ✔ !
TruncNormal$(1.5, 5.76, [-6, 6])$			
	2	0.0171 ✔ !	0.0182 ✔ !
	5	0.0071 ✔ !	0.0095 ✔ !
	10	0.0044 ✔ !	0.0066 ✔ !
Continuous Bernoulli$(\pi = 0.3)$			
	5	0.0516 ✔	0.0527 ✔
	8	0.0374 ✔ !	0.0387 ✔ !
	12	0.0340 ✔ !	0.0352 ✔ !
TruncExp$(\lambda = 2/3, [0, 4])$			
	6	0.0667 ✗	0.0757 ✗
	10	0.0558 ✔	0.0617 ✗
	15	0.0391 ✔ !	0.0524 ✔

✔ Null hypothesis is not rejected at significance level 0.05.
✔ ! Null hypothesis is not rejected at significance level 0.2.
✗ Null hypothesis is rejected at significance level 0.05.

Table 3.7: Kolmogorov-Smirnov distances and significance test results for truncated normal on extended support and normal reference distributions.



(a) $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [-2, 6])$ **(b)** $\phi \sim Normal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f))$

(c) $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [-2, 7])$ **(d)** $\phi \sim Normal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f))$

(e) $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [-8, 8])$ **(f)** $\phi \sim Normal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f))$

(g) $\phi \sim TruncNormal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f), [-2, 3])$ **(h)** $\phi \sim Normal(\mathbb{E}(f), \mathbb{V}\mathrm{ar}(f))$

Figure 3.17: Approximations of the truncated exponential pdf, the truncated gamma pdf, the truncated normal pdf and the continuous Bernoulli using K-series with truncated normal reference on the extended support (left) and normal reference on the whole real line (Gram-Charlier, right).

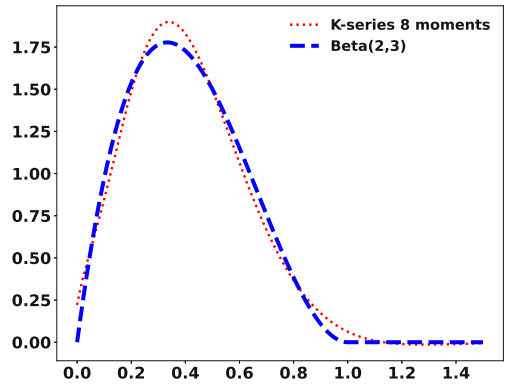


Figure 3.18: K-series estimates using first 8 moments of a Beta distribution with parameters (2, 3) and exponential reference with scale parameter 0.2.

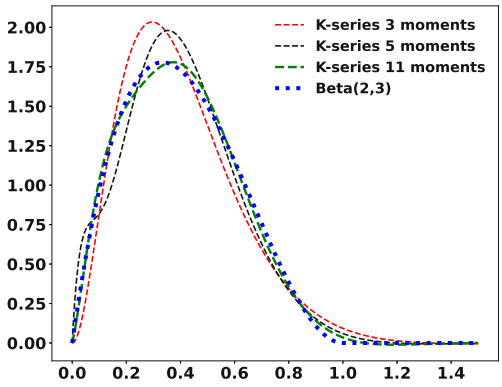


Figure 3.19: K-series estimates using the first 3, 5, and 11 moments, respectively, of a Beta distribution with parameters (2, 3) and a Gamma reference with shape and scale 2 and 0.14, respectively.

To show that any continuous reference pdf that is positive on its support which contains the support of the unknown target can be used in K-series, we present an example where the reference is exponential with scale parameter 0.2 in Fig. 3.18, and an example where the reference is Gamma with shape parameter 2 in Fig. 3.19. The latter serves to illustrate that the requirement for the reference to be positive everywhere on its support is sufficient, but not necessary, in general. The limit at point $x = 0$ of the ratio $f^2(x)/\phi(x)$, in this case, is zero and the integral exists. But in general, this condition cannot be checked when the true target pdf is unknown.

3.5 Conclusion

K-series is a general distribution recovery method that approximates the density function as a series in terms of a finite number of its moments. It targets the unknown density via the choice of the reference probability density function and includes existing series-based density estimation, such as Gram-Charlier, as special cases. The K-series estimator converges to the true pdf in L_1, satisfies the moment matching principle, and is fast to compute. The method is complemented by an estimation algorithm of the minimal support of the target distribution.

K-series requires the target pdf have bounded support. This is not a serious limitation since, in practice, as in nature, observable values occur with effectively nonzero probability within an interval, and values outside a certain range are never realized. The choice of the reference is based on subject-matter knowledge, if available. We study the effect of the reference pdf on estimation in Appendix 3.4. The uniform reference distribution results in accurate estimates provided its support is close to the support of the true pdf. Both truncated and regular normal reference pdfs lead to accurate K-series estimates the closer the target pdf is to a normal. Overall, the truncated normal distribution typically results in better estimation.

Characterizing the distribution of random quantities generated in probabilistic programming languages (PPLs) [BKS20a] is essential: Distributions are the building blocks of inference. PPLs codify probabilistic models and are used, for example, in computer security/privacy protocols [Dwo06], distributed consensus algorithms [Her90], randomized algorithms [Raj95], generative machine learning models [Gha15] and scenario-based testing [FDG+19] of cyber-physical systems operating in uncertain environments.

In future work, we will extend K-series to recover probability mass functions for discrete random variables. We also aim to compute error bounds and explore the Fourier series representation of functions in conjunction with [JWW21], which obtains exact moments for sine and cosine assignments, to reduce the estimation error for fixed loop iterations. We will also develop a tool to automate the entire procedure in Algorithm 1.

4 Exact Upper and Lower Bounds for the Output Distribution of Neural Networks with Random Inputs

This chapter is based on the following preprint: [KKB+25]

- *Andrey Kofnov, Daniel Kapla, Ezio Bartocci, and Efstathia Bura. Exact Upper and Lower Bounds for the Output Distribution of Neural Networks with Random Inputs.*

4.1 Introduction

Increased computational power, availability of large datasets, and the rapid development of new NN architectures contribute to the ongoing success of neural network (NN) based learning in image recognition, natural language processing, speech recognition, robotics, strategic games, etc. A limitation of NN machine learning (ML) approaches is that it is difficult to infer the uncertainty of the predictions from their results: there is no internal mechanism in NN learning to assess how trustworthy the network outputs are with *unseen* data. A NN is a model of the form

$$Y = f(X, \boldsymbol{\Theta}), \tag{4.1}$$

where Y is the output and X the input (typically multivariate), and f is a *known* function modeling the relationship between X and Y parametrized by Θ. Model (4.1) incorporates uncertainty neither in Y nor in X and NN fitting is a numerical algorithm for minimizing a loss function. Lack of uncertainty quantification, such as assessment mechanisms for prediction accuracy beyond the training data, prevents neural networks, despite their potential, from being deployed in safety-critical applications ranging from medical diagnostic systems (e.g., [HB20]) to cyber-physical systems such as autonomous vehicles, robots or drones (e.g., [YLC+20]). Also, the deterministic nature of NNs renders them highly vulnerable to not only adversarial noise but also to even small perturbations in inputs ([BAG18; FFF18; GPM+14; GSS14; HXP17]).

Uncertainty in modeling is typically classified as *epistemic* or *systematic*, which derives from lack of knowledge of the model, and *aleatoric* or *statistical*, which reflects the inherent randomness in the underlying process being modeled (see, e.g., [HW21]). The *universal approximation theorem* (UAT) [Cyb89; HSW89] states that a neural network with one hidden layer can approximate any continuous function for inputs within a specific range by increasing the number of neurons. When working with neural networks, epistemic uncertainty is often considered less significant than aleatoric uncertainty, which directly impacts predictions and is usually prioritized in tasks requiring

robust modeling against data variability. Herein, we focus on studying the effect of random inputs on the output distribution of NNs and derive uniform upper and lower bounds for the cumulative distribution function (cdf) of the outputs of a NN subject to noisy (stochastic) input data.

We evaluate our proposed framework on three benchmark datasets (Iris [Fis36], Wine [AF92] and Diabetes [EHJ+04]) and demonstrate the efficacy of our approach to bound the cdf of the NN output subject to Gaussian and Gaussian mixture inputs. We demonstrate that our bounds cover the true underlying cdf over its entire support. In contrast, the competing approach of [KHM+24], as well as large sample Monte-Carlo simulations, are shown to produce estimates outside the bounds over several areas of the output range.

4.2 Problem Overview

A neural network is a mathematical model that produces outputs from inputs. The input is typically a vector of predictor variables, $\mathbf{X} \in \mathbb{R}^{n_0}$, and the output Y, is univariate or multivariate, continuous or categorical.

A *feedforward NN* with L layers from $\mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ is a composition of L functions,

$$f_L(\mathbf{x}; \boldsymbol{\Theta}) = f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}(\mathbf{x}), \tag{4.2}$$

where the l-th layer is given by

$$f^{(l)}(\mathbf{x}; \mathbf{W}^{(l)}, \mathbf{b}^{(l)}) = \sigma^{(l)}(\mathbf{W}^{(l)}\mathbf{x} + \mathbf{b}^{(l)}),$$

with weights $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, bias terms $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, and a non-constant, continuous activation function $\sigma^{(l)} : \mathbb{R} \to \mathbb{R}$ that is applied component-wise. The NN parameters are collected in $\boldsymbol{\Theta} = (\text{vec}(\mathbf{W}_1), \mathbf{b}_1, \ldots, \text{vec}(\mathbf{W}_L), \mathbf{b}_L) \in \mathbb{R}^{\sum_{l=1}^{L}(n_{l-1} \cdot n_l + n_l)}$.[1] The first layer that receives the input $\mathbf{x}$ is called the *input layer*, and the last is the *output layer*. All other layers are called *hidden*. For categorical outputs, the class label is assigned by applying a decision function, such as $\arg\max$ as the final step.

Despite not being typically acknowledged, the training data in NNs are drawn from larger populations, and hence they contain only limited information about the corresponding population. We incorporate the uncertainty associated with the observed data assuming that they are random draws from an unknown distribution of bounded support. That is, the data are comprised of m draws from the joint distribution of $(\mathbf{X}, Y)$, and the network is trained on observed $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in_0})$, and y_i, $i = 1, \ldots, m$.[2] A NN with L layers and n_l neurons at each layer, $l = 1, \ldots, L$, is trained on the observed $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, m$, to produce m outputs $\tilde{y}_i$, and the vector of the NN parameters, $\boldsymbol{\Theta} = (\text{vec}(\mathbf{W}_1), \mathbf{b}_1, \ldots, \text{vec}(\mathbf{W}_L), \mathbf{b}_L)$, is obtained. $\boldsymbol{\Theta}$ uniquely identifies the trained NN. Given $\boldsymbol{\Theta}$, we aim to quantify the robustness of the corresponding NN, to perturbations in the input variables. Let $F_{\mathbf{X}}$ denote the cumulative distribution function (cdf) of $\mathbf{X}$,

$$\mathbf{X} \sim F_{\mathbf{X}}, \tag{4.3}$$

[1] The operation vec : $\mathbb{R}^{n_{l-1} \times n_l} \to \mathbb{R}^{n_{l-1} \cdot n_l}$ stacks the columns of a matrix one after another.

[2] We use the convention of denoting random quantities with capital letters and their realizations (observed) by lowercase letters.

where $\mathbf{X} \in \mathbb{R}^{n_0}$ are the randomly perturbed input variables. We assume $\mathbf{X}$ has probability density function (pdf) $\phi(\mathbf{x})$ that is piecewise continuous and bounded on a compact support. We study the *propagation of uncertainty* (effect of the random perturbation) in the NN by deriving upper and lower bounds of the cdf $F_{\widetilde{\mathbf{Y}}}(y) = \mathbb{P}(\widetilde{\mathbf{Y}} \leq y)$ of the *random* output, $\widetilde{\mathbf{Y}} = f_L(\mathbf{X} \mid \mathbf{\Theta})$.[3]

Our contributions:

1. We develop a method to compute the exact cdf of the output of ReLU NNs with random input pdf, which is a piecewise polynomial over a compact hyperrectangle. This result, which can be viewed as a stochastic analog to the Stone-Weierstrass theorem,[4] significantly contributes to the characterization of the distribution of the output of NNs with piecewise linear activation functions under any input continuous pdf.

2. We derive *guaranteed* upper and lower bounds of the NN output distribution resulting from random input perturbations on a given support. This provides *exact* upper and lower bounds for the output cdf provided the input values fall within the specified support. No prior knowledge about the true cdf is required to guarantee the validity of our bounds.

3. We show the convergence of our bounds to the true cdf; that is, our bounds can be refined to arbitrary accuracy.

4. We provide a constructive proof that any feedforward NN with continuous monotonic *piecewise twice continuously differentiable*[5] activation functions can be approximated from above and from below by a fully connected ReLU network, achieving any desired level of accuracy. Moreover, we enable the incorporation of multivariate operations such as $\max$, $\mathrm{product}$ and $\mathrm{softmax}$, as well as some non-monotonic functions such as $|x|$ and $x^n, n \in \mathbb{N}$.

5. We prove a new *universal* ***distribution*** *approximation theorem* (UDAT), which states that we can estimate the cdf of the output of any continuous function of a random variable (or vector) that has a continuous distribution supported on a compact hyperrectangle, achieving any desired level of accuracy.

4.3 Our Approximation Approach

We aim to estimate the cdf $F_{\widetilde{\mathbf{Y}}}(y)$ of the output $\widetilde{\mathbf{Y}} = f_L(\mathbf{X} \mid \mathbf{\Theta})$ of the NN in (4.2) under (4.3); i.e., subject to random perturbations of the input $\mathbf{X}$. We do so by computing upper and lower bounds of $F_{\widetilde{\mathbf{Y}}}$; that is, we compute $\overline{F}_{\widetilde{\mathbf{Y}}}$, $\underline{F}_{\widetilde{\mathbf{Y}}}$ such that

$$\underline{F}_{\widetilde{\mathbf{Y}}}(y) \leq F_{\widetilde{\mathbf{Y}}}(y) \leq \overline{F}_{\widetilde{\mathbf{Y}}}(y), \forall y \tag{4.4}$$

We refer to the NN in (4.2) as *prediction NN* when needed for clarity. We estimate the functions $\overline{F}_{\widetilde{\mathbf{Y}}}$, $\underline{F}_{\widetilde{\mathbf{Y}}}$ on a "superset" of the output domain of the prediction NN (4.2) via an integration procedure.

[3] The notation $f_L(\mathbf{X} \mid \mathbf{\Theta})$ signifies that $\mathbf{\Theta}$, equivalently the NN, is fixed and only $\mathbf{X}$ varies.

[4] A significant corollary to the Stone-Weierstrass theorem is that any continuous function defined on a compact set can be uniformly approximated as closely as desired by a polynomial.

[5] A wide class of the most common continuous activation functions, including ReLU, $\tanh$ and logistic function.

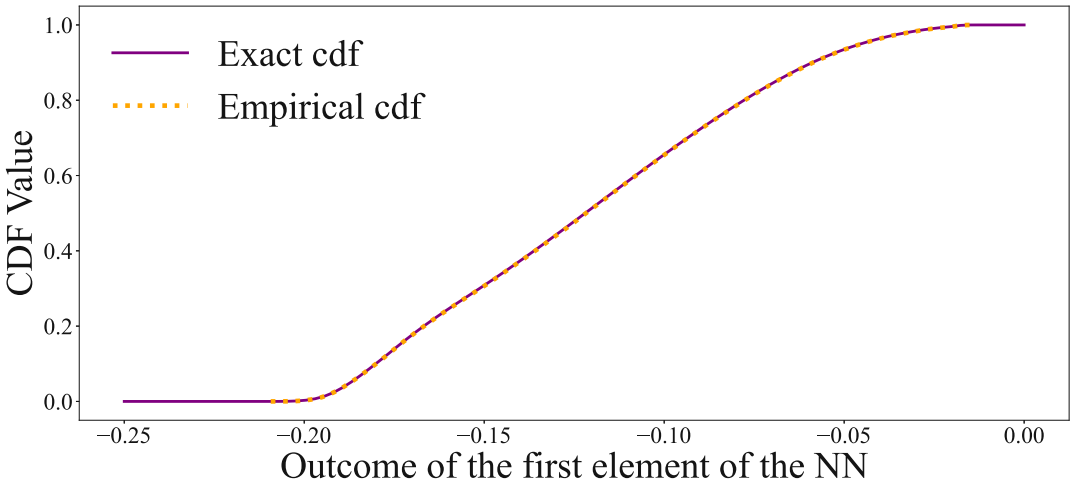


Figure 4.1: Exact CDF of the ReLU neural network outcome for the class *Setosa* in the Iris problem, assuming Beta-distributed inputs.

