2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Stochastic Reachability of Uncontrolled Systems via Probability Measures: Approximation via Deep Neural Networks

Karthik Sivaramakrishnan, Vignesh Sivaramakrishnan, Rosalyn A. Devonport, Meeko M. K. Oishi

***Abstract*— This paper poses a theoretical characterization of the stochastic reachability problem in terms of probability measures, capturing the probability measure of the state of the system that satisfies the reachability specification for all probabilities over a finite horizon. We achieve this by constructing the level sets of the probability measure for all probability values and, since our approach is only for autonomous systems, we can determine the level sets via forward simulations of the system from a point in the state space at some time step in the finite horizon to estimate the reach probability. We devise a training procedure which exploits this forward simulation and employ it to design a deep neural network (DNN) to predict the reach probability provided the current state and time step. We validate the effectiveness of our approach through three examples.**

I. Introduction

Stochastic reachability is an important tool to provide probabilistic assurances of safety in stochastic, dynamical systems, by ensuring that constraints on the state space are met with at least a desired likelihood, despite bounded control authority. However, computing stochastic reachable sets is a difficult challenge, because the solution to the stochastic reachability problem is based in dynamic programming [1]–[3]. While some progress has been made by exploiting linearity in the dynamics and structure in the stochasticity [4]–[7], computational solutions remain challenging, particularly for systems described by general nonlinear dynamics or with arbitrary stochastic processes. In such cases, methods based in approximate dynamic programming have been proposed [8], [9], that learn the value function at each time step, which can be expensive for longer horizons.

The main contribution of this paper is a formulation of the stochastic reachability problem via probability measures, that captures the state distribution which satisfies the reachability specification at every time step. We synthesize a probability measure that ensures compliance with reachability specifications for all probabilities. We propose a measure-based approach to approximately computing reachable sets using neural nets as surrogate functions. The probability measure of a random variable returns, for a set input (e.g., a set of states), the likelihood of the variable lying in that set. With a probability measure, we can find a set associated with any prescribed likelihood [10], [11]. Occupation measures, a type of probability measure, have been used to compute regions of attraction [12] and control invariant sets [13]. Essentially, we use measures to represent the stochastic nature of the evolution of the system state. We additionally seek numerical implementations that can approximate stochastic reachable sets by exploiting this formalism. We capture the level sets for some likelihood as a root-finding problem over possible probability measures, then propose a data-driven approach based in empirical assessments of constraint satisfaction. This approach reduces problem of constructing a state probability measure to one of forward propagation of the dynamics, to empirically evaluate probabilities. We construct these measures using a backwards recursion formalism; this formalism yields a sequence of value functions that we estimate from forwards trajectory data using a neural network surrogate function. The value functions themselves are not novel– indeed, they are central to the dynamic programming approach of [1]– but they are intensive to compute in general. The method by which we approximate them from data, and to use them to generalize the observed evolutions into an approximate reachable set over the continuum of the state space, is another of our contributions. This approach to approximate reachability has proven successful in other domains where dynamic programming is employed, e.g. in [14], but has not yet been applied to the case of stochastic reachability. We focus on *autonomous systems*, that is, dynamical systems with no control input.

Our deep neural net learns the probability measure across all safety likelihoods simultaneously, but has computational complexity that scales with the network size (as determined by the number of neurons and layers).

In the following, Section II presents the preliminaries and

This material is based upon work supported by the National Science Foundation under NSF Grant Number CNS-1836900. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The NASA University Leadership initiative (Grant 80NSSC20M0163) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity. This material is based upon research supported, in collaboration with Verus Research, by the Air Force Research Lab (AFRL) under agreement number FA9453-23-C-A025. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and or the U.S. Government.

Karthik Sivaramakrishnan is a graduate student with Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM.
Email: ksivaramakrishnan20@unm.edu.

Vignesh Sivaramakrishnan is a graduate student with Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM.
Email: vigsiv@unm.edu.

Rosalyn Alex Devonport is a postdoctoral scholar with Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM.
Email: devonport@unm.edu.

Meeko M. K. Oishi is a professor with Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM.
Email: oishi@unm.edu.

the problem formulation. In Section III, We characterize the stochastic reachability problem via probability measures. Section IV describes the training procedure to create a deep neural net that predicts the stochastic reachability probability for a finite time horizon, for a given state and time step. Finally, in Section V, we apply our approach to multiple scenarios, including a satellite pointing problem.

II. Preliminaries and Problem Formulation

We presume Borel space, $(\mathcal{X}, \mathscr{B}(\mathcal{X}))$ where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathscr{B}(\mathcal{X})$ is a Borel sigma algebra, i.e. a non-empty collection of subsets $X \in \mathscr{B}(\mathcal{X})$ [10]. A probability measure, $\mu : \mathscr{B}(\mathcal{X}) \to [0, 1]$, maps from the Borel sigma algebra outputs a probability value between zero and one. Random variables are in boldface, $\mathbf{x}$, where we specify the probability measure we define them over as $\mathbf{x} \sim \mu$.