The cdf of $\widetilde{\mathbf{Y}}$ is given by

$$F_{\widetilde{\mathbf{Y}}}(y) = \mathbb{P}(\widetilde{\mathbf{Y}} \leq y) = \int_{\{\widetilde{\mathbf{Y}} \leq y\}} \phi(\mathbf{x}) d\mathbf{x}, \tag{4.5}$$

where $\phi(\mathbf{x})$ is the pdf of $\mathbf{X}$. To bound $F_{\widetilde{\mathbf{Y}}}$, we bound ϕ by its upper ($\overline{\phi}$) and lower ($\underline{\phi}$) estimates on the bounded support of ϕ as described in Section 4.3.4. If ϕ is a piecewise polynomial, then (4.5) can be computed exactly for a ReLU prediction network, as we show in Section 4.3.1. Once $\overline{\phi}$, $\underline{\phi}$ are estimated, then

$$\underline{F}_{\widetilde{Y}}(y) = \int_{\{\widetilde{\mathbf{Y}} \leq y\}} \underline{\phi}(\mathbf{x}) dx \leq F_{\widetilde{\mathbf{Y}}}(y) \leq \int_{\{\widetilde{\mathbf{Y}} \leq y\}} \overline{\phi}(\mathbf{x}) dx = \overline{F}_{\widetilde{\mathbf{Y}}}(y)$$

Remark 1. *$\underline{F}_{\widetilde{\mathbf{Y}}}(y)$ and $\overline{F}_{\widetilde{\mathbf{Y}}}(y)$ are not always true cdfs since we allow the lower estimator not to achieve 1, while the upper bound is allowed to take the smallest value greater than 0.*

4.3.1 Exact cdf evaluation for a fully connected NN with ReLU activation function

Definition 6 (Almost disjoint sets)**.** *We say that sets $\mathcal{A}$ and $\mathcal{B}$ are almost disjoint with respect to measure α, if $\alpha(\mathcal{A} \cap \mathcal{B}) = 0$.*

Definition 7 (Closed halfspace)**.** *A n_0-dimensional closed halfspace is a set $H = \{x \in \mathbb{R}^{n_0} | \mathbf{v}^T x \leq c\}$ for $c \in \mathbb{R}$ and some $\mathbf{v} \in \mathbb{R}^{n_0}$, which is called the normal of the halfspace.*

It is known that a convex polytope can be represented as an intersection of halfspaces, called $\mathcal{H}$-representation [Zie95].

Definition 8 ($\mathcal{H}$-polytope)**.** *A n_0-dimensional $\mathcal{H}$-polytope $\mathcal{P} = \bigcap_{j=1}^{h} H_i$ is the intersection of finitely many closed halfspaces.*

Definition 9 (Simplex)**.** *A n_0-dimensional simplex is a n_0-dimensional polytope with n_0+1 vertices.*

Definition 10 (Piecewise polynomial)**.** *A function $p : K \to \mathbb{R}^{n_L}$ is a* piecewise polynomial*, if there exists a finite set of n_0-simplices such that $K = \bigcup_{i=1}^{q} k_i$ and the function p constrained to the interior k_i° of k_i is a polynomial; that is, $p\big|_{k_i^{\circ}} : k_i^{\circ} \to \mathbb{R}^{n_L}$ is a polynomial for all $i = 1, \ldots, q$.*

Remark 2. *We do not require piecewise polynomials to be continuous everywhere on the hyperrectangle. Specifically, we allow discontinuities at the borders of simplices. However, the existence of left and right limits of the function at every point on the bounded support is guaranteed by properties of polynomials.*

[RPK+17] showed that ReLU deep networks divide the input domain into activation patterns (see [SKS+20]) that are disjoint convex polytopes $\{\mathcal{P}_j\}$ over which the output function is locally represented as the affine transformation $f_L(\mathbf{x}) = NN^j(\mathbf{x}) = \mathbf{c}^j + \mathbf{V}^j\mathbf{x}$ for $\mathbf{x} \in \{\mathcal{P}_j\}$, the number of which grows at the order $\mathcal{O}((\max\{n_l\}_{l=1,\ldots,L})^{n_0 L})$. [SKS+20] outline an algorithm for extracting the full set of polytopes and determining local affine transformations, including the coefficients $\mathbf{c}^j, \mathbf{V}^j$ for all $\{\mathcal{P}_j\}$, by propagating through the layers. For our computations, we utilize a recent GPU-accelerated algorithm from [Ber23].

We aim to derive a superset of the range of the network output. For this, we exploit the technique of Interval Bound Propagation (IBP), based on ideas from [GDS+18; WAP+22; GMD+18]. Propagating the n_0-dimensional box through the network leads to the superset of the range of the network output. We compute the cdf of the network's output at each point of a grid of the superset of the output range.

Theorem 9 (Exact cdf of ReLU NN w.r.t. piecewise polynomial pdf)**.** *Let $\widetilde{\mathbf{Y}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ be a feed-forward ReLU neural network, which splits the input space into a set of almost disjoint polytopes $\{\mathcal{P}_j\}_{j=1}^{q_Y}$ with local affine transformations $\widetilde{\mathbf{Y}}(\mathbf{x}) = NN^j(\mathbf{x})$ for $\mathbf{x} \in \mathcal{P}_j$. Let $\phi(\mathbf{x})$ denote the pdf of the random vector $\mathbf{X}$ that is a piecewise polynomial with local polynomials, $\phi(\mathbf{x}) = \phi_i(\mathbf{x})$ for all $\mathbf{x} \in k_i^{\circ}$ over an almost disjoint set of simplices $\{k_i\}_{i=1}^{q_\phi}$, and a compact hyperrectangle support $K \subset \mathbb{R}^{n_0}$. Then, the cdf of $\widetilde{\mathbf{Y}}$ is*

$$F_{\widetilde{\mathbf{Y}}}(\mathbf{y}) = \mathbb{P}\left[\widetilde{\mathbf{Y}} \leq \mathbf{y}\right] = \sum_{i=1}^{q_\phi}\sum_{j=1}^{q_Y} \mathcal{I}\left[\phi_i(\mathbf{x}); \mathcal{P}_{j,i}^r\right] = \sum_{i=1}^{q_\phi}\sum_{j=1}^{q_Y}\sum_{s=1}^{S_{i,j}} \mathcal{I}\left[\phi_i(\mathbf{x}); \mathcal{T}_{i,j,s}\right],$$

where $\mathcal{I}\left[\phi_i(\mathbf{x}); \mathcal{T}_{i,j,s}\right]$ is the integral of the polynomial $\phi_i(\mathbf{x})$ over the simplex $\mathcal{T}_{i,j,s}$ such that the reduced polytope

$$\mathcal{P}_{j,i}^r = \mathcal{P}_j \cap k_i \cap \{\mathbf{x} : NN^j(\mathbf{x}) \leq \mathbf{y}\} = \bigcup_{s=1}^{S_{i,j}} \mathcal{T}_{i,j,s} \tag{4.6}$$

is defined by the intersection of polytopes $\mathcal{P}_j$ and k_i, and the intersection of halfspaces

$$\{\mathbf{x} : NN^j(\mathbf{x}) \leq \mathbf{y}\} = \bigcap_{t=1}^{n_L} \left\{\mathbf{x} : NN_t^j(\mathbf{x}) \leq y_t\right\}.$$

Proof. Suppose the activation function in the prediction NN (4.2) is ReLU and n_0 and n_L are the number of input and output neurons, respectively. The integral of a function over a given domain can be expressed as the sum of integrals over a partition of the domain (disjoint subdomains whose union constitutes the original domain). To compute the cdf of $\widetilde{\mathbf{Y}} = f_L(\mathbf{x})$ at y, $F_{\widetilde{\mathbf{Y}}}(y) = \Pr[f_L(\mathbf{x}) \leq y]$, we compute the sum of $\mathbb{P}[NN^j(\mathbf{x}) \leq y \mid \mathbf{x} \in \mathcal{P}_j]$, each of which is the integral of the pdf of the input over the given polytopes subject to $NN^j(\mathbf{x}) \leq y$. These sets of polytopes $\{\mathcal{P}_j\}$ and the corresponding local affine transformations $\{NN^j(\mathbf{x})\}$ always exist, as shown in [RPK+17].

$\mathcal{P}_j$ is a convex polytope and can be represented as the intersection of halfspaces (see [Zie95]). The set $\{\mathbf{x} : NN^j(\mathbf{x}) = \mathbf{c}^j + \mathbf{V}^j\mathbf{x} \leq y\}$ is defined as the intersection of halfspaces

$$\bigcap_{t=1}^{n_L} \left\{ \mathbf{x} : NN_t^j(\mathbf{x}) = c_t^j + \sum_{z=1}^{n_0} x_z v_{t,z}^j \leq y_t \right\},$$

which when intersected with $\mathcal{P}_j$ defines the reduced complex polytope $\mathcal{P}_{j,i}^r$. The desired local probability, $\mathbb{P}[NN^j(\mathbf{x}) \leq y \mid \mathbf{x} \in \mathcal{P}_j]$, can be found as the integral of the pdf of $\mathbf{X}$ over the reduced polytope.

Using the Delaunay triangulation (see [Del34]) one can decompose any convex polytope $\mathcal{P}_{j,i}^r$ into a disjoint set of simplices $\mathcal{T}_{i,j,s}$. This triangulation allows us to compute the integral over the polytope as a sum of integrals over each simplex. Assuming that the pdf of the input is a piecewise polynomial allows us to use the algorithm from [Las21] to compute exact integrals over all simplices. The sum of all these localized integrals (probabilities) is the exact cdf value at point y. □

The proof relies on the algorithm for evaluating the integral of a polynomial over a simplex, as described in [Las21], justifying its applicability. The right-hand side of (4.6) results from the Delaunay triangulation [Del34], dividing the reduced polytope $\mathcal{P}_{j,i}^r$ into $S_{i,j}$ almost disjoint simplices.

Remark 3. *Theorem 9 is a tool for approximating the output cdf of any feedforward neural network with piecewise linear activation functions on a compact domain, given random inputs with arbitrary continuous distribution at any desired degree of accuracy (see 4).*

Remark 4. *Theorem 9 serves as an applicable tool for deriving the exact cdf of the ReLU neural network on a compact support with a piecewise polynomial pdf. This contrasts with Theorem 2 from [KHM+24], which provides a formal theoretical result for computing the pdf of a piecewise affine function of a random variable with a piecewise pdf but does not offer a practical method for exact computation. Specifically, [KHM+24] offer no method to evaluate the underlying integral, except for the special case of a piecewise constant input pdf, as done later in their Theorem 3.*

Example 6. *We compute the output cdf of a 3-layer, 12-neuron fully connected ReLU neural network with the last (before* softmax*) linear 3-neuron layer trained on the Iris dataset [Fis36]. The Iris dataset consists of 150 samples of iris flowers from three different species: Setosa, Versicolor, and Virginica. Each sample includes four features: Sepal Length, Sepal Width, Petal length, and Petal width. We focused on two features, Sepal Length and Sepal Width (scaled to* $[0, 1]$*), classifying objects into three classes. Specifically, we recovered the distribution of the first component (class*

Setosa) before applying the softmax *function, assuming Beta-distributed inputs with parameters* $(2, 2)$ *and* $(3, 2)$*. The exact cdf is plotted in purple in Figure 4.1, with additional details provided in Section 4.4.1. The agreement with the empirical cdf is almost perfect.*

4.3.2 Algorithm for Upper and Lower Approximation of the Neural Network using ReLU activation functions.

Theorem 10. *Let* $\widetilde{Y}$ *be a feedforward neural network with* L *layers of arbitrary width with continuous activation functions. There exist sequences of fully connected ReLU neural networks* $\{\overline{Y}_n\}$, $\{\underline{Y}_n\}$*, which are monotonically decreasing and increasing, respectively, such that for any* $\epsilon > 0$ *and any compact hyperrectangle* $K \subset \mathbb{R}^{n_0}$*, one can find* $N \in \mathbb{N}$ *such that for all* $n \geq N$

$$0 \leq \widetilde{Y}(\mathbf{x}) - \underline{Y}_n(\mathbf{x}) < \epsilon, \quad 0 \leq \overline{Y}_n(\mathbf{x}) - \widetilde{Y}(\mathbf{x}) < \epsilon$$

for all $\mathbf{x} \in K$*.*

Proof. Since $\widetilde{Y}$ is a feedforward neural network with continuous activation functions on a compact support, for any $\epsilon > 0$, let $\{\epsilon_n\}$, $\epsilon > \epsilon_n > 0$ be a decreasing sequence. By the universal approximation theorem (UAT) [HSW89], there exists a sequence of ReLU networks $\{Y_{\epsilon_n}\}$, such that

$$\sup_K \|\widetilde{Y}(\mathbf{x}) - Y_{\epsilon_n}(\mathbf{x})\|_{\mathbb{R}^{n_0}} < \epsilon_n.$$

Setting $\underline{Y}'_n(\mathbf{x}) = Y_{\epsilon_n}(\mathbf{x}) - \epsilon_n$ and $\overline{Y}'_n(\mathbf{x}) = Y_{\epsilon_n}(\mathbf{x}) + \epsilon_n$, we have

$$\underline{Y}'_n(\mathbf{x}) \leq \widetilde{Y}(\mathbf{x}) \leq \overline{Y}'_n(\mathbf{x}).$$

Now, we let $\underline{Y}_n(\mathbf{x}) = \max_{1 \leq i \leq n} \underline{Y}'_n(\mathbf{x})$, which is still not greater than $\widetilde{Y}(\mathbf{x})$, and $\overline{Y}_n(\mathbf{x}) = \min_{1 \leq i \leq n} \overline{Y}'_n(\mathbf{x})$, which is still not smaller than $\widetilde{Y}(\mathbf{x})$. One can see that $\{\overline{Y}_n\}$ and $\{\underline{Y}_n\}$ are monotonically decreasing and increasing, respectively. It should be noted that the $\min$ and $\max$ operators can be represented as ReLU networks (see Fig. 4.4), and the composition of ReLU networks is itself a ReLU network. □

Definition 11. *Let* $\Omega = [\underline{a}, \overline{a}] \subset \mathbb{R}$ *be a closed interval and* $g : \Omega \to \mathbb{R}$ *is well-defined, continuous on* Ω*. We will say that* g *is* ***piecewise twice continuously differentiable*** *on* Ω*, that is* $g \in \mathcal{C}^2_{p.w.}(\Omega)$*, if there exists a finite partition of* Ω *into closed subintervals* $\bigcup_{i=1}^{n} [a_i, a_{i+1}] = \Omega$ *where* $\underline{a} = a_1 < a_2 < \ldots < a_{n+1} = \overline{a}$ *and* $n \in \mathbb{N}$*, such that* $g\big|_{[a_i, a_{i+1}]}$ *is twice continuously differentiable.*

We develop an approach to approximate the activation functions of a neural network provided they are non-decreasing continuous functions in $\mathcal{C}^2_{\text{p.w.}}(\Omega_i)$ at each node i, where Ω_i is the input domain of node i. Consequently, the output of the neural network is approximated using the ReLU activation function to create piecewise linear upper and lower bounds. These bounds approximate the true activation function, making the neural network more analyzable. This method provides a constructive proof for the restricted case of Theorem 10 for non-decreasing activation functions from $\mathcal{C}^2_{\text{p.w.}}$ and applies to any fully connected or convolutional (CNN) feedforward neural network with non-decreasing continuous piecewise twice continuously differentiable activation functions analyzed on a hyperrectangle. The key features of our approach are:

- **Local adaptability:** The algorithm adapts to the curvature of the activation function, providing an adaptive approximation scheme depending on whether the function is locally convex or concave.
- **Streamlining:** By approximating the network with piecewise linear functions, the complexity of analyzing the network output is significantly reduced.

How it works:

Input/Output range evaluation: Using IBP [GDS+18], we compute supersets of the input and output ranges of the activation function for every neuron and every layer.

Segment Splitting: First, input intervals are divided into macro-areas based on inflection points (different curvature areas) and points of discontinuity in the first or second derivative (e.g., 0 for ReLU). Next, these macro-areas are subdivided into intervals based on user-specified points or their predefined number within each range. The algorithm utilizes knowledge about the behavior of the activation function and differentiates between concave and convex regions of the activation function, which impacts how the approximations are constructed and how to choose the points of segment splitting. A user defines the number of splitting segments and the algorithm ensures the resulting disjoint sub-intervals are properly ordered and on each sub-interval the function is either concave or convex. If the function is linear in a given area, it remains unchanged, with the upper and lower approximations equal to the function itself.

Upper and Lower Approximations: The method constructs tighter upper and lower bounds through the specific choice of points and subsequent linear interpolation. It calculates new points within each interval (one per interval) and uses them to refine the approximation, ensuring the linear segments closely follow the curvature of the activation function.

The method guarantees that the piecewise linear approximation of the activation function for each neuron (a) is a non-decreasing function, and (b) the output domain remains the same.

To see this, consider a neuron of a layer. For upper (lower) approximation on a convex (concave) segment, we choose a midpoint $a_{k'}^{lin\_int} = (a_k + a_{k+1})/2$ for each subinterval $[a_k, a_{k+1}]$ and compute a linear interpolation, as follows:

$$\kappa_1 = \frac{f(a_{k'}^{lin\_int}) - f(a_k)}{a_{k'}^{lin\_int} - a_k}, \qquad \kappa_2 = \frac{f(a_{k+1}) - f(a_{k'}^{lin\_int})}{a_{k+1} - a_{k'}^{lin\_int}},$$

$$\widetilde{f}^{lin\_int}(\tau) = \begin{cases} f(a_k) + (\tau - a_k)\kappa_1, & \tau \in \left[a_k, a_{k'}^{lin\_int}\right] \\ f(a_{k'}^{lin\_int}) + (\tau - a_{k'})\kappa_2, & \tau \in \left[a_{k'}^{lin\_int}, a_{k+1}\right] \end{cases}$$

For upper (lower) approximation on a concave (convex) segment, we compute derivatives and look for tangent lines at border points of the sub-interval $[a_k, a_{k+1}]$. We choose a point $a_{k'}^{pie\_tan} : a_k \leq a_{k'}^{pie\_tan} \leq a_{k+1}$ to be the intersection of the tangent lines. The original function is approximated

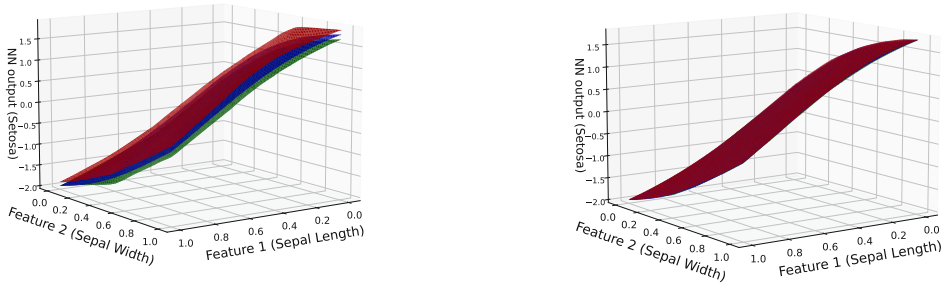


Figure 4.2: *Tanh* NN output for the *Setosa* class (blue) in the Iris dataset, and its ReLU NN upper (red) and lower (green) approximations with 5 (left panel) and 10 (right panel) segments of bounding per convex section.

by the following two tangent line segments:

$$a_{k'}^{pie\_tan} = \frac{f(a_k) - f(a_{k+1}) - (f'_+(a_k)a_k - f'_-(a_{k+1})a_{k+1})}{f'_-(a_{k+1}) - f'_+(a_k)}$$

$$\widetilde{f}^{pie\_tan}(\tau) = \begin{cases} f(a_k) + f'_+(a_k)(\tau - a_k), & \tau \in \left[a_k, a_{k'}^{pie\_tan}\right] \\ f(a_{k+1}) + f'_-(a_{k+1})(\tau - a_{k+1}), & \tau \in \left[a_{k'}^{pie\_tan}, a_{k+1}\right], \end{cases}$$

where $f'_-(\cdot), f'_+(\cdot)$ are left and right derivatives, respectively.

For monotonically increasing functions, this procedure guarantees that the constructed approximators are exact upper and lower approximations. Moreover, decreasing the step size (increasing the number of segments) reduces the error at each point, meaning that the sequences of approximators for the activation functions at each node are monotonic, that is $\overline{f}_{n+1}(x) \leq \overline{f}_n(x), \underline{f}_{n+1}(x) \geq \underline{f}_n(x)$ for all x in the IBP domain. This procedure of piecewise linear approximation of each activation function in each neuron is equivalent to the construction of a one-layer ReLU network for a given neuron which will approximate it. By the UAT, for any continuous activation function $\sigma_{l,i}$ at each neuron and any positive $\epsilon_{l,i}$ we can always find a ReLU network-approximator $NN_{l,i}$, such that $|NN_{l,i}(x) - \sigma_{l,i}(x)| < \epsilon_{l,i}$ for all x in the input domain, defined by the IBP. To find such an approximating network we need to choose the corresponding number of splitting segments of the IBP input region. Additionally, uniform convergence is preserved by Dini's theorem, which states that a monotonic sequence of continuous functions that converges pointwise on a compact domain to a continuous function also converges uniformly. Moreover, the approximator always stays within the range of the limit function, ensuring that the domain in the next layer remains unchanged and preserves its uniform convergence.

Transformation into ReLU-Equivalent Form: The approximating piecewise linear upper-lower functions are converted into a form that mimics the ReLU function's behavior; i.e., $\tilde{f}(x) =$

$W^{(2)}\mathrm{ReLU}(W^{(1)}x + b^{(1)}) + b^{(2)}$. This involves creating new weighting coefficients and intercepts that replicate the ReLU's activation pattern across the approximation intervals.

Specifically, we receive a set of intervals $\{[x_{i-1}, x_i]\}_{i=1}^n$ with the corresponding set of parameters of affine transformations $\{(a_i, b_i)\}_{i=1}^n$:

$$\widetilde{f}(\tau) = b_i + a_i\tau, \qquad \tau \in [x_{i-1}, x_i]\,,$$

and additionally set $a_0 = 0$. Then, the corresponding ReLU-equivalent definition of approximation $\widetilde{f}$ on $[x_0, x_n]$ is

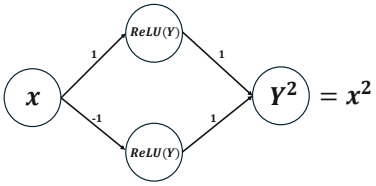
$$\widetilde{f}(\tau) = x_0a_1 + b_1 + \sum_{i=1}^{n} \xi_i \mathrm{ReLU}(|a_i - a_{i-1}|(\tau - x_{i-1})),$$
$$\xi_i = \mathrm{sign}(a_i - a_{i-1}).$$

Full Neural Network Approximation: The entire NN is approximated by applying the above techniques to each neuron, layer by layer, and then merging all intermediate weights and biases. For each neuron, both upper and lower approximations are generated, capturing the range of possible outputs under different inputs. To ensure the correct propagation of approximators, to create an upper approximation, we connect the upper approximation of the external layer with the upper approximation of the internal subnetwork if the internal subnetwork has a positive coefficient, or with the lower approximation if it has a negative coefficient. The reverse applies to the lower approximation. This leads to a composition of uniformly convergent sequences, guaranteeing the overall uniform convergence of the final estimator to the original neural network. The final output is a set of piecewise linear approximations that bound the output of the original neural network, which can then be used for further analysis or verification.

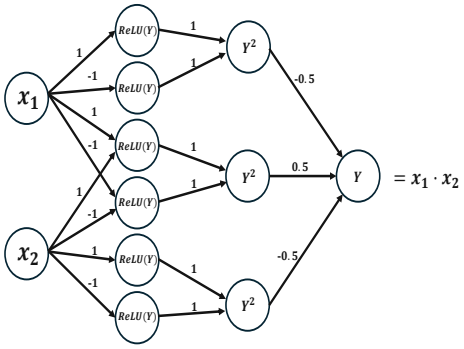
Example 7. *Using the same setup as in Example 6, we train a fully connected neural network with 3 layers of 12 neurons each with* $\tanh$ *activation followed by 1 layer of 3 output neurons with linear activation on the Iris dataset, focusing on the re-scaled features Sepal Length and Sepal Width. We construct upper and lower approximations of the network's output by ReLU neural networks with linear output layer. Two approximations are performed: with 5 and 10 segments of bounding per convex section at each node (left and right panels of Figure 4.2, respectively). Notably, with 10 segments, the original network and its approximations are nearly indistinguishable.*

This procedure applies to various non-monotonic functions, such as monomials $x^n, n \in \mathbb{N}$. These functions can be attained by a sequence of transformations that ensure monotonicity at all intermediate steps. These transformations can be represented as a subnetwork. Furthermore, multivariate functions like $\mathrm{softmax}$ and the product operation also have equivalent subnetworks with continuous monotonic transformations, as shown next.

Square. The quadratic function x^2 is not monotone on an arbitrary interval. But x^2 is monotonic on $[0, \infty)$. We can modify the form of the function by representing it as a subnetwork to be a valid set of sequential monotonic operations. Since $x^2 = |x|^2$, $x \in \mathbb{R}$, and the output range of $|x|$ is exactly $[0, \infty)$, we represent $|x|$ as a combination of monotonic ReLU functions, $|x| = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x)$. The resulting subnetwork is drawn in Figure 4.3a.



(a) Subnetwork equivalent to operation of taking a square.



(b) Subnetwork equivalent to the product operation.

Figure 4.3

Product of two values. To find a product of two values x_1 and x_2 one can use the formula $x_1 \cdot x_2 = 0.5 \cdot ((x_1 + x_2)^2 - x_1^2 - x_2^2)$. This leads us to the feedforward network structure in Figure 4.3b.

Maximum of two values. The maximum operation can be expressed via a subnetwork with ReLU activation functions only, as follows. Observing that $\max\{x_1, x_2\} = 0.5 \cdot (x_1 + x_2 + |x_1 - x_2|)$ results in the corresponding network structure in Figure 4.4.

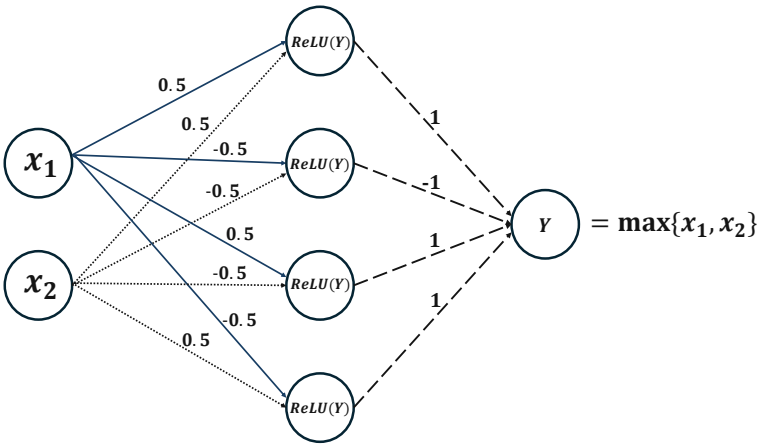


Figure 4.4: Subnetwork equivalent to a maximum of two values.

Softmax. The function softmax transforms a vector of real numbers to a probability distribution. That is, if $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, then there is a multivariate function $SfMax : \mathbb{R}^n \to \mathbb{R}^n$, so that

$$SfMax_i = \text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=i}^{n} e^{x_j}}$$

Then, $log(SfMax_i) = x_i - \log \sum_{j=i}^{n} e^{x_j}$, which is a composition of monotonic functions. This leads to the feedforward network structure in Figure 4.5.

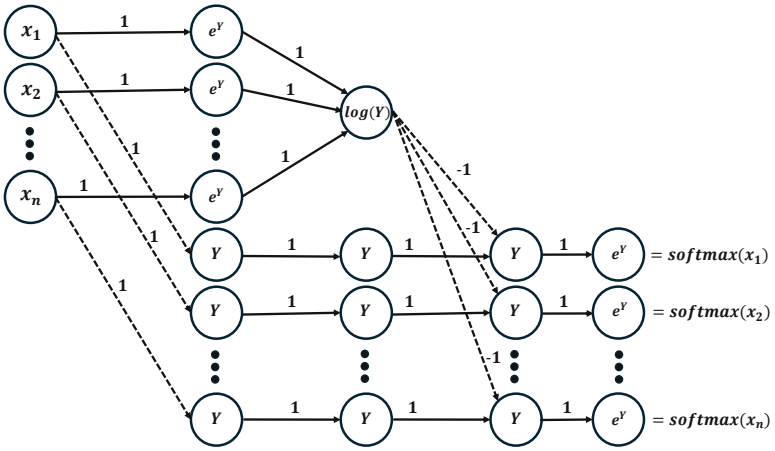


Figure 4.5: Subnetwork equivalent to one softmax node.

4.3.3 Convergence of the approximation

The main result of this section is Theorem 13 that shows that the upper and lower ReLU bounds converge uniformly and monotonically to the target NN. The proof requires a several steps that are shown in the following lemmas.

Lemma 2. *Let $a_k, a_{k+1} \in \mathbb{R}$ with $a_{k+1} > a_k$ and assume $f : [a_k, a_{k+1}] \to \mathbb{R}$ is twice continuously differentiable, monotone increasing and strictly convex (concave) on $[a_k, a_{k+1}]$. Then the values of the linear interpolation and piecewise tangent approximation at boundary points coincide with the original function. That is, $\widetilde{f}^{lin\_int}(a_k) = \widetilde{f}^{pie\_tan}(a_k) = f(a_k)$ and $\widetilde{f}^{lin\_int}(a_{k+1}) = \widetilde{f}^{pie\_tan}(a_{k+1}) = f(a_{k+1})$.*

Proof.

$$\widetilde{f}^{lin\_int}(a_k) = f(a_k) + (a_k - a_k)\kappa_1 = f(a_k),$$

$$\begin{aligned}\widetilde{f}^{lin\_int}(a_{k+1}) &= f(a_{k'}^{lin\_int}) + (a_{k+1} - a_{k'}^{lin\_int})\kappa_2 \\ &= f(a_{k'}^{lin\_int}) + (a_{k+1} - a_{k'}^{lin\_int})\frac{f(a_{k+1}) - f(a_{k'}^{lin\_int})}{a_{k+1} - a_{k'}^{lin\_int}} \\ &= f(a_{k+1}),\end{aligned}$$

$$\widetilde{f}^{pie\_tan}(a_k) = f(a_k) + f'_+(a_k)(a_k - a_k) = f(a_k),$$

$$\widetilde{f}^{pie\_tan}(a_{k+1}) = f(a_{k+1}) + f'_-(a_{k+1})(a_{k+1} - a_{k+1}) = f(a_{k+1}).$$

□

Lemma 3. *Let* $[a_k, a_{k+1}] \in \mathbb{R}$ *be a closed interval with* $a_{k+1} > a_k$*, and let* $f : [a_k, a_{k+1}] \to \mathbb{R}$ *be twice continuously differentiable, monotonically increasing, and strictly convex (or concave) on* $[a_k, a_{k+1}]$*. Then, the intermediate points lie strictly within the interval* $[a_k, a_{k+1}]$*,* $a_k < a_{k'}^{lin\_int} < a_{k+1}$ *and* $a_k < a_{k'}^{pie\_tan} < a_{k+1}$*, and the functions of linear interpolation and piecewise tangent approximation are continuous on* $[a_k, a_{k+1}]$*.*

Proof. For linear interpolation, the statement follows immediately from the definition. Specifically, for the midpoint interpolation, we have

$$a_{k'}^{lin\_int} = \frac{a_k + a_{k+1}}{2},$$
$$a_k < \frac{a_k + a_{k+1}}{2} < a_{k+1}.$$