Definition 1 ([10, Definition 7.12])**.** *A discrete-time, autonomous stochastic system is a Borel measurable transition kernel, $Q : \mathscr{B}(\mathcal{X}) \times \mathcal{X} \to [0, 1]$, that takes a set from the Borel sigma algebra and outputs a value between zero and one. We denote this function as*

$$Q(\cdot|x). \tag{1}$$

We denote whether a state resides within a set, $T \in \mathscr{B}(\mathcal{X})$, by an indicator function,

$$1_T(x) = \begin{cases} 1, & x \in T \\ 0, & x \notin T. \end{cases} \tag{2}$$

The probability of the current state residing within a set, provided we know the probability measure of the current state, $\mu_k(\cdot)$, is described by the integral

$$\mu_k(T_k) = \int_{\mathcal{X}} 1_{T_k}(x_k)\mu_k(\mathrm{d}x_k). \tag{3}$$

In addition, we can relate the probability measure of the next state for all sets in the Borel sigma algebra to the probability measure of the current state by

$$\mu_{k+1}(X) = \int_{\mathcal{X}} Q(X|x_k)\mu_k(\mathrm{d}x_k),\ \forall X \in \mathscr{B}(\mathcal{X}) \tag{4}$$

We note that μ_{k+1} is unique [15, Lemma 10.4.3].

Problem 1. *Given a target tube, T_k for $k = \{0, \dots, N\}$, find the state probability measures $\mu_k(\cdot)$ for a finite horizon N such that the system in Definition 1 resides in the target tube for all probability values $\alpha \in [0, 1]$.*

Problem 1 can be adapted to different types of stochastic reachability specifications including reach-avoid [8], terminal-hitting time [1], and, as it is now, for target-tube [6]. This is accomplished by writing the reachability specifications using the appropriate indicator functions in (2).

We address Problem 1 by providing a theoretical characterization of the solution to Problem 1 in Section III, and a numerical implementation based in deep neural nets (DNNs) in Section IV.

III. The Measure-Theoretic Characterization of Stochastic Reachability

We now present the measure-theoretic formulation of the stochastic reachability problem that will form the basis for our neural network-based reachable set approximation method. First, we define the stochastic reachability specifications in terms of probability measures; then, we characterize how to use measures to solve stochastic reachability problems.

A. Backward Recursion of the State Probability Measure

First, we seek to characterize the backward recursion that enables propagation of probability measures that satisfy staying within a target tube for all probability values. Satisfaction of the target tube requires additional constraints to be placed on this propagation. To ensure that the state probability measure μ_k resides within the target tube set for a time step k, we constrain

$$\mu_k(T_k) = \int_{\mathbb{R}^n} 1_{T_k}(x_k)\mu_k(\mathrm{d}x_k), \tag{5}$$

to equal α. For the final state probability measure to meet the target tube requirement for any probability values $\alpha \in [0, 1]$, we constrain μ_{N-1},

$$\mu_N(T_N) = \int_{\mathcal{X}} 1_{T_N}(x_N)Q(\mathrm{d}x_N|x_{N-1})\mu_{N-1}(\mathrm{d}x_{N-1}), \tag{6}$$

to equal α.

We utilize these constraints in the following theorem to establish satisfaction of the reach measure via state probability measures.

Theorem 1. *If we constrain the state measures such that*

- *(5) holds for $k \in \mathbb{N}_{[0,N-2]}$;*
- *(6) holds;*

then the system in Definition 1 satisfies the reachability specification in Problem 1 for all probability values, α.

Proof. Consider the case in which $k = N - 1$: Note that (6) holds as long as μ_{N-1} satisfies (5). Now consider $k \in \mathbb{N}_{[0,N-2]}$. We employ backwards induction, beginning with the base case $k = N - 2$,

$$\mu_{N-1}(X) = \int_{\mathcal{X}} Q(X \mid x_{N-2})\mu_{N-2}(\mathrm{d}x_{N-2}), \tag{7}$$

holds $\forall X \in \mathscr{B}(\mathcal{X})$ provided that μ_{N-2} satisfies (5). If the equality holds for the case $k = j$ where $j < N - 2$, then it must also hold for the case $k = j - 1$. Observe that for $k = j$ and $k = j - 1$ respectively, the recursions are,

$$\mu_{j+1}(X) = \int_{\mathcal{X}} Q(X \mid x_j)\mu_j(\mathrm{d}x_j), \tag{8a}$$

$$\mu_j(X) = \int_{\mathcal{X}} Q(X \mid x_{j-1})\mu_{j-1}(\mathrm{d}x_{j-1}), \tag{8b}$$

holds $\forall X \in \mathscr{B}(\mathcal{X})$ as long as μ_j and μ_{j-1} satisfy (5). Thus, the propagation of the state probability measures satisfy the reach specification in Problem 1. □

Theorem 1 draws inspiration from [10, Proposition 7.28] which proves the existence of a unique probability measure, stepping backwards in time, to an initial probability measure via the transition kernel. Here, we additionally constrain the state measure to reside in the target tube for some probability α. Nonetheless, by restricting the propagation of the state measures, we are able to characterize state probability measures addressing Problem 1.

B. Constructing Probability Measures via Level Sets

The backward recursion in Theorem 1, if computed directly, would yield the probability measure μ_k associated with the state at each time step. However, a direct computation constitutes a search through a space of probability measures, which is cannot practically be done in the case of continuous domains. We instead use a more tractable formulation based on the relationship between the level sets of the probability of the state lying within the reach tube, α, and the state probability measure. This relationship is predicated on the relationship between the value function in standard stochastic reachability and the relationship to the measure theoretic formulation.