For piecewise tangent approximation, we define $a_{k'}^{pie\_tan}$ as

$$a_{k'}^{pie\_tan} = \frac{f(a_k) - f(a_{k+1}) - \left(f'_+(a_k)a_k - f'_-(a_{k+1})a_{k+1}\right)}{f'_-(a_{k+1}) - f'_+(a_k)}.$$

Consider first the case where f is convex on $[a_k, a_{k+1}]$. By the convexity of f, we have the inequality

$$(a_k - a_{k+1})f'_-(a_{k+1}) < f(a_k) - f(a_{k+1}) < (a_k - a_{k+1})f'_+(a_k).$$

Adding the terms $f'_-(a_{k+1})a_{k+1} - f'_+(a_k)a_k$ to each part of the inequality, we get

$$\begin{aligned} a_k f'_-(a_{k+1}) - a_k f'_+(a_k) &< f(a_k) - f(a_{k+1}) - f'_+(a_k)a_k + f'_-(a_{k+1})a_{k+1} \\ &< a_{k+1}f'_-(a_{k+1}) - a_{k+1}f'_+(a_k). \end{aligned}$$

Since the denominator of $a_{k'}^{pie\_tan}$, i.e., $f'_-(a_{k+1}) - f'_+(a_k)$, is strictly positive by convexity, dividing the entire inequality by this denominator yields

$$a_k < a_{k'}^{pie\_tan} < a_{k+1}.$$

In the case where f is concave on $[a_k, a_{k+1}]$, we have a similar inequality:

$$(a_{k+1} - a_k)f'_-(a_{k+1}) < f(a_{k+1}) - f(a_k) < (a_{k+1} - a_k)f'_+(a_k).$$

Adding the terms $f'_+(a_k)a_k - f'_-(a_{k+1})a_{k+1}$ to each part of the inequality, we get

$$\begin{aligned} a_k f'_+(a_k) - a_k f'_-(a_{k+1}) &< f(a_{k+1}) - f(a_k) + f'_+(a_k)a_k - f'_-(a_{k+1})a_{k+1} \\ &< a_{k+1}f'_+(a_k) - a_{k+1}f'_-(a_{k+1}). \end{aligned}$$

Since the denominator $f'_-(a_{k+1}) - f'_+(a_k)$ of $a_{k'}^{pie\_tan}$ is strictly negative for concave functions, dividing the entire inequality by this negative denominator yields

$$a_k < a_{k'}^{pie\_tan} < a_{k+1}.$$

Next, we show that the interpolation functions are continuous at the intermediate point. For linear interpolation, we check the continuity by verifying that the two parts meet at $a_{k'}^{lin\_int}$. We have

$$f(a_{k'}^{lin\_int}) = f(a_{k'}^{lin\_int}) + (a_{k'}^{lin\_int} - a_{k'}^{lin\_int})\kappa_2 \overset{\text{right}}{=} \widetilde{f}^{lin\_int}(a_{k'}^{lin\_int}),$$
$$\overset{\text{left}}{=} f(a_k) + (a_{k'}^{lin\_int} - a_k)\kappa_1 = f(a_k) + (a_{k'}^{lin\_int} - a_k)\frac{f(a_{k'}^{lin\_int}) - f(a_k)}{a_{k'}^{lin\_int} - a_k} = f(a_{k'}^{lin\_int}).$$

Thus, the two parts of the linear interpolation meet at $a_{k'}^{lin\_int}$, and $\widetilde{f}^{lin\_int}$ is continuous at $a_{k'}^{lin\_int}$.

For piecewise tangent approximation, we check that the left and right parts meet at $a_{k'}^{pie\_tan}$. From the left, we have

$$\widetilde{f}^{pie\_tan}(\tau) = f(a_k) + f'_+(a_k)(\tau - a_k), \quad \tau \in [a_k, a_{k'}^{pie\_tan}].$$

Substituting $\tau = a_{k'}$, we get

$$\widetilde{f}^{pie\_tan}(a_{k'}^-) = f(a_k) + f'_+(a_k)(a_{k'}^- - a_k).$$

From the right, we have

$$\widetilde{f}^{pie\_tan}(\tau) = f(a_{k+1}) + f'_-(a_{k+1})(\tau - a_{k+1}), \quad \tau \in [a_{k'}^{pie\_tan}, a_{k+1}].$$

Substituting $\tau = a_{k'}$, we get

$$\widetilde{f}^{pie\_tan}(a_{k'}^+) = f(a_{k+1}) + f'_-(a_{k+1})(a_{k'}^+ - a_{k+1}).$$

Setting these equal, we have

$$f(a_k) + f'_+(a_k)(a_{k'} - a_k) = f(a_{k+1}) + f'_-(a_{k+1})(a_{k'} - a_{k+1}).$$

Solving for $a_{k'}$, we obtain

$$a_{k'} = \frac{f(a_k) - f(a_{k+1}) - (a_k f'_+(a_k) - a_{k+1} f'_-(a_{k+1}))}{f'_-(a_{k+1}) - f'_+(a_k)} = a_{k'}^{pie\_tan}.$$

This confirms the continuity of $\widetilde{f}^{pie\_tan}(\tau)$ at $a_{k'}^{pie\_tan}$.

□

Lemma 4. *Let $a_{k+1} > a_k$ and $[a_k, a_{k+1}] \subset \mathbb{R}$, and let $f : [a_k, a_{k+1}] \to \mathbb{R}$ be a twice continuously differentiable, monotonically increasing, and strictly convex (or strictly concave) function on $[a_k, a_{k+1}]$. Then, the estimating functions defined by linear interpolation $\widetilde{f}^{lin\_int}$ and piecewise tangent approximation $\widetilde{f}^{pie\_tan}$ are non-decreasing on $[a_k, a_{k+1}]$.*

Proof. We prove that both local estimators are non-decreasing functions.

Case 1: Linear Interpolation. The slopes of the linear interpolation segments are given by

$$\kappa_1 = \frac{f(a_{k'}^{lin\_int}) - f(a_k)}{a_{k'}^{lin\_int} - a_k} > 0, \quad \kappa_2 = \frac{f(a_{k+1}) - f(a_{k'}^{lin\_int})}{a_{k+1} - a_{k'}^{lin\_int}} > 0.$$

Since the original function f is non-decreasing and $a_k < a_{k'}^{lin\_int} < a_{k+1}$, as established by Lemma 3, both slopes are non-negative, ensuring that the interpolated function is non-decreasing.

Case 2: Piecewise Tangent Approximation. The derivatives of both the left and right segments of the piecewise tangent approximation are non-negative due to the increasing nature of the approximated function. Furthermore, by Lemma 3, we have $a_k < a_{k'}^{pie\_tan} < a_{k+1}$, and the approximation remains continuous everywhere. Since both segments are non-decreasing linear functions, their combination also results in a non-decreasing estimator over $[a_k, a_{k+1}]$.

Thus, both estimation methods preserve the monotonicity of f. □

Lemma 5. *Let $[\underline{a}, \overline{a}] \subset \mathbb{R}$ be a closed interval, and let $f : [\underline{a}, \overline{a}] \to \mathbb{R}$ be a continuous function satisfying $f \in \mathcal{C}^2_{p.w.}([\underline{a}, \overline{a}])$. Assume that there exist points $\underline{a} = a_1 < a_2 < \cdots < a_{n+1} = \overline{a}$ for some $n \in \mathbb{N}$ such that f is twice continuously differentiable, monotonically increasing, and either strictly convex, strictly concave, or linear on each subinterval $[a_k, a_{k+1}]$ for $1 \leq k \leq n$. Then, the approximation method described in Section 4.3.2 constructs a continuous, non-decreasing estimating function $\widetilde{f}(x)$ over the entire interval $[\underline{a}, \overline{a}]$.*

Proof. The claim follows from the following observations:

a) Each subinterval $[a_k, a_{k+1}]$ is suitable for approximating f using either linear interpolation, piecewise tangent approximation, or local linear approximation.

b) All the methods mentioned in (a) ensure that the estimator matches the original function at the boundary points of each subinterval, by Lemma 2.

c) The approximations described in (a) are continuous within their respective subintervals (see Lemma 3).

d) The methods in (a) produce non-decreasing functions within each subinterval (see Lemma 4).

By sequentially linking the estimators across all segments $\{[a_k, a_{k+1}]\}$, we obtain a continuous, non-decreasing, piecewise linear function $\widetilde{f}(x)$ over $[\underline{a}, \overline{a}]$. □

Lemma 6 (Image of the local estimator)**.** *Let $f : [\underline{a}, \overline{a}] \to \mathbb{R}$ be continuous, $f \in \mathcal{C}^2_{p.w.}([\underline{a}, \overline{a}])$ with $\underline{a} = a_1 < a_2 < \ldots < a_{n+1} = \overline{a}$ for $n \in \mathbb{N}$, such that $f\big|_{[a_k, a_{k+1}]}$ is twice continuously differentiable, monotonic increasing and either strictly convex (concave), or linear on $[a_k, a_{k+1}]$ for $1 \leq k \leq n$. Then the image of the estimating function $\widetilde{f}(x)$ defined by the approximation method in Section 4.3.2 coincides with the image of the target function f, that is $f([\underline{a}, \overline{a}]) = \widetilde{f}([\underline{a}, \overline{a}])$.*

Proof. Since function f is continuous on a compact $[\underline{a}, \overline{a}]$, and monotonic, then it maps $[\underline{a}, \overline{a}]$ into the closed interval (compact) $[f(\underline{a}), f(\overline{a})] \in \mathbb{R}$. But the estimator $\widetilde{f}$ is also continuous monotonic function $[\underline{a}, \overline{a}]$, and by Lemmas 2, 3, $f(\underline{a}) = \widetilde{f}(\underline{a})$ and $f(\overline{a}) = \widetilde{f}(\overline{a})$. That is why, the ranges of values of f and $\widetilde{f}$ on $[\underline{a}, \overline{a}]$ coincide and equal to $[f(\underline{a}), f(\overline{a})] \in \mathbb{R}$. □

Theorem 11 (Convergence of piecewise linear interpolation [DB18], Ch.5)**.** *Suppose that $f(x)$ has a continuous second derivative in $[a_k, a_{k+1}]$, that is $f \in C^2([a_k, a_{k+1}])$. Let $p_{n_{int}}(x)$ be the piecewise linear interpolant of $(a_{k_i}, f(a_{k_i}))$ for $i = 0, \ldots, n_{int}$, where*

$$a_{k_i} = a_k + ih, \quad h = \frac{a_{k+1} - a_k}{n_{int}}.$$

Then, the error bound satisfies

$$\|f - p_{n_{int}}\|_\infty = \max_{x\in[a_k,a_{k+1}]} |f(x) - p_{n_{int}}(x)| \leq Mh^2,$$

where

$$M = \max_{[a_k,a_{k+1}]} f''(x)$$

Theorem 12 (Convergence of piecewise tangent approximation)**.** *Suppose that $f \in C^2([a_k, a_{k+1}])$ and is strictly convex (or concave) in $[a_k, a_{k+1}]$. Let $\widetilde{f}^{pie\_tan}_{n_{tan}}(x)$ be the piecewise tangent approximator over subsegments $\left[a_{k_i}, a_{k_{i+1}}\right]$ for $i = 0, \ldots, n_{tan}$, where*

$$a_{k_i} = a_k + ih, \quad h = \frac{a_{k+1} - a_k}{n_{tan}}.$$

Then, the error bound satisfies

$$\|f - \widetilde{f}^{pie\_tan}_{n_{tan}}\|_\infty = \max_{x\in[a_k,a_{k+1}]} |f(x) - \widetilde{f}^{pie\_tan}_{n_{tan}}(x)| \leq Mh^2,$$

where

$$M = \max_{[a_k,a_{k+1}]} f''(x)$$

Proof. Each element of the piecewise tangent approximation is the Taylor series expansion of the first order around the boundary point of the subsegment. We consider the double Taylor series approximation on the refinement $\left[a_{k_i}, a_{k_{i+1}}\right]$ of the segment $[a_k, a_{k+1}]$ for $i = 0, \ldots, n_{int}$, where

$$a_{k_i} = a_k + ih, \quad h = \frac{a_{k+1} - a_k}{n_{tan}}.$$

Since for the twice continuously differentiable in $\left[a_{k_i}, a_{k_{i+1}}\right]$ the Lagrange Remainder of the Taylor series expansion [Rud76], which represents an error term, can be bounded with

$$\max_{\left[a_{k_i}, a^{pie\_tan}_{k'_i}\right]} \left[f(x) - \widetilde{f}^{pie\_tan}(x)\right] \leq M_{i_1} \frac{(a^{pie\_tan}_{k'_i} - a_{k_i})^2}{2} \leq M_{i_1} \frac{(a_{k_{i+1}} - a_{k_i})^2}{2},$$

$$\max_{\left[a^{pie\_tan}_{k'_i}, a_{k_{i+1}}\right]} \left[f(x) - \widetilde{f}^{pie\_tan}(x)\right] \leq M_{i_2} \frac{(a_{k_{i+1}} - a^{pie\_tan}_{k'_i})^2}{2} \leq M_{i_2} \frac{(a_{k_{i+1}} - a_{k_i})^2}{2}$$

where

$$M_{i_1} = \max_{\left[a_{k_i}, a^{pie\_tan}_{k'_{i+1}}\right]} f''(x) \leq \max_{[a_k,a_{k+1}]} f''(x) = M$$

$$M_{i_2} = \max_{\left[a^{pie\_tan}_{k'_i}, a_{k_{i+1}}\right]} f''(x) \leq \max_{[a_k,a_{k+1}]} f''(x) = M$$

The maximum M exists and is attainable due to the continuity of the second derivative on a compact $[a_k, a_{k+1}]$. That is, the maximum error on the whole segment of approximation can be bounded as

$$M\left[\frac{a_{k+1}-a_k}{n_{tan}}\right]^2 = Mh^2 \xrightarrow[n_{tan}\to\infty]{} 0$$

□

The following lemma demonstrates that the approximation procedure presented in Section 4.3.2 generates sequences of estimators that: i) serve as valid bounds for the target function, and ii) converge monotonically to the target function. This implies that each new estimator can only improve upon the previous one.

Lemma 7. *Suppose that $f \in C^2([a_k, a_{k+1}])$ is monotonic increasing and strictly convex (or concave) on $[a_k, a_{k+1}]$. Let $\widetilde{f}_n^{lin\_int}(x)$ be the piecewise linear interpolant and $\widetilde{f}_n^{pie\_tan}(x)$ be the piecewise tangent approximator over subsegments $\left[a_{k_i}, a_{k_{i+1}}\right]$ for $i = 0, \ldots, 2^n$, where*

$$a_{k_i} = a_k + ih, \quad h = \frac{a_{k+1}-a_k}{2^n},$$

with $n \in \mathbb{N}$. Then the estimating functions defined by the linear interpolation $\widetilde{f}_n^{lin\_int}$ and piecewise tangent approximation $\widetilde{f}_n^{pie\_tan}$ define upper (lower) and lower (upper), respectively, bounds on the target function f. Moreover, $\{\widetilde{f}_n^{lin\_int}(x)\}_n$ and $\{\widetilde{f}_n^{pie\_tan}(x)\}_n$ are non-increasing (non-decreasing) and non-decreasing (non-increasing) sequences, respectively.

Proof. By the definition of the convex (concave) function,

$$f(\alpha x_1 + (1-\alpha)x_2) \leq (\geq)\alpha f(x_1) + (1-\alpha)f(x_2)$$

for any $\alpha \in [0,1]$ and x_1, x_2 from the region of convexity, and plot of the linear interpolant between any x_1, x_2 lies above (below) the plot of the function. That is, linear interpolation is always an upper (lower) approximation of the convex (concave) function.