To make this link clear, we reintroduce the backward recursion from [1], [2] with our reachability specification with target tubes, T_k,

$$V_{N-1}(x_{N-1}) = 1_{T_{N-1}}(x_{N-1})\mathbb{E}[1_{T_N}(\mathbf{x}_N)|x_{N-1}], \quad (9a)$$

$$V_{N-2}(x_{N-2}) = 1_{T_{N-2}}(x_{N-2})\mathbb{E}[V_{N-1}(\mathbf{x}_{N-1})|x_{N-2}], \quad (9b)$$

$$\vdots$$

$$V_k(x_k) = 1_{T_k}(x_k)\mathbb{E}[V_{k+1}(\mathbf{x}_{k+1})|x_k]. \quad (9c)$$

We compute the conditional expectation of the value function, $V_{k+1} : \mathbb{R}^n \to [0, 1]$ at time step $k \in \mathbb{N}$, given x_k, from the transition kernel in (1) as

$$\mathbb{E}[V_{k+1}(\mathbf{x}_{k+1})|x_k] = \int_{\mathcal{X}} V_{k+1}(x_{k+1})Q(\mathrm{d}x_{k+1}|x_k), \quad (10)$$

where $\mathbf{x}_{k+1}$ denotes the random state at the next time step. To infer the probability measure from (9) and relate it to the measure-theoretic approach, we introduce the following proposition.

Proposition 1. *If the expectation of the value function, represented by the Lebesgue integral with respect to state probability measure $\hat{\mu}_k$, equals α, That is*

$$\mathbb{E}[V_{N-1}(\mathbf{x}_{N-1})] = \int_{\mathcal{X}} 1_{T_{N-1}}(x_{N-1})\mathbb{E}[1_{T_N}(\mathbf{x}_N)|x_{N-1}]\hat{\mu}_{N-1}(\mathrm{d}x_{N-1}), \quad (11a)$$

$$\mathbb{E}[V_k(\mathbf{x}_k)] = \int_{\mathcal{X}} 1_{T_k}(x_k)\mathbb{E}[V_{k+1}(\mathbf{x}_{k+1})|x_k]\hat{\mu}_k(\mathrm{d}x_k), \quad \text{for } k \in \mathbb{N}_{[0,N-2]}, \quad (11b)$$

both equal α, then the state probability measure $\hat{\mu}_k$ from (11) *and μ_k from Theorem 1 are equivalent.*

Proof. We show this by stepping backwards in time, starting with $k = N - 1$. Given $\hat{\mu}_{N-1}$ which satisfies (11a), it also satisfies both (5) and (6) for some α. Therefore, $\hat{\mu}_{N-1} = \mu_{N-1}$. Likewise, given state probability measures $\hat{\mu}_k$ which satisfy (11b) for $k \in \mathbb{N}_{[0,N-2]}$, they must satisfy (5) since,

$$\hat{\mu}_{k+1}(X) = \int_{\mathcal{X}} Q(X \mid x_k)\hat{\mu}_k(\mathrm{d}x_k), \quad (12)$$

$\forall X \in \mathscr{B}(\mathcal{X})$via (4), results in

$$\hat{\mu}_{k+1}(T_{k+1}) = \int_{\mathcal{X}} 1_{T_{k+1}}(x_{k+1})\hat{\mu}_{k+1}(\mathrm{d}x_{k+1}), \quad (13)$$

equaling α. Thus $\hat{\mu}_k = \mu_k$. □

The value function determines if the current state, x_k, is within the target tube prior to evaluating the conditional expectation, (10). This conditional expectation represents both the state probability measure via the transition kernel and whether each state probability measure lies in the reach tube for the rest of the time horizon. In contrast, as shown in Theorem 1, our approach constrains the current state probability measure, μ_k, *prior* to the propagation to the next constrained state probability measure, μ_{k+1}, which we define via the transition kernel. More importantly, what Proposition 1 identifies is that, for each time step $k \in \mathbb{N}_{[0,N-1]}$, the evaluation of the value function at state x_k corresponds to the level sets of the state probability measure. That is, by identifying the states x_k that align with a given level set probability, α, we can infer the respective level sets of the state probability measures, μ_k, such that (11) holds.

IV. Approximating the Level Sets of the State Probability Measure

To obtain the exact probability measures μ_k would require a direct solution of (9) using direct knowledge of the transition kernel. In cases where this knowledge is not available, we must instead turn to an approximation using what information is available. In this section, we consider an approximation method based on the assumption that the only information available to us is trajectory data. Our ultimate goal is to form a surrogate of the measure as shown in Figure 1: a function approximation that returns, for a given state value and time step, as accurate an estimate of V_k as our data allow.

The essence of our strategy is as follows: first, form empirical estimates $\hat{V}_k$ of the measures μ_k using sample data; then, use the empirical estimates to form training data comprising (state, time step, empirical measure) tuples; and finally, to train a function approximator on that data.

A. Forming the Training Data

We assume that the information available to us takes the following form.

Assumption 1. *The transition kernel is not given, but we can compute state samples, i.e. $x_{k+1,i} \sim Q(\cdot|x_k)$ where $x_{k+1,i} \in \mathbb{R}^n$ provided some $x_k \in \mathbb{R}^n$, where $x_k \in \mathbb{R}^n$ is the current state and $x_{k+1,i} \in \mathbb{R}^n$ is the sample of the next state.*

From these data we compute the empirical estimates

$$\hat{V}_{N-1}(x_{N-1}) = 1_{T_{N-1}}(x_{N-1})\frac{1}{L}\sum_{j=1}^{L} 1_{T_N}(x_{N,j}), \quad (14a)$$

$$\hat{V}_{N-2}(x_{N-2}) = 1_{T_{N-2}}(x_{N-2})\frac{1}{L}\sum_{j=1}^{L} \hat{V}_{N-1}(x_{N-1,j}), \quad (14b)$$

$$\vdots$$

$$\hat{V}_k(x_k) = 1_{T_k}(x_k)\frac{1}{L}\sum_{j=1}^{L} \hat{V}_{k+1}(x_{k+1,j}). \quad (14c)$$