On the other hand, any tangent line lies below (above) the plot of the convex (concave) function. Indeed, since on a convex segment the derivative of the function increases, that is $f'_+(a_{k_i}) \leq f'(x) \leq f'_-(a_{k_{i+1}})$ for all $x \in \left[a_{k_i}, a_{k_{i+1}}\right]$, then

$$\begin{aligned} f(x) = f(a_{k_i}) + \int_{a_{k_i}}^{x} f'(t)dt &\geq f(a_{k_i}) + \int_{a_{k_i}}^{x} f'_+(a_{k_i})dt \\ &= f(a_{k_i}) + f'_+(a_{k_i})(x - a_{k_i}), \quad x \in \left[a_{k_i}, a_{k'_i}\right] \end{aligned}$$

$$\begin{aligned} f(x) = f(a_{k_{i+1}}) - \int_{a_{k_{i+1}}}^{x} f'(t)dt &\geq f(a_{k_{i+1}}) - \int_{a_{k_{i+1}}}^{x} f'_-(a_{k_{i+1}})dt \\ &= f(a_{k_{i+1}}) + f'_-(a_{k_{i+1}})(x - a_{k_{i+1}}), \quad x \in \left[a_{1'}, a_{k_{i+1}}\right] \end{aligned}$$

Similarly, we can show that the piecewise tangent upper bounds the true concave function. Without loss of generality, we consider the case of a convex segment. We fix n. The current element of the sequences of piecewise linear interpolants $\widetilde{f}_n^{lin\_int}(x)$ includes the local linear item based on the interval $[t_1, t_2]$. The case of a concave segment is analogous. We define the current local linear approximation of the element of sequence of upper approximation as

$$\widetilde{f}_{\{t\}_n}^{lin\_int}(x) = f(t_1) + (x - t_1)\frac{f(t_2) - f(t_1)}{t_2 - t_1}$$

Let α be such that $0 \leq \alpha \leq 1$ and define a new point t as a convex combination $t = \alpha t_1 + (1-\alpha)t_2$. Let us show that $\widetilde{f}_{\{t\}_{n+1}}^{lin\_int}(x)(x) \leq \widetilde{f}_{\{t\}_n}^{lin\_int}(x)$ for $x \in [t_1, t_2]$, where

$$\widetilde{f}_{\{t\}_{n+1}}^{lin\_int}(x) = \begin{cases} f(t_1) + (x - t_1)\frac{f(t) - f(t_1)}{t - t_1}, & x \in [t_1, t] \\ \\ f(t) + (x - t)\frac{f(t_2) - f(t)}{t_2 - t}, & x \in [t, t_2] \end{cases}$$

Taking into account the convex segment,

$$\begin{aligned} f(t_1) + (x - t_1)\frac{f(t) - f(t_1)}{t - t_1} &= f(t_1) + (x - t_1)\frac{f(\alpha t_1 + (1-\alpha)t_2) - f(t_1)}{\alpha t_1 + (1-\alpha)t_2 - t_1} \\ &\leq f(t_1) + (x - t_1)\frac{\alpha f(t_1) + (1-\alpha)f(t_2) - f(t_1)}{(1-\alpha)(t_2 - t_1)} \\ &= f(t_1) + (x - t_1)\frac{f(t_2) - f(t_1)}{t_2 - t_1} \end{aligned}$$

$$\begin{aligned} f(t) + (x - t)\frac{f(t_2) - f(t)}{t_2 - t} &= f(\alpha t_1 + (1-\alpha)t_2) \\ &\quad + (x - \alpha t_1 + (1-\alpha)t_2)\frac{f(t_2) - f(\alpha t_1 + (1-\alpha)t_2)}{t_2 - \alpha t_1 + (1-\alpha)t_2} \\ &\leq \alpha f(t_1) + (1-\alpha)f(t_2) \\ &\quad + (x - t_1)\frac{\alpha f(t_1) + (1-\alpha)f(t_2) - f(t_2)}{\alpha t_1 + (1-\alpha)t_2 - t_2} \\ &\quad + (t_1 - \alpha t_1 + (1-\alpha)t_2)\frac{\alpha f(t_1) + (1-\alpha)f(t_2) - f(t_2)}{\alpha t_1 + (1-\alpha)t_2 - t_2} \\ &= f(t_1) + (x - t_1)\frac{f(t_2) - f(t_1)}{t_2 - t_1} \end{aligned}$$

Since the refinement on each subsegment leads to the reduced next element of the sequence, the sequence is decreasing on the whole convex segment.

We next consider the piecewise tangent approximation on $[t_1, t_2]$,

$$\widetilde{f}_{\{t\}_n}^{pie\_tan}(x) = \begin{cases} f(t_1) + f'_+(t_1)(x - t_1), & x \in [t_1, t_{1'}], \\ f(a_2) + f'_-(t_2)(x - t_2), & x \in [t_{1'}, t_2], \end{cases}$$

where $t_{1'}$ is the point of intersection of the tangents. If we choose some parameter α, where $0 \leq \alpha \leq 1$, and define the corresponding intermediate point as $t_* = \alpha t_1 + (1-\alpha)t_2$, then the refined approximation is given by:

$$\widetilde{f}^{pie\_tan}_{\{t\}_{n+1}}(x) = \begin{cases} f(t_1) + f'_+(t_1)(x - t_1), & x \in [t_1, t_{1*}], \\ f(t_*) + f'(t_*)(x - t_*), & x \in [t_{1*}, t_{2*}], \\ f(t_2) + f'_-(t_2)(x - t_2), & x \in [t_{2*}, t_2], \end{cases}$$

where t_{1*} is the intersection point of the left and middle lines, and t_{2*} is the intersection point of the middle and right lines. We aim to show that $\widetilde{f}^{pie\_tan}_{\{t\}_{n+1}}(x)(x) \geq \widetilde{f}^{pie\_tan}_{\{t\}_n}(x)$ for $x \in [t_1, t_2]$.

Thus, the plot of the linear function corresponding to the middle curve lies above that of the left curve for $x > t_{1*}$ and above that of the right curve for $x < t_{2*}$ due to the monotonicity of the estimator, by Lemma 4. Consequently, the refined estimator $\underline{f}_2$ coincides with the previous estimator $\underline{f}_1$ on the left and right segments, i.e., for $x \in [t_1, t_{1*}]$ and $x \in [t_{2*}, t_2]$, while it takes higher values for $x \in [t_{1*}, t_{2*}]$. To confirm this, it remains to show that $t_{1*} \leq t_{1'} \leq t_{2*}$.

First, we note that $t_1 \leq t_{1'} \leq t_2$, by Lemma 3. This automatically leads to $t_1 \leq t_{1*} \leq t_* \leq t_{2*} \leq t_2$.

Consider the function

$$f_{\text{left}}(x) = \frac{x f'_-(x) - f(x) - C}{f'_-(x) - K},$$

defined on (t_1, t_2), where $C = t_1 f'_+(t_1) - f(t_1)$ and $K = f'_+(t_1)$. Its derivative is given by

$$f'_{\text{left}}(x) = \frac{f''(x)(f(x) + C - Kx)}{(f'(x) - K)^2}.$$

The numerator simplifies to

$$f''(x)(f(x) + C - Kx) = \underbrace{f''(x)}_{>0} \underbrace{\big[f(x) - (f(t_1) + f'_+(t_1)(x - t_1))\big]}_{>0},$$

which is positive due to the convexity of f. This implies that shifting the right boundary t_2 to the left, reaching position t_*, also shifts the intersection point $t_{1'}$ to the left, reaching t_{1*}. A similar argument holds for the right boundary.

The same reasoning applies to the concave segment. □

Lemma 8. *Let $f_n : \mathcal{A}_f \to \mathcal{A}_g$ be continuous functions, uniformly convergent to a continuous function $f : \mathcal{A}_f \to \mathcal{A}_g$ on a compact interval $\mathcal{A}_f \subset \mathbb{R}$, and let $g_n : \mathcal{A}_g \to \mathcal{A}$ be continuous functions, uniformly convergent to a continuous function $g : \mathcal{A}_g \to \mathcal{A}$ on a compact interval $\mathcal{A}_g \subset \mathbb{R}$. Then the sequence of composition functions $g_n(f_n(x))$ converges uniformly to $g(f(x))$ on $\mathcal{A}_f$ with $n \to \infty$.*

Proof. An outer limit function, g, is uniformly continuous by the Heine–Cantor theorem [Rud76], since it is continuous and defined on the compact set $\mathcal{A}_g$. That is, for any $\epsilon_1 > 0$, there exists $\epsilon_2 > 0$ such that

$$|g(y_1) - g(y_2)| < \frac{\epsilon_1}{2}, \quad \text{whenever } y_1, y_2 \in \mathcal{A}_g \text{ and } |y_1 - y_2| < \epsilon_2.$$

Since the sequence $\{f_n(x)\}$ converges uniformly to $f(x)$ on $\mathcal{A}_f$, and $\{g_n(y)\}$converges uniformly to $g(y)$ on $\mathcal{A}_g$, we can conclude that for any $\epsilon_1 > 0$ and $\epsilon_2 > 0$, there exists $N \in \mathbb{N}$ such that for all $n \geq N$, we simultaneously have

$$|g_n(y) - g(y)| < \frac{\epsilon_1}{2}, \qquad \text{for all } y \in \mathcal{A}_g,$$
$$|f_n(x) - f(x)| < \epsilon_2, \qquad \text{for all } x \in \mathcal{A}_f.$$

Since the range of possible values of $f_n(x)$ coincides with the range of values of $f(x)$, which equals $f_i(\mathcal{A}_f) = \mathcal{A}_g$, the domain of g, we obtain

$$\begin{aligned} |g_n(f_n(x)) - g(f(x))| &= |g_n(f_n(x)) - g(f_n(x)) + g(f_n(x)) - g(f(x))| \\ &\leq \underbrace{|g_n(f_n(x)) - g(f_n(x))|}_{\text{uniform convergence of the outer}} + \underbrace{|g(f_n(x)) - g(f(x))|}_{\text{uniform continuity of the outer}} \\ &< \frac{\epsilon_1}{2} + \frac{\epsilon_1}{2} = \epsilon_1 \end{aligned}$$

Thus, the uniform convergence of the composition follows.

□

Lemma 9. *Let $f_n^i : \mathcal{A}_i \to \mathbb{R}$ be continuous functions that converge uniformly to continuous functions $f^i : \mathcal{A}_i \to \mathbb{R}$ on compacts $\mathcal{A}_i \subset \mathbb{R}$, and let $\alpha_i \in \mathbb{R}$ be constants for $i = 1, \ldots, n_l$. Then a linear combination of sequences, $f_{\boldsymbol{\alpha},n} = \sum_i^{n_l} \alpha_i f_n^i$, defined on the direct product $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_{n_l} \subset \mathbb{R}^{n_l}$, converges uniformly to $f_{\boldsymbol{\alpha}} = \sum_i^{n_l} \alpha_i f^i : \mathcal{A} \to \mathbb{R}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{n_l})$; that is,*

$$\sup_{\boldsymbol{x} \in \mathcal{A}} |f_{\boldsymbol{\alpha},n}(\boldsymbol{x}) - f_{\boldsymbol{\alpha}}(\boldsymbol{x})| \xrightarrow[n \to \infty]{} 0.$$

The images of $f_{\boldsymbol{\alpha},n}(\mathcal{A})$ and $f_{\boldsymbol{\alpha}}(\mathcal{A})$ are compact.

Proof. Since $f_{\boldsymbol{\alpha},n}(\mathcal{A})$ and $f_{\boldsymbol{\alpha}}(\mathcal{A})$ are linear combinations of continuous functions, defined on compact sets, they are continuous functions. Also, since a direct product of compact sets is compact, by the continuous mapping theorem [Rud76], the images of $f_{\boldsymbol{\alpha},n}(\mathcal{A})$ and $f_{\boldsymbol{\alpha}}(\mathcal{A})$ are also compact.

If every sequence of functions $\{f_n^i(x^i)\}$ converges uniformly to the corresponding limit function $f^i(x^i)$, then for any $\epsilon > 0$, there exists an integer $N \in \mathbb{N}$ such that for all $n \geq N$, we have

$$\sup_{x^i \in \mathcal{A}_i} \left| f_n^i(x^i) - f^i(x^i) \right| < \frac{\epsilon}{\max_i\{\alpha_i\} n_l}$$

for all $i = 1, \ldots, n_l$. Consequently, the supremum of the differences in the linear combination can be bounded as

$$\begin{aligned} \sup_{\boldsymbol{x} \in \mathcal{A}} \left| f_{\boldsymbol{\alpha},n}(\boldsymbol{x}) - f_{\boldsymbol{\alpha}}(\boldsymbol{x}) \right| &= \sup_{\boldsymbol{x} \in \mathcal{A}} \left| \sum_{i=1}^{n_l} \alpha_i (f_n^i(x^i) - f^i(x^i)) \right| \\ &\leq \sum_{i=1}^{n_l} |\alpha_i| \sup_{x^i \in \mathcal{A}_i} \left| f_n^i(x^i) - f^i(x^i) \right| < \epsilon \end{aligned}$$

This establishes the uniform convergence of the linear combination of functions.

□

Lemma 10. *Let $\overline{f}^i, \underline{f}^i, f^i : \mathcal{A}_i \to \mathbb{R}$ be continuous functions on compact intervals $\mathcal{A}_i \subset \mathbb{R}$, satisfying*

$$\underline{f}^i(x^i) \leq f^i(x^i) \leq \overline{f}^i(x^i)$$

for all $x^i \in \mathcal{A}_i$ and for every $i = 1, \ldots, n_l$.

Suppose that the index sets I and J are disjoint and their union forms the full sequence:

$$I \cup J = \{1, \ldots, n_l\}.$$

Then, for any coefficients $\alpha_i, \beta_j \geq 0$ for $i \in I$, $j \in J$, the following inequality holds:

$$\sum_{i\in I} \alpha_i \underline{f}^i(x^i) - \sum_{j\in J} \beta_j \overline{f}^j(x^j) \leq \sum_{i\in I} \alpha_i f_i(x^i) - \sum_{j\in J} \beta_j f_j(x^j) \leq \sum_{i\in I} \alpha_i \overline{f}^i(x^i) - \sum_{j\in J} \beta_j \underline{f}^j(x^j),$$

for all $x^i \in \mathcal{A}_i$ and $x^j \in \mathcal{A}_j$.

Proof. For any non-negative coefficients α_i, β_j and given that $\underline{f}_i(x^i) \leq f_i(x^i) \leq \overline{f}_i(x^i)$, we derive the following inequalities:

$$\alpha_i \underline{f}^i(x^i) \leq \alpha_i f^i(x^i) \leq \alpha_i \overline{f}^i(x^i),$$
$$-\beta_j \overline{f}^j(x^j) \leq -\beta_j f^j(x^j) \leq -\beta_j \underline{f}^j(x^j).$$

Summing these inequalities over all indices completes the proof. □

Lemma 11. *Let $\overline{f}, \underline{f}, f : \mathcal{A} \to \mathcal{B}$ be continuous functions on a compact interval $\mathcal{A} \subset \mathbb{R}$, satisfying*

$$\underline{f}(x) \leq f(x) \leq \overline{f}(x) \quad \text{for all } x \in \mathcal{A}.$$

Furthermore, let $\overline{g}, \underline{g}, g : \mathcal{B} \to \mathbb{R}$ be continuous and monotonically increasing functions on a compact interval $\mathcal{B} \subset \mathbb{R}$, satisfying

$$\underline{g}(y) \leq g(y) \leq \overline{g}(y) \quad \text{for all } y \in \mathcal{B}.$$

Then, for all $x \in \mathcal{A}$, the following inequality holds:

$$\underline{g}(\underline{f}(x)) \leq g(f(x)) \leq \overline{g}(\overline{f}(x)).$$

Proof. By assumption, for any $y \in \mathcal{B}$,

$$\underline{g}(y) \leq g(y) \leq \overline{g}(y).$$

Since $\underline{g}(y), g(y)$, and $\overline{g}(y)$ are monotonically increasing, it follows that for any $y_1, y_2 \in \mathcal{B}$ such that $y_1 \leq y \leq y_2$,

$$\underline{g}(y_1) \leq \underline{g}(y) \leq g(y) \leq \overline{g}(y) \leq \overline{g}(y_2).$$

Setting $y = f(x)$, $y_1 = \underline{f}(x)$, and $y_2 = \overline{f}(x)$, and using that $\underline{f}(x) \leq f(x) \leq \overline{f}(x)$ for all $x \in \mathcal{A}$, we obtain the desired result:

$$\underline{g}(\underline{f}(x)) \leq g(f(x)) \leq \overline{g}(\overline{f}(x)).$$

□

Theorem 13 (Main Theorem: Uniform Monotonic Convergence of the Bounds)**.** *The sequences of estimating functions that are ReLU neural networks generated by the method in Section 4.3.2, establish upper and lower bounds for the target neural network. These sequences are monotonically decreasing for the upper bounds and monotonically increasing for the lower bounds, and they converge uniformly to the target network.*

Proof. We establish the uniform monotonic convergence of the bounds by considering several key properties.