Here, L denotes the number of samples of the subsequent states $x_{k+1,j}$ starting from a state x_k that is sampled from a uniform distribution over the state space, i.e. $x_k \sim \mathrm{Uniform}(\mathcal{X})$. We restrict our attention to a hyperrectangle in the state space, taking $\mathcal{X} = [-b, b]^n$ with bounds $b \in \mathbb{R}$, which we assume encompass the tube sets T_k. This process is contingent on the fact that since we are handling dynamical systems which have no control input, we can avoid a backward recursion via dynamic programming and, instead, employ forward simulation to obtain an empirical estimate. We also note that this process would not be effective for a dynamical system with a control input.

The following algorithm outlines the procedure to generate the sample trajectories $\{\{x_{l,j}\}_{j=1}^{L}\}_{l=k+1}^{N}$ given a single state sample x_k and L x_{k+1} samples via Assumption 1.

Algorithm 1 Trajectory generation procedure.

Input: $x_k \in \mathbb{R}^n$, $L \in \mathbb{N}$
Output: $\{\{x_{l,j}\}_{j=1}^{L}\}_{l=k+1}^{N}$

```
for l ∈ ℕ_[k,N−1] do
    for j ∈ ℕ_[1,L] do
        x_{j,l+1} ∼ Q(·|x_k)
    end for
end for
```

To demonstrate the consistency of our choice of empirical measures, the following result shows that (14) converges to (9) in the limit for a single evaluation point of the state, x_k, as L goes to infinity.

Proposition 2. *If the state samples are produced according to Algorithm 1, then* $\lim_{L\to\infty} \hat{V}_k(x_k) = V_k(x_k)$, $\forall k \in \mathbb{N}_{[0,N-1]}$.

Proof. Let (9) denote the true value function that is defined via a Lebesgue integral [15, Ch. 2, Definition 2.4.1] of a function with respect to the previous transition kernels as well as prior state probability measures and let (14) denote an empirical value function averaged over L samples detailed in Algorithm 1. The strong law of large numbers [15, Ch. 10, 10.10(v)] ensures, as L increases, $\hat{V}_k$ will converge almost everywhere to V_k. Thus, for $k \in \mathbb{N}_{[0,N-1]}$, $\lim_{L\to\infty} \hat{V}_k(x_k) = V_k(x_k)$. □

Fig. 1. The function approximator takes as input the state and time step, then outputs a reach probability value of satisfying the reach specification.

Having established the trajectory data and empirical measures, we proceed to the second step of our strategy: forming the training data. We associate to each trajectory datum x_k a value of the corresponding measure value as estimated by the empirical measures $\hat{V}_k$ to form these estimates. Algorithm 1 demonstrates the exact procedure. To facilitate the training process, we restructure the training dataset, denoted by data, into data' so that each entry, $\mathrm{data}'_{i'}$, corresponds to $\mathrm{data}_{i,k}$ for every i within $[1, M]$ and k within $[0, N-1]$. Thus, the training dataset comprises a collection of tuples, $\mathrm{data}' = \{(x_{i'}, k_{i'}), \hat{\alpha}_{i'}\}_{i'=1}^{M+N}$, where the input includes the state and the time step, while the output is the probability value corresponding to that state and time step.

Algorithm 2 Training data generation procedure.

Input: $[-b, b]^n \subseteq \mathbb{R}^n$, $M, L \in \mathbb{N}$
Output: $\mathrm{data} \in \mathbb{R}^{M\times N}$

```
for k ∈ {N − 1, N − 2, . . . , 1, 0} do
    for i ∈ ℕ_[1,M] do
        x_i ∼ Uniform([−b, b]^n)
        {{x_{l,j}}_{j=1}^{L}}_{l=k+1}^{N} via Algorithm 1
        α̂_{i,k} = V̂_k(x_{k,i}) with {{x_{l,j}}_{j=1}^{L}}_{l=k+1}^{N}.
        data_{i,k} = ((x_{i,k}, k), α̂_{i,k})
    end for
end for
```

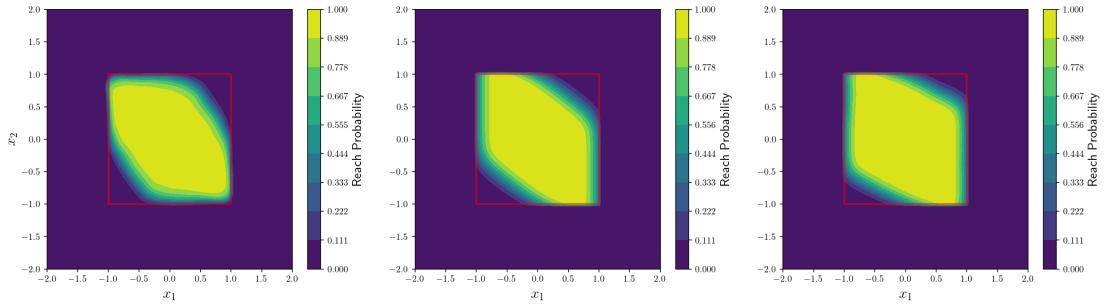
With the training data formed, we proceed to the final step: training the function approximator. Training a function approximator, $\hat{V}_\theta : \mathcal{X} \times \mathbb{N} \to [0, 1]$ according to the training data tuples from the previous step corresponds to solving the following optimization problem:

$$\underset{\theta}{\text{minimize}} \quad \frac{1}{M+N}\sum_{i'=1}^{M+N} J\left(\hat{V}_\theta(x_{i'}, k_{i'}), \hat{\alpha}_{i'}\right), \quad (15)$$

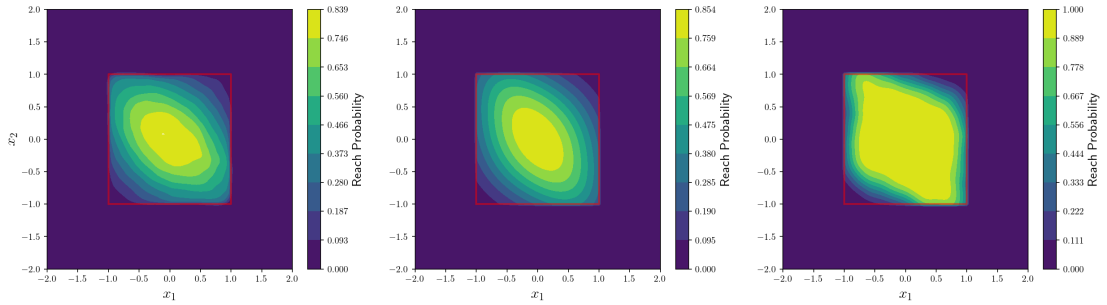
where $J : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function and θ represents the parameters of the function approximator. Once the training is complete, we may use the resulting approximator to solve Problem 1 for any reachability specification we like: the procedure reduces to numerical root-finding to learn the level sets of the state probability measure that satisfy the reach tube specification at time step k.

B. Using DNNs as the Function Approximator

With the strategy for training a function-approximator surrogate for the measures established for an abstract function approximator, we turn to the design choices required to implement the method using a DNN. There are three



(a) Double integrator with variance of $\sigma^2 = 0.01$



(b) Double integrator with variance of $\sigma^2 = 0.1$

Fig. 2. We compare our approach to the ground truth via dynamic programming (DP) as well as to a reproducing kernel Hilbert space (RKHS) approach [9]. In the top row, for a Gaussian disturbance with $\sigma^2 = 0.01$, the RKHS (right) outperforms our method (left), matching closely with DP (middle), as the neural network struggles to learn the sharp drop off of probabilities at the boundaries. However, in the second row, when the Gaussian disturbance has $\sigma^2 = 0.1$ variance, our approach (left) is close to the DP solution (middle) in comparison to RKHS (right). Note that we are able to train the neural network with more samples ($M = 5000$ state samples at training time with $L = 2000$ to compute α offline) versus the RKHS method ($M = 10,201$ samples at training time). When attempting to use the same number of samples ($M \cdot L = 10,000,000$ samples) for the RKHS approach, we ran out of memory.

design choices to make: the loss function, the layer activation functions, and the layer architecture.

Activation Functions: Since the output must lie between zero and one, the last layer uses a sigmoid function. All others use ReLU functions.

Loss Function: Recall that the training problem at hand is to estimate a measure from point evaluations. For this type of supervised learning problem, an effective loss function is the binary cross entropy (BCE) loss,

$$BCE = -\frac{1}{s}\sum_{i'=1}^{s}\Big[\hat{\alpha}_{i'} \cdot \log\Big(\tilde{V}_\theta(x_{i'}, k_{i'})\Big) + (1-\hat{\alpha}_{i'}) \cdot \log\Big(1-\tilde{V}_\theta(x_{i'}, k_{i'})\Big)\Big]. \tag{16}$$

Here, $s \in \mathbb{N}$ denotes the batch size of the data that is being learned in a single pass of the training loop, which continues until the model has trained on the entire data set, i.e. $s < M + N$, then repeats the process over again for a finite number of iterations known as epochs.

Neural Network Structure: We design our DNN as a fully-connected, feed-forward neural network. We use 4 hidden layers where each layer consists of 64 neurons, each employing the Rectified Linear Unit (ReLU) activation function. The input into the neural network is the state, $x \in \mathcal{X}$, and the current time step, $k \in \mathbb{N}$.

Before turning to the examples, we briefly consider the matter of computational complexity involved in evaluating the function approximator. Note that DNNs evaluations scale with the architecture's size, specifically the arrangement and connectivity of its layers, and not the sample size. This means that a trained neural network can accommodate additional data without a proportional increase in the evaluation time. In comparison, kernel methods, which scale with the number of samples, struggle at evaluation time due to the size of the Gram matrix [16]. Although kernel methods have seen scalability improvements via random Fourier features [17], neural networks have made parallel strides in scalability through techniques such as quantization [18], to represent neural networks on a smaller memory footprint, and batching of the data for training with limited memory [19, Ch. 8].

V. Examples

We demonstrate our approach on three examples. For training the algorithm we use PyTorch [20] with the Adam

optimizer [21]. Specifically, we use a learning rate scheduler, which initially starts with a rate of 0.001, and adjusts the learning rate by multiplying it by 0.1 when the loss function flattens, i.e. stops improving, during the training. We use an Xavier initialization scheme to prevent exploding and vanishing gradients which keeps the variance of the activation functions the same across each layer [22]. All computations were done in Python on an Apple M1 Macbook Pro with 16GB of RAM with PyTorch running in CPU mode, making no use of GPU or neural network hardware acceleration. For comparisons to the proposed approach, we employ both dynamic programming (DP) [23] and a reproducing kernel Hilbert spaces (RKHS) approach [9].