First, we analyze the approximation of neuron domains. Using the concept of over-approximating the input domain of each neuron (as in IBP [GDS+18]), we define $\overline{\Omega}_i$ as a superset of the true input domain Ω_i for each neuron i. It is well known that if a sequence of approximations converges uniformly on $\overline{\Omega}_i$, it must also converge uniformly on any subset of $\overline{\Omega}_i$, including Ω_i. Therefore, we perform all neuron-wise approximations over the supersets of their original domains.

By Lemma 3, both the upper and lower bounds for each neuron in every layer are continuous functions over a bounded domain. Additionally, by Lemma 6, the images of these bounds coincide with the image of the activation function. Since the activation function and its estimators are continuous, and the domain is compact, the output range remains compact throughout the approximation process. This compactness follows from the continuity of the mapping [Rud76].

Furthermore, the error between the linear interpolation (or piecewise tangent approximation) and the original activation function converges to zero as the approximation grid is refined, as established in Theorems 11 and 12. Consequently, the sequence of bounds on the activation function converges uniformly to the target activation function for all neurons in the network.

By Lemma 6, the image of each activation function is preserved by both the upper and lower bounds, ensuring that the domain of uniform convergence for the bounds of the outer functions is also preserved. Moreover, by Lemma 9, the linear combination of these estimators converges uniformly to the corresponding linear combination of the true activation functions, thereby preserving the image of the original linear combination. As a result, Lemma 8 guarantees the uniform convergence of the composed bounds on the outer functions and the linear combinations of the inner functions.

Next, we establish monotonicity. By Lemma 7, the sequences of upper and lower bounds on the activation functions are monotonic: the upper bounds $\{\overline{f}_n(x)\}$ are monotonically decreasing, while the lower bounds $\{\underline{f}_n(x)\}$ are monotonically increasing. Additionally, by construction, these sequences are bounded by the target function itself, ensuring they remain within the correct range. By Lemma 10, the linear combination of neurons' estimators forms the overall upper and lower bounds for input arguments in subsequent layers.

Moreover, the compositions of the upper and lower bounds for the inner and outer functions provide valid upper and lower bounds for the composition of the target inner and outer activation functions. Since the outer activation function (and consequently its bounds, by Lemma 4) is non-decreasing, Lemma 11 ensures these bounds are valid. Finally, these bounds converge monotonically by Lemmas 7 and 11.

Combining all the results above, we conclude that the sequences of upper and lower ReLU bounds for the network converge uniformly and monotonically to the target network. The monotonicity of the sequences, the uniform convergence of individual approximations, and the preservation of continuity and compactness through compositions collectively ensure the uniform convergence of the entire network. This completes the proof. □

Remark 5. *Although Theorem 10 follows from the UAT, the proof of its particular case, Theorem*

13, is constructive and explicitly provides the sequences that form the bounds.

4.3.4 Application to an arbitrary function on a compact domain

We present a universal *distribution* approximation theorem, which may serve as a starting point for further research in the stochastic behavior of functions and the neural networks that describe them.

Theorem 14 (Universal distribution approximation theorem)**.** *Let* $\mathbf{X}$ *be a random vector with continuous pdf* $\phi(\mathbf{x})$ *supported over a compact hyperrectangle* $K \subset \mathbb{R}^{n_0}$*. Let* $Y = \mathcal{W}(\mathbf{x})$ *be a continuous function of* $\mathbf{X}$ *with domain* K*, and let* $F(y)$ *denote its cdf. Then, there exist sequences of cdf bounds* $\{\underline{F}_n\}$*,* $\{\overline{F}_n\}$*,* $n = 1, 2, \ldots$*, which can be constructed by bounding the distributions of sequences of ReLU NNs and such that*

$$\underline{F}_n(y) \leq F(y) \leq \overline{F}_n(y)$$

for all $y \in \{\mathcal{W}(\mathbf{x}) : \mathbf{x} \in K\}$ *and*

$$\underline{F}_n(y) \to F(y), \quad \overline{F}_n(y) \to F(y)$$

for all y *where* $F(y)$ *is continuous. Moreover, if* $\mathcal{W}(\mathbf{X})$ *is almost surely nowhere locally constant, that is*

$$\int_{\{\mathcal{W}(\mathbf{x})=y\}} \phi(\mathbf{x})d\mathbf{x} = 0 \tag{4.7}$$

for all $y \in \{\mathcal{W}(\mathbf{x}) : \mathbf{x} \in K\}$*, then both bounds* $\underline{F}_n, \overline{F}_n$ *converge uniformly to the true cdf* F*.*

Proof. By the UAT [HSW89], for any $\epsilon > 0$ there exist one-layer networks $\widetilde{Y}$, $\widetilde{\phi}$ with ReLU activation function, such that

$$\sup_{\mathbf{x}\in K} \parallel \mathcal{W}(\mathbf{x}) - \widetilde{Y}(\mathbf{x}) \parallel_{\mathbb{R}^{n_0}} < \epsilon, \qquad \sup_{\mathbf{x}\in K} \parallel \phi(\mathbf{x}) - \widetilde{\phi}(\mathbf{x}) \parallel_{\mathbb{R}^{n_0}} < \epsilon.$$

Define $\underline{Y}_n(\mathbf{x}) = \widetilde{Y}(\mathbf{x}) - \epsilon$, which is also a neural network. Similarly, for $\overline{Y}_n$, $\underline{\phi}_n$, $\overline{\phi}_n$. Then,

$$\begin{aligned}
\mathcal{W}(\mathbf{x}) - 2\epsilon \leq \underline{Y}_n(\mathbf{x}) = \widetilde{Y}(\mathbf{x}) - \epsilon \leq \mathcal{W}(\mathbf{x}) \leq \widetilde{Y}(\mathbf{x}) + \epsilon = \overline{Y}_n(\mathbf{x}) < \mathcal{W}(\mathbf{x}) + \epsilon \\
\phi(\mathbf{x}) - 2\epsilon < \underline{\phi}_n(\mathbf{x}) = \widetilde{\phi}(\mathbf{x}) - \epsilon \leq \phi(\mathbf{x}) \leq \widetilde{\phi}(\mathbf{x}) + \epsilon = \overline{\phi}_n(\mathbf{x}) < \phi(\mathbf{x}) + 2\epsilon,
\end{aligned}$$

which proves that, as $\epsilon \to 0$, $\underline{Y}_n, \overline{Y}_n \to \mathcal{W}$ and $\underline{\phi}_n, \overline{\phi}_n \to \phi$, uniformly on a compact domain while guaranteeing to be upper/lower bounds.

Let

$$\overline{F}_n(y) = \min\left[1, \int_{\{\mathbf{x}:\mathbf{x}\in K \cap \underline{Y}_n(\mathbf{x}) \leq y\}} \overline{\phi}_n(\mathbf{x})d\mathbf{x}\right], \quad \underline{F}_n(y) = \max\left[0, \int_{\{\mathbf{x}:\mathbf{x}\in K \cap \overline{Y}_n(\mathbf{x}) \leq y\}} \underline{\phi}_n(\mathbf{x})d\mathbf{x}\right]$$

The limit cdf is

$$F(y) = \int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap Y(\mathbf{x})\leq y\}} \phi(\mathbf{x})d\mathbf{x}$$

Since $\underline{\phi}_n(\mathbf{x}) \leq \phi(\mathbf{x}) \leq \overline{\phi}_n(\mathbf{x})$ and $\underline{Y}_n(\mathbf{x}) \leq \mathcal{W}(\mathbf{x}) \leq \overline{Y}_n(\mathbf{x})$ for any $\mathbf{x} \in K$, $\{\mathbf{x} : \mathbf{x} \in K \cap \underline{Y}_n(\mathbf{x}) \leq y\} \supseteq \{\mathbf{x} : \mathbf{x} \in K \cap \mathcal{W}(\mathbf{x}) \leq y\}$ and $\{\mathbf{x} : \mathbf{x} \in K \cap \overline{Y}_n(\mathbf{x}) \leq y\} \subseteq \{\mathbf{x} : \mathbf{x} \in K \cap \mathcal{W}(\mathbf{x}) \leq y\}$. Since $0 \leq F(y) \leq 1$ for all y,

$$\underline{F}_n(y) \leq F(y) \leq \overline{F}_n(y)$$

for all $y \in \{\mathcal{W}(\mathbf{x}) : \mathbf{x} \in K\}$.

Now let us fix an arbitrary $y = \mathcal{W}(\mathbf{x})$ for $\mathbf{x} \in K$, such that y is a continuity point of F.

$$\begin{aligned}
\overline{F}_n(y) - F(y) &\leq \int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\underline{Y}_n(\mathbf{x})\leq y\}} \overline{\phi}_n(\mathbf{x})dx - \int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\mathcal{W}(\mathbf{x})\leq y\}} \phi(\mathbf{x})d\mathbf{x} \\
&= \underbrace{\int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\underline{Y}_n(\mathbf{x})\leq y\}} (\overline{\phi}_n(\mathbf{x}) - \phi(\mathbf{x}))dx}_{A} \\
&+ \underbrace{\int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\underline{Y}_n(\mathbf{x})\leq y\}} \phi(\mathbf{x})dx - \int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\mathcal{W}(\mathbf{x})\leq y\}} \phi(\mathbf{x})dx}_{B}
\end{aligned}$$

$$\begin{aligned}
F(y) - \underline{F}_n(y) &\leq \int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap Y\leq y\}} \phi(\mathbf{x})dx - \int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\overline{Y}_n(x)\leq y\}} \underline{\phi}_n(\mathbf{x})d\mathbf{x} \\
&= \underbrace{\int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\overline{Y}_n(\mathbf{x})\leq y\}} (\phi(\mathbf{x}) - \underline{\phi}_n(\mathbf{x}))dx}_{C} \\
&+ \underbrace{\int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap Y\leq y\}} \phi(\mathbf{x})d\mathbf{x} - \int\limits_{\{\mathbf{x}:\mathbf{x}\in K\cap\overline{Y}_n(\mathbf{x})\leq y\}} \phi(\mathbf{x})d\mathbf{x}}_{D}
\end{aligned}$$

The left integrals in both equations (A and C) converge to zero due to the uniform convergence to zero of the integrands over the whole set K. The second differences (B and D) converge to zero, since the superset $\{\mathbf{x} : \mathbf{x} \in K \cap \underline{Y}_n(\mathbf{x}) \leq y\}$ and subset $\{\mathbf{x} : \mathbf{x} \in K \cap \overline{Y}_n(\mathbf{x}) \leq y\}$ of the limits of integrals, respectively, converge to the true limit set $\{\mathbf{x} : \mathbf{x} \in K \cap \mathcal{W}(\mathbf{x}) \leq y\}$ due to continuity.

We showed the pointwise convergence at every continuity point of the limiting cdf; that is, convergence in distribution.

Requiring (4.7) means that the limiting distribution has no point mass; i.e., it is continuous. We can then apply Polya's theorem [Rao62] that the convergence of both bounds is uniform since the sequences converge in distribution to random variables with continuous cdfs.

□

The proof leverages the UAT [HSW89] to approximate the function $\mathcal{W}(\mathbf{x})$ and input pdf $\phi(\mathbf{x})$ with ReLU NNs to arbitrary accuracy. Cdf bounds are then computed over polytope intersections, with greater NN complexity yielding more simplices for finer local affine approximations of the pdf.

To bound the cdf of a given NN with respect to a specified input pdf, we construct upper and lower bounding ReLU NNs to approximate the target NN. Next, we subdivide the resulting polytopes of the bounding ReLU NNs into simplices as much as needed to achieve the desired accuracy and locally approximate the input pdf with constant values (the simplest polynomial form) on these simplices, transforming the problem into the one described in Section 4.3.1.

4.4 Experiments

In numerical experiments, we consider the following three datasets: Diabetes [EHJ+04], Iris [Fis36] and Wine [AF92][6].

For all of the experiments, we compare our guaranteed bounds for the output cdf with cdf estimates obtained via Monte Carlo (MC) simulation (100 million samples), as well as with the "Piecewise Linear Transformation" (PLT) method of [KHM+24]. Given that the true cdf is contained within these limits, the experiments assess the tightness of our bounds compared to both MC and PLT by tallying the number of "out of bounds" instances. Essentially, achieving very tight bounds makes it challenging to stay within those limits, whereas even imprecise estimates can fall within broader bounds.

Our experimental setup is based on small pre-trained (fixed) neural networks. For the Diabetes dataset we trained a ReLU network with 3 fully connected layers with 32, 16, and 8 neurons, respectively. The univariate output has no activation. Training was performed on 70% of the data, randomly selected. The remaining 30%, consisting of 73 observations, comprise the test set. As there is no randomness in the observations, we added univariate normal noise to one randomly selected feature of every observation (different features in different observations). The standard deviation of the Gaussian noise is set to the sample standard deviation of the selected feature within the test set.

For both, Iris and Wine, we replicated the exact experimental setup from [KHM+24] using the same test sets, Gaussian Mixtures as randomness as well as the same pre-trained networks[7]. The 1 to 3 dimensional Gaussian mixtures were computed by first deleting 25% or 50% of the test dataset. Subsequently, 50 new observations are imputed using MICE [vBG11].The only difference in the experimental setup is the MC estimate, which we recomputed with a higher sample count. Here, the MC estimate does not play the role of the "ground truth" as in their experiments, but as another estimate.

We summarize our results in Table 4.1. We observe a high ratio of "out of bounds" samples for both MC and PLT compared to the number of grid points the cdf was evaluated at. This has two components: (a) In regions where the cdf is very flat, we obtain very tight bounds leading to small errors in a bucketed estimation approach easily falling outside these tight bounds; (b) due to either pure random effect in the case of MC or numerical estimation inaccuracies in case of PLT, it produces estimates outside the bounds. We note, however, that in these examples, especially as regards PLT, a "coarse grid" can cause inaccuracies in areas where the pdf is volatile despite that

[6] All three datasets are provided by the Python package `scikit-learn`

[7] GitHub page `https://github.com/URWI2/Piecewise-Linear-Transformation`, accessed Jan 25, 2025

their estimator targets the pdf directly and it should be more accurate than ours (as our methods goal is to bound the cdf instead).

Table 4.1: Comparison of our approach (guaranteed upper and lower bounds) with pointwise estimators from Monte-Carlo simulations and PLT [KHM+24]. Under column headed by n_0 are numbers of input variables, *# tests* gives the number of observations deleted, under *U/L-dist (std)* are the mean distance (standard deviation) upper and lower bounds, under $\mathrm{OOB}_{\mathrm{MC}}$ (median) and $\mathrm{OOB}_{\mathrm{PLT}}$ (median) are the average (standard deviation) of the median of number of points outside our bounds for Monte-Carlo simulations and PLT, respectively. The grid size we used was 20 raised to the power of the number of output classes, that is, 20^{n_L}.

dataset	%unc	n_0	# tests	U/L-dist (std)	$\mathrm{OOB}_{\mathrm{MC}}$ (median)	$\mathrm{OOB}_{\mathrm{PLT}}$ (median)	Grid size
diabetes	-	1	73	0.01337 (0.005)	7 (7)	18 (19)	20
iris	25	1	25	0.00988 (0.016)	1093 (1080)	1604 (2148)	8000
		2	10	0.01236 (0.021)	793 (697)	1528 (2162)	8000
		3	–	—	—	—	8000
iris	50	1	20	0.05534 (0.077)	305 (30)	500 (4)	8000
		2	23	0.04709 (0.090)	980 (1016)	1312 (2031)	8000
		3	8	0.06864 (0.108)	779 (599)	1150 (1000)	8000
wine	25	1	32	0.02014 (0.055)	1066 (1242)	1708 (2172)	8000
		2	8	0.00909 (0.017)	1188 (1347)	1839 (2275)	8000
		3	2	0.00008 (0.000)	1324 (1324)	2516 (2516)	8000
wine	50	1	27	0.07590 (0.109)	757 (273)	1027 (45)	8000
		2	21	0.04214 (0.079)	1031 (1193)	1304 (2057)	8000
		3	13	0.06612 (0.102)	550 (128)	797 (40)	8000

4.4.1 Description of the Iris Experiments

We trained a fully connected $[3 \times 12]$ ReLU NN with a final $[1 \times 3]$ linear layer, as well as a fully connected $[3 \times 12]$ $\tanh$ NN with the same final $[1 \times 3]$ linear layer, on the Iris dataset. The networks classify objects into three classes: *Setosa*, *Versicolor*, and *Virginica*, using two input features: *Sepal Length* and *Sepal Width*. The allocation of the data for these two variables in the three classes is shown in Figure 4.6a. The input data were rescaled to be within the interval $[0, 1]$.