A. Double Integrator Experiments

1) 2D Double Integrator: This example compares our approach against dynamic programming and RKHS [9] approaches. Consider dynamics of a double integrator with time horizon $N = 3$ and sampling time $\Delta T = 0.25$,

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k, \tag{17}$$

where $\mathbf{x}_k \in \mathcal{X} = [-2, 2]^2$ is the random state vector. We consider Gaussian disturbances, $\mathbf{w}_k \sim \mathcal{N}(0_2, \sigma^2 I_{2\times 2})$. We consider variances of $\sigma^2 = 0.01$ and $\sigma^2 = 0.1$. The target sets are $T_k = [-1, 1]^2$ for $k \in \mathbb{N}_{[0,N]}$.

We trained the DNN with $M = 5000$ initial samples of the state and $L = 2000$ state samples, which we split into batches of size 101 over 50 epochs to train the neural network. Results for both values of σ are shown in Figure 2a, as well as comparisons with RKHS and dynamic programming. We train the RKHS approach with a sample size of $10,201$ for both variances. For the example with variance $\sigma^2 = 0.01$, the RKHS approach matches closely with the dynamic programming solution, while our approach has error at drop offs in probability at the edges (Figure 2a). However, with variance $\sigma^2 = 0.1$, the RKHS approach does not accurately match the dynamic programming solution. In contrast, our approach is able to handle the larger noise, and is close to the dynamic programming solution, as seen in Figure 2b. We report the errors for both examples in Table I. When using the same number of state samples ($M \cdot L = 10,000,000$) as we use to train the DNN, we ran out of memory attempting to train the RKHS method.

2) n-Dimensional Stochastic Chain of Integrators: In this example, we show that our approach scales linearly with dimension. We consider the n-dimensional stochastic chain of integrators from [5], [24],

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & \Delta T & \cdots & \frac{\Delta T^{n-1}}{(n-1)!} \\ 0 & 1 & \cdots & \frac{\Delta T^{n-2}}{(n-2)!} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k, \tag{18}$$

where we vary n from 2 to $10,000$ dimensions, $\mathbf{x}_k \in \mathcal{X} \subseteq \mathbb{R}^n$ is the random state vector evaluate with a zero vector 0_n,

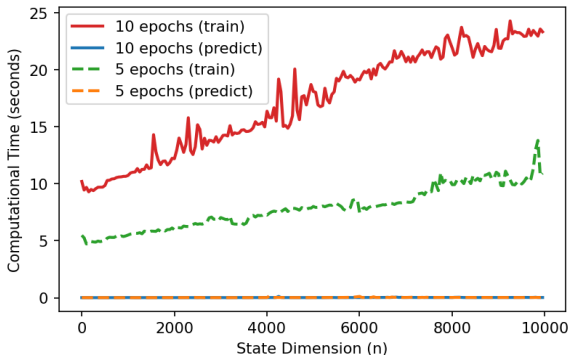


Fig. 3. This plot shows that the training time scales with dimension for the neural network when we fix the number of samples during training. For a fixed sample size, but with increasing state dimension, the neural network scales linearly, similarly to RKHS [24]. Note that increasing the number of epochs increases the slope of the training curve.

and $\mathbf{w}_k \sim \mathcal{N}(0_n, 0.01 I_{n\times n})$ is the Gaussian noise vector, sampling rate is $\Delta T = 0.25$, and the time horizon is $N = 5$. For the reachability specification, we specify the tube sets to be $T_k = [-1, 1]^n$ for $k \in \mathbb{N}_{[0,N]}$. We trained our DNN with $M = 1024$ initial samples of the state and $L = 1$ state samples, which we split into batches of 10 samples during training time. We iterate through the entire training set for 5 and 10 epochs, respectively, to determine how the training procedure scales with additional passes through the training set. Our approach scales linearly and has performance similar to RKHS [24], as shown in Figure 3. In contrast to RKHS, in which samples are contained within the Gram matrix, a neural network can utilize batch processes to train over large amounts of data, without increases to the time needed to evaluate the neural network. Note that the increase in epochs results in a steeper slope.

B. Quaternion Attitude Dynamics

This example demonstrates solution to stochastic reachability problems with nonlinear dynamics. The rigid body dynamics of a torque-free, tumbling object is given by Euler's rotation equations,

$$\begin{aligned} \dot{\omega}_x &= [(I_{yy} - I_{zz})/I_{xx}]\, \omega_y \omega_z \\ \dot{\omega}_y &= [(I_{zz} - I_{xx})/I_{yy}]\, \omega_z \omega_x, \\ \dot{\omega}_z &= [(I_{xx} - I_{yy})/I_{zz}]\, \omega_x \omega_y \end{aligned} \tag{19}$$

where $I \in \mathbb{R}^3$ is the inertia matrix with diagonal entries, I_{xx}, I_{yy}, and I_{zz}, since we assume no products of inertia, and angular velocity about the principle axes is denoted by $\omega = [\omega_x\ \omega_y\ \omega_z]^\top \in \mathbb{R}^3$. The evolution of a unit quaternion representing attitude in the earth-centered inertial frame with the angular velocity in the body frame is

$$\dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix}, \tag{20}$$

where $q = a + bi + cj + dk$ is a unit quaternion on the 3-sphere, denoted by $\mathbb{S}^3$, and it is of unit length, i.e., $a^2 + b^2 + c^2 + d^2 = 1$ [25]. The state vector is $x = [\omega_x, \omega_y, \omega_z, q_w, q_x, q_y, q_z]^\top \in \mathbb{R}^7$, where $[\omega_x, \omega_y, \omega_z]^\top$ represents the angular velocity about the principal axes, q_w is a scalar that represents the angle of rotation, and $[q_x, q_y, q_z]^\top$ is a unit vector that represents the axis of rotation.