Experiment 1: ReLU-Based Network with Random Inputs. The ReLU network was pre-trained. We next introduced randomness to the input variables by modeling them as Beta-distributed with parameters $(2, 2)$ and $(3, 2)$, respectively. The pdfs of these input distributions are shown in Figure 4.6b. The first is symmetric about 0.5 and the second is left-skewed.

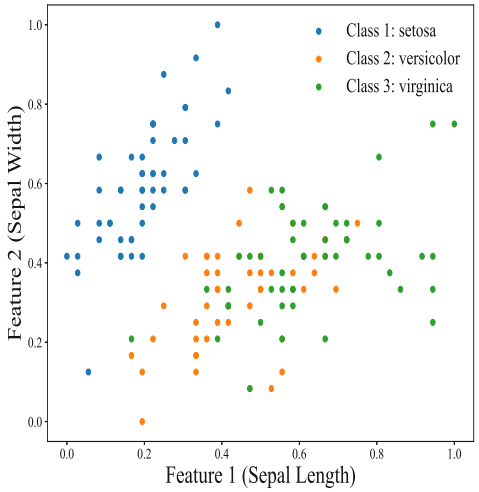
In our first experiment, Example 6, we computed the exact cdf of the first output neuron (out of three) in the ReLU network before applying the softmax function. Due to the presence of a final linear layer, the output may contain negative values. To validate our computation, we compared it against a conditional *ground truth* obtained via extensive Monte Carlo simulations, where the empirical cdf was estimated using 10^5 samples. As shown in Figure 4.1, both cdf plots coincide. The

cdf values were computed at 100 grid points across the estimated support of the output, determined via the IBP procedure.

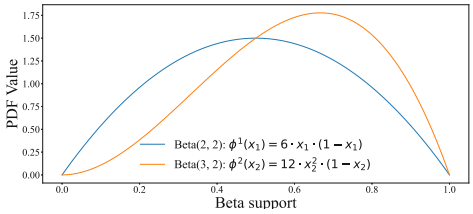
For further comparison, Figure 4.6c presents an approximation of the output pdf based on the previously computed cdf values. This is compared to a histogram constructed from Monte Carlo samples. Additionally, we plot the Gaussian kernel density estimate (KDE) of the pdf computed from the sampled data using a smoothing parameter of $h = 0.005$. The results indicate that our pdf approximation better represents the underlying distribution compared to KDE and tracks the histogram more closely.

Experiment 2: Bounding a Tanh-Based Network with ReLU Approximations. In our second experiment, we used a pre-trained NN with a similar structure but replaced the ReLU activation functions in the first three layers with *tanh*. To approximate this network, we constructed two bounding fully connected NNs—one upper and one lower—using only ReLU activations.

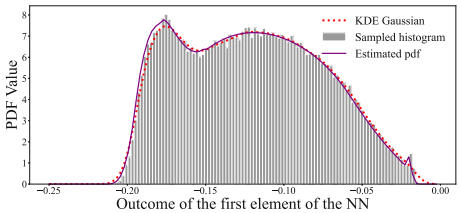
We conducted computations in two regimes: one using 5 segments and another using 10 segments, into which both convex and concave regions of the *tanh* activation function at each neuron in the first three layers were divided. Over each segment, we performed piecewise linear approximations according to the procedure described in Section 4.3.2 and combined these approximations into three-layer ReLU networks with an additional final linear layer for both upper and lower bounds. The results of these approximations are shown in Figure 4.2. In the 10-segment regime, both the upper and lower approximations closely align with the original NN's output.



(a) Plot of *Sepal Width* vs *Sepal Length* with class indication.



(b) Plots of the $Beta(2, 2)$ and $Beta(3, 2)$ pdfs, resp.



(c) Estimated output pdf compared with KDE with smoothing parameter $h = 0.005$ and histogram of MC simulations.

Figure 4.6: Iris Dataset

4.5 Related Work

The literature on NN verification is not directly related to ours as it has been devoted to standard non-stochastic input NNs, where the focus is on establishing guarantees of local robustness. This line of work develops testing algorithms for whether the output of a NN stays the same within a specified neighborhood of the deterministic input (see, e.g., [GDS+18; XSZ+20; ZWC+18; ZCX+20; SJK+24; BLT+19; FMJ+22; KBD+17; KHI+19; WIZ+24]).

To handle noisy data or aleatoric uncertainty (random input) in NNs, two main approaches have been proposed: sampling-based and probability density function (pdf) approximation-based. Sampling-based methods use Monte Carlo simulations to propagate random samples through the NN (see, e.g., [AWH+15; JRL20]), but the required replications to achieve similar accuracy to theoretical approaches such as ours, as can be seen in Table 4.1, can be massive. Pdf approximation-based methods assume specific distributions for inputs or hidden layers, such as Gaussian [AWH+15] or Gaussian Mixture Models [ZS21], but these methods often suffer from significant approximation errors and fail to accurately quantify predictive uncertainty. Comprehensive summaries and reviews of these approaches can be found in sources like [SAS+22] and [GTA+23].

In the context of verifying neural network properties within a probabilistic framework, [WCN+19] proposed PROVEN, a general probabilistic framework that ensures robustness certificates for neural networks under Gaussian and Sub-Gaussian input perturbations with bounded support with a given probability. It employs CROWN [ZWC+18; ZCX+20] to compute deterministic affine bounds and subsequently leverages straightforward probabilistic techniques based on Hoeffding's inequality [Hoe63]. PROVEN provides a probabilistically sound solution to ensuring the output of a NN is the same for small input perturbations with a given probability, its effectiveness hinges on the activation functions used. It cannot refine bounds or handle various input distributions, which may limit its ability to capture all adversarial attacks or perturbations in practical scenarios.

The most relevant published work to ours we could find in the literature is [KHM+24]. They propagate input densities through NNs with piecewise linear activations like ReLU, without needing sampling or specific assumptions beyond bounded support. Their method calculates the propagated pdf in the output space using the ReLU structure. They estimate the output pdf, which is shown to be very close to a pdf estimate obtained by Monte Carlo simulations. Despite its originality, the approach has drawbacks, as they compare histograms rather than the actual pdfs in their experiments. Theorem 5 (App. C) in [KHM+24] suggests approximating the distribution with fine bin grids and input subdivisions, but this is practically difficult. Without knowledge of the actual distribution, it is challenging to define a sufficiently "fine" grid. In contrast, we compute exact bounds of the true output cdf over its entire support (at any point, no grid required), representing the maximum error over its support, and show convergence to the true cdf. [KHM+24] use a piecewise constant approximation for input pdfs, which they motivate by their Lemma 3 (App. C) to deduce that exact propagation of piecewise polynomials through a neural network cannot be done. We demonstrate that it is feasible and provide a method for exact integration over polytopes. Additionally, their approach is limited to networks with piecewise linear activations, excluding locally nonlinear functions. In contrast, our method adapts to CNNs and any NN with continuous, monotonic piecewise twice continuously differentiable activations.

4.6 Conclusion

We develop a novel method to analyze the probabilistic behavior of the output of a neural network subject to noisy (stochastic) inputs. We formulate an algorithm to compute bounds (upper and lower) for the cdf of a neural network's output and prove that the bounds are guaranteed and that they converge uniformly to the true cdf.

Our approach enhances deterministic local robustness verification using non-random function approximation. By bounding intermediate neurons with piecewise affine transformations and known ranges of activation functions evaluated with IBP [GDS+18], we achieve more precise functional bounds. These bounds converge to the true functions of input variables as local linear units increase.

Our method addresses neural networks with continuous monotonic piecewise twice continuously differentiable activation functions using tools like Marabou [WIZ+24], originally designed for piecewise linear functions. While the current approach analyzes the behavior of NNs on a compact hyperrectangle, we can easily extend our theory to unions of bounded polytopes. In future research, we plan to bound the cdf of a neural network where the input admits arbitrary distributions with bounded piecewise continuous pdf supported on arbitrary compact sets. Moreover, we intend to improve the algorithmic performance so that our method applies to larger networks.

5 Summary and Future Perspectives

We have presented two complementary methodologies for analyzing probabilistic loops with non-polynomial updates: one for computing state variable moments in closed form and another for recovering unknown probability density functions from a finite number of moments. These methods contribute to the broader goal of advancing automated probabilistic analysis in diverse domains, including stochastic dynamical systems, probabilistic programming, and machine learning.

Our first approach provides both an exact and an approximate method for the computation of the moments of random state variables in probabilistic loops. The exact method applies to loops with trigonometric and exponential assignments under independent random perturbations across iterations. The approximate method leverages polynomial chaos expansion (PCE) to represent non-polynomial updates as orthogonal polynomial series, ensuring optimal exponential convergence when function parameters exhibit stable distributions. Moreover, we establish the L_1 convergence of the PCE estimator to the true function in the presence of unstable parameter distributions. These methods enable moment-based characterization of stochastic models across multiple domains and significantly reduce computational time compared to existing tools. Future work in this direction will focus on leveraging these moment-based representations to analyze stability properties, such as Lyapunov and asymptotic stability, in stochastic dynamical systems.

Our second approach, the K-series estimation method, recovers a probability density function from a finite set of its moments. By selecting a reference density function, K-series generalizes existing series-based density estimation techniques, such as Gram-Charlier expansions, and ensures convergence to the true density in L_1 norm. The method also includes an algorithm to estimate the minimal support of the target distribution, a crucial feature for practical applications where distributions have bounded support. The choice of the reference density affects estimation accuracy, with the truncated normal distribution typically yielding the best results. Future work will focus on extending K-series to recover probability mass functions for discrete random variables, deriving explicit error bounds, and integrating Fourier series representations to enhance estimation accuracy for finite loop iterations. Additionally, we aim to develop an automated tool that implements the full K-series methodology.

Together, these approaches provide a robust framework for probabilistic analysis, offering efficient and accurate solutions for moment computation and density estimation. By extending these methods to broader classes of probabilistic models, we seek to further bridge the gap between formal verification, uncertainty quantification, and real-world applications in probabilistic programming, control systems, and machine learning.

We plan to develop a fully automated tool that streamlines the entire process of moment computation and density recovery. This tool will serve as an add-on to Polar [MSB+22], an existing framework for computing moment-based invariants. By integrating our methodologies within an automated pipeline, we aim to provide a seamless and efficient solution for probabilistic loop analysis. This extension will enable users to compute moments in closed form and reconstruct probability density functions with minimal manual intervention, significantly enhancing accessibility and us-

ability.

We have also developed a novel framework for analyzing the probabilistic behavior of neural networks subjected to stochastic (noisy) inputs. Our approach introduces an algorithm to compute upper and lower bounds for the cumulative distribution function of a neural network's output and rigorously proves that these bounds are both guaranteed and converge uniformly to the true cdf. This method provides a fundamental step toward understanding uncertainty propagation in neural networks, offering formal probabilistic guarantees beyond traditional deterministic robustness verification.

A key contribution of our work is enhancing deterministic local robustness analysis by leveraging non-random function approximation. By bounding intermediate neurons through piecewise affine transformations and utilizing known activation function ranges evaluated with interval bound propagation (IBP) [GDS+18], we establish precise functional bounds. These bounds improve in accuracy as the number of local linear units increases, leading to tighter approximations of the network's behavior.

Our method applies to neural networks with continuous, monotonic, and piecewise twice continuously differentiable activation functions. We extend existing verification tools such as Marabou [WIZ+24], originally designed for piecewise linear functions, to analyze broader classes of networks. While the current framework evaluates neural network behavior within a compact hyperrectangle, our theoretical foundations allow for a natural extension to unions of bounded polytopes, providing greater flexibility in handling more complex input spaces.

Looking ahead, we aim to extend our methodology to input data with arbitrary bounded piecewise continuous probability density functions supported on general compact sets. This generalization will enable a more comprehensive analysis of neural networks operating under realistic, data-driven uncertainty models. Additionally, we plan to enhance the computational efficiency of our algorithm, ensuring scalability to larger and deeper networks while maintaining rigorous probabilistic guarantees. These advancements will narrow the gap between formal verification and real-world neural network applications, ensuring safer and more reliable deployments in high-risk areas such as autonomous systems and medical AI.

References

[AF92] S. Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: 10.24432/C5PC7J.

[AO01] A. Atkeson and L. E. Ohanian. Are Phillips curves useful for forecasting inflation? *Quarterly Review*, 25(Win):2–11, 2001.

[AWH+15] A. Abdelaziz, S. Watanabe, J. Hershey, E. Vincent, and D. Kolossa. Uncertainty propagation through deep neural networks. English. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2015-January:3561–3565, 2015. Publisher Copyright: Copyright © 2015 ISCA.; 16th Annual Conference of the International Speech Communication Association, INTERSPEECH 2015 ; Conference date: 06-09-2015 Through 10-09-2015.

[BAG18] A. Bibi, M. Alfadly, and B. Ghanem. Analytic expressions for probabilistic moments of pl-dnn with gaussian input. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9099–9107, 2018. DOI: 10.1109/CVPR.2018.00948.

[Ber23] A. Berzins. Polyhedral complex extraction from ReLU networks using edge subdivision. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2234–2244. PMLR, 2023.

[BGP+16] O. Bouissou, E. Goubault, S. Putot, A. Chakarov, and S. Sankaranarayanan. Uncertainty propagation using probabilistic affine forms and concentration of measure inequalities. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 225–243. Springer, 2016.

[Bil12] P. Billingsley. *Probability and Measure*. Wiley, anniversary edition edition, 2012.

[BKS19] E. Bartocci, L. Kovács, and M. Stankovič. Automatic generation of moment-based invariants for prob-solvable loops. In Y.-F. Chen, C.-H. Cheng, and J. Esparza, editors, *Automated Technology for Verification and Analysis*, pages 255–276, Cham. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-31784-3_15.

[BKS20a] G. Barthe, J.-P. Katoen, and A. Silva. *Foundations of Probabilistic Programming*. Cambridge University Press, 2020.

[BKS20b] E. Bartocci, L. Kovács, and M. Stankovič. Analysis of bayesian networks via prob-solvable loops. In V. K. I. Pun, V. Stolz, and A. Simao, editors, *Theoretical Aspects of Computing – ICTAC 2020*, pages 221–241, Cham. Springer International Publishing, 2020. DOI: 10.1007/978-3-030-64276-1_12.

[BKS21] E. Bartocci, L. Kovacs, and M. Stankovic. Mora – automatic generation of moment-based invariants, 2021. arXiv: 2103.03908 [cs.FL].

[BLT+19] R. Bunel, J. Lu, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar. Branch and bound for piecewise linear neural network verification. *ArXiv*, abs/1909.06588, 2019.

[CB01] G. Casella and R. L. Berger. *Statistical Inference*. Cengage Learning, 2nd edition, 2001.

[Chi78] T. S. Chihara. *An Introduction to Orthogonal Polynomials*. Gordon and Breach, Science Publishers, 1978.

[Cra57] H. Cramér. *Mathematical Methods of Statistics*. eng. Princeton Univ. Press, Princeton, NJ, 1957.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. DOI: 10.1007/BF02551274.

[DB18] T. A. Driscoll and R. J. Braun. *Fundamentals of numerical computation*. eng. Society for Industrial and Applied Mathematics, Philadelphia, 2018.

[Del34] B. Delaunay. Sur la sphère vide. French. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 1934(6):793–800, 1934.

[Dur19] R. Durrett. *Probability: Theory and Examples*. Cambridge University Press, 2019. DOI: 10.1017/9781108591034.

[Dwo06] C. Dwork. Differential privacy. In *Proc. of ICALP 2006: the 33rd International Colloquium on Automata, Languages and Programming*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006. DOI: 10.1007/11787006.

[EHJ+04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*:407–451, 2004.