To obtain (1), we discretized (19) and (20) via a fourth-order Runge-Kutta scheme, then added Gaussian noise with mean 0 and variance $\sigma^2 = 0.1$ to each of the elements of the quaternion kinematics. We generated points on the surface of the unit sphere, which we represent as pure quaternions, i.e. $q = [0, x, y, z]$, then simulated the attitude dynamics via fourth-order Runge Kutta with time horizon $N = 3$, sampling time $\Delta T = 0.25$, initial constant angular velocity $\omega_0 = [0, 0.5, 1]^\top$, and moments of inertia $I_{xx} = 10$, $I_{yy} = 5$, and $I_{zz} = 7$. Our reachability specification is on a projected space comprised of elevation and azimuth, i.e. $T_k = \left\{\theta, \phi \mid \frac{\pi}{4} \leq \theta \leq \frac{\pi}{2}, \frac{\pi}{4} \leq \phi \leq \frac{3\pi}{4}\right\}$ for $k \in \mathbb{N}_{[0,N]}$, which we can derive via the unit vector component of the quaternion that represents a point in three-dimensional space.

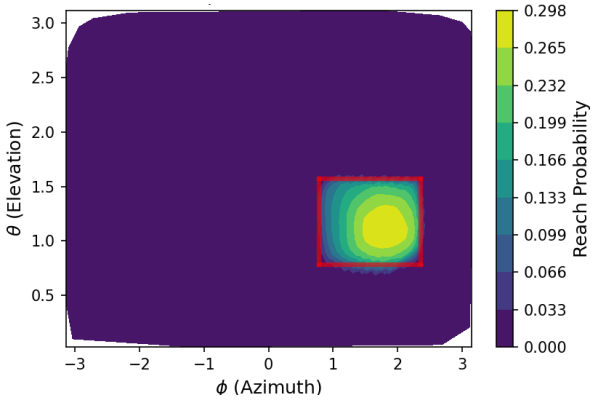
$$\theta = \arccos(q_z) \tag{21a}$$

$$\phi = \arctan(q_y / q_x) \tag{21b}$$

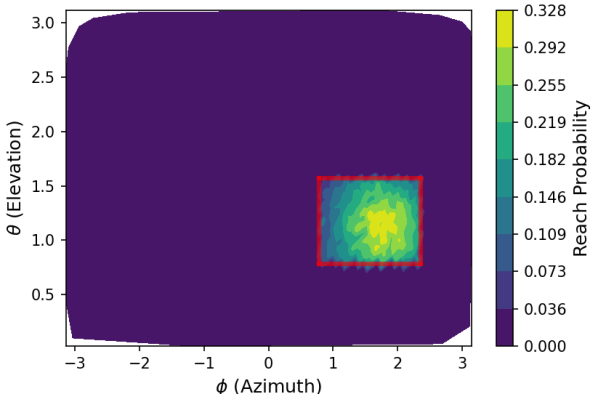
These equations are standard for converting unit quaternions to spherical coordinates [26].

We trained our DNN with $M = 2500$ initial samples of the state and $L = 500$ state samples, which we split into batches of size 10 over 50 epochs to train the neural network. Due to the dimensionality of the state, we could not utilize dynamic programming. However, as shown in Figure 4, the DNN approach matches closely with the empirical estimate via (14), whereas the RKHS does not match closely and yields higher probability values. The error with respect to (14) is provided in Table I. The RKHS approach has both a higher maximum absolute error and average error, which could indicate that it requires additional hyperparameter tuning, cross-validation, and additional data.

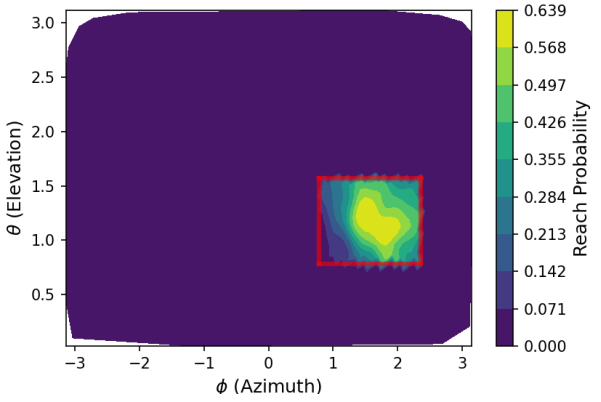
The differences in runtime between the RKHS and DNN approaches in the double integrator and attitude dynamics examples are influenced by the complexity of the systems and number of samples used. In the double integrator example, the RKHS takes longer to compute due to the cost of evaluating a large Gram matrix with more samples. However, in the attitude dynamics example, the RKHS method is quicker, likely because the kernel evaluations and matrix operations are less computationally intensive for this particular sample size of 2500 compared to the DNN's extensive training requirements.



(a) Our approach



(b) Empirical estimate via (14)



(c) RKHS approach

Fig. 4. We compare the proposed approach with an empirical estimate via (14), and the RKHS approach. The neural network (Figure 4a) is closer to the empirical estimate (Figure 4b) than the RKHS (Figure 4c).

VI. Conclusion

We provided a measure theoretic formalism of the stochastic reachability problem for uncontrolled systems, that characterizes stochastic reachability in terms of probability measures. This approach enables new computational tools for approximate solutions to stochastic reachable sets, by computing level sets of the probability measure for all probability values. We developed a numerical implementation that employs deep neural nets to approximate the stochastic reachability probability, for a given state and time step within a finite time horizon. Future work will focus on extensions

Double Integrator with variance $\sigma^2 = 0.01$

Method	State Samples	Max Absolute Error with DP	Average Error with DP	Time (s)
DNN	5000	0.8433	0.0205	18.905
DP	10201	–	–	8.99
RKHS	10201	0.4074	0.0142	53.23

Double Integrator with variance $\sigma^2 = 0.1$

Method	State Samples	Max Absolute Error with DP	Average Error with DP	Time (s)
DNN	5000	0.3106	0.0063	20.35
DP	10201	–	–	9.13
RKHS	10201	0.5091	0.0669	50.158

Quaternion Attitude Dynamics

Method	State Samples	Max Absolute Error with (14)	Average Error with (14)	Time (s)
DNN	2500	0.1477	0.0023	52.186
RKHS	2500	0.3901	0.0164	1.108

TABLE I. Number of state samples, maximum absolute error, average error, and computation times for each method for the double integrator and the quaternion attitude dynamics. The time reported for dynamic programming consists of evaluation on a uniform grid to compute the probabilities via backward recursion. For the DNN and RKHS approaches, we show the combined training and evaluation times.

to accommodate dynamical systems with control inputs. The essence of this extension will be to generalized the transition kernel framing of the dynamics into one that admits inputs, and to extend the sampling procedure to sample over inputs as well. This will require a restriction of the space of control inputs, at least initially, to a finite dimensional space, e.g. to piecewise constant signals.

ACKNOWLEDGEMENTS

We would like to thank Adam J. Thorpe and Kendric R. Ortiz for providing their code to replicate the results for the double integrator example in [9] and for several discussions. We also thank Krishna C. Kalagarla for many discussions regarding our numerical implementation via deep neural networks.

REFERENCES

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[2] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010.

[3] S. Summers, M. Kamgarpour, C. Tomlin, and J. Lygeros, "Stochastic system controller synthesis for reachability specifications encoded by random sets," *Automatica*, vol. 49, no. 9, pp. 2906–2910, 2013.

[4] K. Lesser, M. Oishi, and R. S. Erwin, "Stochastic reachability for control of spacecraft relative motion," in *IEEE Conference on Decision and Control*, 2013, pp. 4705–4712.

[5] A. P. Vinod and M. M. Oishi, "Scalable underapproximation for the stochastic reach-avoid problem for high-dimensional lti systems using fourier transforms," *IEEE control systems letters*, vol. 1, no. 2, pp. 316–321, 2017.

[6] A. Vinod and M. Oishi, "Stochastic reachability of a target tube: Theory and computation," *Automatica*, vol. 125, p. 109458, 2021.

[7] N. Kariotoglou, M. Kamgarpour, T. H. Summers, and J. Lygeros, "The linear programming approach to reach-avoid problems for markov decision processes," *Journal of Artificial Intelligence Research*, vol. 60, pp. 263–285, 2017.

[8] N. Kariotoglou, S. Summers, T. Summers, M. Kamgarpour, and J. Lygeros, "Approximate dynamic programming for stochastic reachability," in *2013 European Control Conference*, 2013, pp. 584–589.

[9] A. J. Thorpe, K. R. Ortiz, and M. M. Oishi, "State-based confidence bounds for data-driven stochastic reachability using hilbert space embeddings," *Automatica*, vol. 138, p. 110146, 2022.

[10] D. Bertsekas and S. E. Shreve, *Stochastic optimal control: The discrete-time case*. Athena Scientific, 1996.

[11] Y. Chow and H. Teicher, *Probability Theory: Independence, Interchangeability, Martingales*. Springer New York, 1997.

[12] D. Henrion and M. Korda, "Convex computation of the region of attraction of polynomial control systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 297–312, 2013.

[13] M. Korda, D. Henrion, and C. N. Jones, "Convex computation of the maximum controlled invariant set for polynomial control systems," *SIAM Journal on Control and Optimization*, vol. 52, no. 5, pp. 2944–2969, 2014.

[14] F. Jiang, G. Chou, M. Chen, and C. J. Tomlin, "Using neural networks to compute approximate and guaranteed feasible hamilton-jacobi-bellman pde solutions," 2017. [Online]. Available: https://arxiv.org/abs/1611.03158

[15] V. Bogachev, *Measure theory*. Springer Berlin, Heidelberg, 2007.

[16] P. Drineas, M. W. Mahoney, and N. Cristianini, "On the nyström method for approximating a gram matrix for improved kernel-based learning." *journal of machine learning research*, vol. 6, no. 12, 2005.

[17] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[18] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[23] A. P. Vinod, J. D. Gleason, and M. M. K. Oishi, "SReachTools: A MATLAB Stochastic Reachability Toolbox," Montreal, Canada, pp. 33 – 38, April 16–18 2019, https://sreachtools.github.io.

[24] A. J. Thorpe and M. M. Oishi, "Model-free stochastic reachability using kernel distribution embeddings," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 512–517, 2019.

[25] Y.-B. Jia, "Quaternions and rotations," *Com S*, pp. 477–577, 2008.

[26] J. E. Marsden and A. Tromba, *Vector Calculus*. Macmillan, 2003.