[EMS+12] Ernst, Oliver G., Mugler, Antje, Starkloff, Hans-Jörg, and Ullmann, Elisabeth. On the convergence of generalized polynomial chaos expansions. *ESAIM: M2AN*, 46(2):317–339, 2012. DOI: 10.1051/m2an/2011045.

[FDG+19] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Scenic: a language for scenario specification and scene generation. In *Proc. of PLDI 2019: the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 63–78. ACM, 2019. DOI: 10.1145/3314221.

[FFF18] A. Fawzi, H. Fawzi, and O. Fawzi. Adversarial vulnerability for any classifier. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 1186–1195, Montréal, Canada. Curran Associates Inc., 2018.

[Fis36] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7:179–188, 1936.

[FMJ+22] C. Ferrari, M. N. Mueller, N. Jovanović, and M. Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations*, 2022.

[FMS13] D. Filipović, E. Mayerhofer, and P. Schneider. Density approximations for multivariate affine jump-diffusion processes. *Journal of Econometrics*, 176(2):93–111, 2013. DOI: 10.1016/j.jeconom.2012.12.003.

[GDS+18] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelović, T. A. Mann, and P. Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *ArXiv*, abs/1810.12715, 2018.

[Gha15] Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015. DOI: 10.1038/nature14541.

[GMD+18] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai2: safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2018. DOI: 10.1109/SP.2018.00058.

[GMV16] T. Gehr, S. Misailovic, and M. T. Vechev. PSI: exact symbolic inference for probabilistic programs. In *Proc. of CAV 2016: the 28th International Conference on Computer Aided Verification*, volume 9779 of *LNCS*, pages 62–83. Springer, 2016. DOI: 10.1007/978-3-319-41528-4_4.

[GPM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[GSS14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

[GSV20] T. Gehr, S. Steffen, and M. Vechev. λPSI: Exact Inference for Higher-Order Probabilistic Programs. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2020. DOI: 10.1145/3385412.3386006.

[GTA+23] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56:1513–1589, 2023. DOI: 10.1007/s10462-023-10562-9.

[Hal00] A. Hald. The Early History of the Cumulants and the Gram-Charlier Series. *International Statistical Review*, 68(2):137–153, 2000.

[HB20] A. M. Hafiz and G. M. Bhat. A survey of deep learning techniques for medical diagnosis. In M. Tuba, S. Akashe, and A. Joshi, editors, *Information and Communication Technology for Sustainable Development*, pages 161–170, Singapore. Springer Singapore, 2020.

[Her90] T. Herman. Probabilistic self-stabilization. *Inf. Process. Lett.*, 35(2):63–67, 1990. DOI: 10.1016/0020-0190(90)90107-9.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. DOI: 10.1016/0893-6080(89)90020-8.

[HW21] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021. DOI: 10.1007/s10994-021-05946-3.

[HWC13] M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric Statistical Methods*. John Wiley & Sons, New York-London-Sydney, 3rd edition, 2013.

[HXP17] H. Hosseini, B. Xiao, and R. Poovendran. Google's cloud vision api is not robust to noise. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 101–105, 2017. DOI: 10.1109/ICMLA.2017.0-172.

[IHR18] G. C. P. Innocentini, A. Hodgkinson, and O. Radulescu. Time dependent stochastic mrna and protein synthesis in piecewise-deterministic models of gene networks. *Frontiers in Physics*, 6, 2018. DOI: 10.3389/fphy.2018.00046.

[JF10] A. M. Z. Jasour and M. Farrokhi. Fuzzy improved adaptive neuro-nmpc for online path tracking and obstacle avoidance of redundant robotic manipulators. *Int. J. Autom. Control.*, 4(2):177–200, 2010. DOI: 10.1504/IJAAC.2010.030810.

[JF14] A. M. Jasour and M. Farrokhi. Adaptive neuro-predictive control for redundant robot manipulators in presence of static and dynamic obstacles: a lyapunov-based approach. *International Journal of Adaptive Control and Signal Processing*, 28(3-5):386–411, 2014. DOI: 10.1002/acs.2459.

[JRL20] W. Ji, Z. Ren, and C. K. Law. Uncertainty propagation in deep neural network using active subspace, 2020. arXiv: 1903.03989 [stat.ML].

[JWW21] A. Jasour, A. Wang, and B. C. Williams. Moment-based exact uncertainty propagation through nonlinear stochastic autonomous systems, 2021. arXiv: 2101.12490.

[KBB25] A. Kofnov, E. Bartocci, and E. Bura. Moment-based density elicitation with applications in probabilistic loops. *Accepted for publication in *ACM Trans. Probab. Mach. Learn.**, 2025.

[KBC+23] L. Klinkenberg, C. Blumenthal, M. Chen, and J.-P. Katoen. Exact bayesian inference for loopy probabilistic programs, 2023. arXiv: 2307.07314 [cs.PL].

[KBD+17] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: an efficient smt solver for verifying deep neural networks. In R. Majumdar and V. Kunčak, editors, *Computer Aided Verification*, pages 97–117, Cham. Springer International Publishing, 2017.

[KF76] A. Kolmogorov and S. Fomin. *Elements of the Theory of Functions and Functional Analysis*. Nauka, Moscow, 4th edition, 1976.

[KHI+19] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. L. Dill, M. J. Kochenderfer, and C. Barrett. The marabou framework for verification and analysis of deep neural networks. In I. Dillig and S. Tasiran, editors, *Computer Aided Verification*, pages 443–452, Cham. Springer International Publishing, 2019.

[KHM+24] T. Krapf, M. Hagn, P. Miethaner, A. Schiller, L. Luttner, and B. Heinrich. Piecewise linear transformation – propagating aleatoric uncertainty in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):20456–20464, 2024. DOI: 10.1609/aaai.v38i18.30029.

[KKB+25] A. Kofnov, D. Kapla, E. Bartocci, and E. Bura. Exact upper and lower bounds for the output distribution of neural networks with random inputs. 2025. DOI: 10.48550/arXiv.2502.11672.

[KMS+22a] A. Karimi, M. Moosbrugger, M. Stanković, L. Kovács, E. Bartocci, and E. Bura. Distribution estimation for probabilistic loops. In E. Ábrahám and M. Paolieri, editors, *Quantitative Evaluation of Systems*, pages 26–42, Cham. Springer International Publishing, 2022.

[KMS+22b] A. Kofnov, M. Moosbrugger, M. Stanković, E. Bartocci, and E. Bura. Moment-based invariants for probabilistic loops with non-polynomial assignments. In E. Ábrahám and M. Paolieri, editors, *Quantitative Evaluation of Systems*, pages 3–25, Cham. Springer International Publishing, 2022. DOI: 10.1007/978-3-031-16336-4_1.

[KMS+24] A. Kofnov, M. Moosbrugger, M. Stanković, E. Bartocci, and E. Bura. Exact and approximate moment derivation for probabilistic loops with non-polynomial assignments. *ACM Trans. Model. Comput. Simul.*, 2024. DOI: 10.1145/3641545.

[KS77] M. Kendall and A. Stuart. *The Advanced Theory of Statistics. Volume 1: Distribution Theory*. Macmillan, New York, NY, 1977.

[KUH19] S. Kura, N. Urabe, and I. Hasuo. Tail probabilities for randomized program runtimes via martingales for higher moments. In T. Vojnar and L. Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 135–153, Cham. Springer International Publishing, 2019.

[Las21] J. B. Lasserre. Simple formula for integration of polynomials on a simplex. *BIT*, 61(2):523–533, 2021. DOI: 10.1007/s10543-020-00828-x.

[MB03] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *Int. J. Pure Appl. Math*, 2003.

[MMR17] J. Munkhammar, L. Mattsson, and J. Rydén. Polynomial probability distribution estimation using the method of moments. *PLOS ONE*, 12(4):1–14, 2017. DOI: 10.1371/journal.pone.0174573.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. DOI: 10.1017/CBO9780511814075.

[MSB+22] M. Moosbrugger, M. Stanković, E. Bartocci, and L. Kovács. This is the moment for probabilistic loops. *Proc. ACM Program. Lang.*, 6(OOPSLA2), 2022. DOI: 10.1145/3563341.

[PHC+22] È. Pairet, J. D. Hernández, M. Carreras, Y. R. Petillot, and M. Lahijanian. Online mapping and motion planning under uncertainty for safe navigation in unknown environments. *IEEE Trans Autom. Sci. Eng.*, 19(4):3356–3378, 2022. DOI: 10.1109/TASE.2021.3118737.

[Rah18] S. Rahman. A polynomial chaos expansion in dependent random variables. *Journal of Mathematical Analysis and Applications*, 464(1):749–775, 2018. DOI: 10.1016/j.jmaa.2018.04.032.

[Raj95] P. R. Rajeev Motwani. *Randomized Algorithms*. Cambridge University Press, 1995.

[Rao62] R. R. Rao. Relations between Weak and Uniform Convergence of Measures with Applications. *The Annals of Mathematical Statistics*, 33(2):659–680, 1962. DOI: 10.1214/aoms/1177704588.

[RBI+24] F. Randone, L. Bortolussi, E. Incerto, and M. Tribastone. Inference of probabilistic programs with moment-matching gaussian mixtures. *Proc. ACM Program. Lang.*, 8(POPL), 2024. DOI: 10.1145/3632905.

[RPK+17] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, 2017.

[RS22] M. Rizzo and G. Szekely. *energy: E-Statistics: Multivariate Inference via the Energy of Data*. R package version 1.7-11. 2022.

[Rud76] W. Rudin. *Principles of mathematical analysis*. English. McGraw-Hill New York, 3d ed. Edition, 1976, x, 342 p.

[Rud86] W. Rudin. *Real and Complex Analysis*. McGraw-Hill Science/Engineering/Math, 1986.

[San20] S. Sankaranarayanan. Quantitative analysis of programs with probabilities and concentration of measure inequalities. *Foundations of Probabilistic Programming*:259, 2020.

[SAS+22] J. Sicking, M. Akila, J. D. Schneider, F. Huger, P. Schlicht, T. Wirtz, and S. Wrobel. Tailored uncertainty estimation for deep learning systems. *ArXiv*, abs/2204.13963, 2022.

[SCG+20] S. Sankaranarayanan, Y. Chou, E. Goubault, and S. Putot. Reasoning about uncertainties in discrete-time dynamical systems using polynomial forms. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17502–17513. Curran Associates, Inc., 2020.

[SD20] J. Son and Y. Du. Probabilistic surrogate models for uncertainty analysis: dimension reduction-based polynomial chaos expansion. *International Journal for Numerical Methods in Engineering*, 121(6):1198–1217, 2020. DOI: 10.1002/nme.6262.

[SJK+24] Z. Shi, Q. Jin, Z. Kolter, S. Jana, C.-J. Hsieh, and H. Zhang. Neural network verification with branch-and-bound for general nonlinearities. *ArXiv*, abs/2405.21063, 2024.

[SKS+20] A. Sudjianto, W. Knauth, R. Singh, Z. Yang, and A. Zhang. Unwrapping the black box of deep relu networks: interpretability, diagnostics, and simplification. *ArXiv*, abs/2011.04041, 2020.

[SR04] G. J. Székely and M. L. Rizzo. Testing for equal distributions in high dimension. In volume 5 of *InterStat*, 2004.

[ST12] J. Steinhardt and R. Tedrake. Finite-time regional verification of stochastic non-linear systems. *The International Journal of Robotics Research*, 31(7):901–923, 2012.

[Sza15] P. J. Szabłowski. A few remarks on orthogonal polynomials. *Applied Mathematics and Computation*, 252:215–228, 2015. DOI: 10.1016/j.amc.2014.11.112.

[Sze39] G. Szegő. *Orthogonal Polynomials*. American Mathematical Society, 1939.

[Tay93] J. B. Taylor. Discretion versus policy rules in practice. *Carnegie-Rochester Conference Series on Public Policy*, 39(1):195–214, 1993.

[UO30] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, 5, 1930. DOI: 10.1103/PhysRev.36.823.

[Vas77] O. Vasicek. An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2):177–188, 1977. DOI: 10.1016/0304-405X(77)90016-2.

[vBG11] S. van Buuren and K. Groothuis-Oudshoorn. Mice: multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011. DOI: 10.18637/jss.v045.i03.

[vdBAG11] J. van den Berg, P. Abbeel, and K. Y. Goldberg. LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. *Int. J. Robotics Res.*, 30:895–913, 7, 2011. DOI: 10.1177/0278364911406562.

[WAP+22] Z. Wang, A. Albarghouthi, G. Prakriya, and S. Jha. Interval universal approximation for neural networks. *Proc. ACM Program. Lang.*, 6(POPL), 2022. DOI: 10.1145/3498675.

[WCN+19] L. Weng, P.-Y. Chen, L. Nguyen, M. Squillante, A. Boopathy, I. Oseledets, and L. Daniel. PROVEN: verifying robustness of neural networks with a probabilistic approach. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6727–6736. PMLR, 2019.

[WIZ+24] H. Wu, O. Isac, A. Zeljić, T. Tagomori, M. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, P. Huang, O. Lahav, M. Wu, M. Zhang, E. Komendantskaya, G. Katz, and C. Barrett. Marabou 2.0: a versatile formal analyzer of neural networks. In A. Gurfinkel and V. Ganesh, editors, *Computer Aided Verification*, pages 249–264, Cham. Springer Nature Switzerland, 2024.

[Xiu10] D. Xiu. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010.

[XK02] D. Xiu and G. Karniadakis. The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(2):619–644, 2002. DOI: 10.1137/S1064827501387826.

[XSZ+20] K. Xu, Z. Shi, H. Zhang, M. Huang, K.-W. Chang, B. Kailkhura, X. Lin, and C.-J. Hsieh. Automatic perturbation analysis on general computational graphs. *ArXiv*, abs/2002.12920, 2020.

[YLC+20] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020. DOI: 10.1109/ACCESS.2020.2983149.

[ZCX+20] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C.-J. Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

[Zie95] G. M. Ziegler. *Lectures on polytopes*. Springer-Verlag, New York, 1995.

[ZS21] B. Zhang and Y. C. Shin. An adaptive gaussian mixture method for nonlinear uncertainty propagation in neural networks. *Neurocomputing*, 458:170–183, 2021. DOI: 10.1016/j.neucom.2021.06.007.

[ZWC+18] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4944–4953, Montréal, Canada. Curran Associates Inc., 2018.

Curriculum Vitae

Personal data

Name	**Andrey Kofnov**
Date of birth	[redacted]
Birth place	Krasnogorsk, Moscow Region, Russia
Nationality	Russia
Email	[redacted]

Education

2021 - 2025	Doctoral Studies in Technical Mathematics, *Vienna University of Technology (TU Wien)*, Vienna, Austria
2015 - 2017	Master's Degree in Applied Economics, *Higher School of Economics (HSE)*, Moscow, Russia
2013 - 2018	Bachelor's Degree in Economics and Finance, *University of London*, London, the United Kingdom
2011 - 2015	Bachelor's Degree in Applied Mathematics and Computer Science, *Lomonosov Moscow State University (MSU)*, Moscow, Russia

Work Experience

Apr 2021 - May 2025	University Assistant, *TU Wien*, Vienna, Austria
Mar 2020 - Apr 2021	Lead Data Scientist, *Sberbank*, Moscow, Russia
Jan 2019 - Feb 2020	Chief Validation Specialist, *DOM.RF*, Moscow, Russia
Oct 2017 - Jan 2019	Economist, *Bank of Russia*, Moscow, Russia
Jun 2016 - Oct 2017	Econometrician, *HAVAS Media Group*, Moscow, Russia

Publications

A. Kofnov et al. Moment-based invariants for probabilistic loops with non-polynomial assignments. In E. Ábrahám and M. Paolieri, editors, *Quantitative Evaluation of Systems*, pages 3–25, Cham. Springer International Publishing, 2022. DOI: 10.1007/978-3-031-16336-4_1

A. Kofnov et al. Exact and approximate moment derivation for probabilistic loops with non-polynomial assignments. *ACM Trans. Model. Comput. Simul.*, 2024. DOI: 10.1145/3641545

A. Kofnov, E. Bartocci, and E. Bura. Moment-based density elicitation with applications in probabilistic loops. *Accepted for publication in *ACM Trans. Probab. Mach. Learn.**, 2025