Neural Bridge Sampling for Evaluating Safety-Critical Autonomous Systems

Aman Sinha*, Matthew O'Kelly*, Russ Tedrake, & John Duchi   Stanford University, University of Pennsylvania, MIT

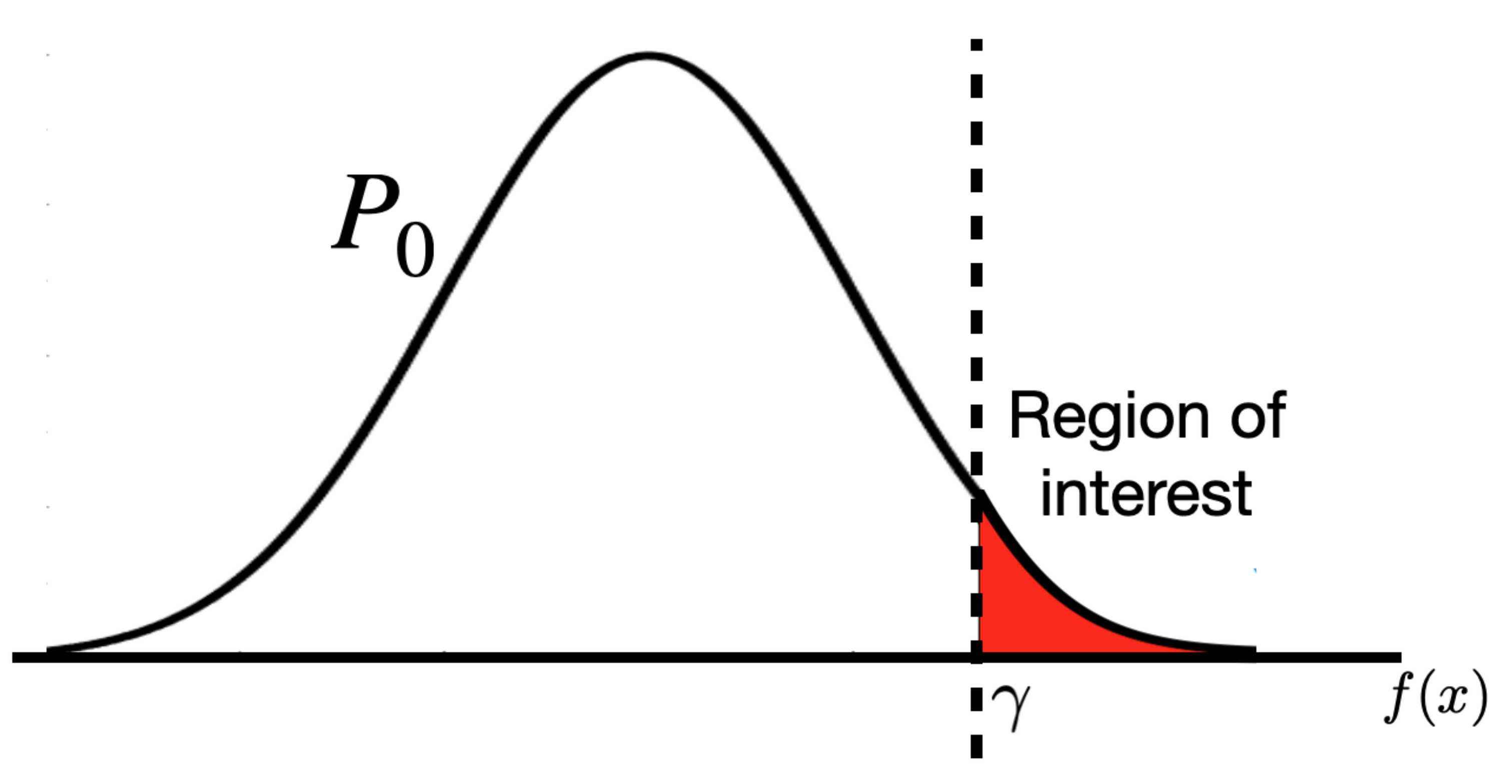
amans@stanford.edu, mokelly@seas.upenn.edu, russt@mit.edu, jduchi@stanford.edu

Introduction

- **Real-world testing** of safety-critical systems is expensive and dangerous
- **Formal verification** of "correctness" is intractable and subjective
- We consider a **risk-based framework**: we evaluate the probability of failures under a generative model of situations that can be evalauted in simulation
- We create a novel **rare-event simulation** techniques that combines **exploration**, **exploitation**, and **optimization** techniques to efficiently find the probability of failures
- We empirically demonstrate the superiority over competing techniques in several real-world applications:
  - sensitivity of a formally-verified system to domain shift
  - design optimization for high-precision rockets
  - model comparisons for two learning-based approaches to autonomous navigation.

Governing problem: failure probability

- **Given**: continuous measure of safety $f : \mathcal{X} \to \mathbb{R}$, threshold level γ, and distribution P_0 of the environment with density rho_0
- **Goal**: Evaluate probability of bad events $p_\gamma := \mathbb{P}_0(f(X) < \gamma)$
- Naive Monte Carlo sampling is too slow for good algorithms / small p_γ: relative variance of estimate $\propto 1/p_\gamma$

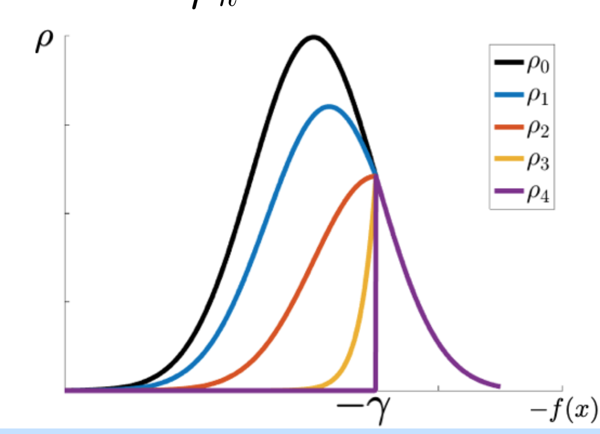


Approach

- Build a ladder towards failure: decompose small probability into a sequence of non-rare probabilites via intermediate distributions ρ_k

$$\rho_k(x) := \rho_0(x) \exp \underbrace{\left(-\beta_k [f(x) - \gamma]_+\right)}_{\text{exponential barrier}},$$

$$Z_k := \int_{\mathcal{X}} \rho_k(x) dx$$



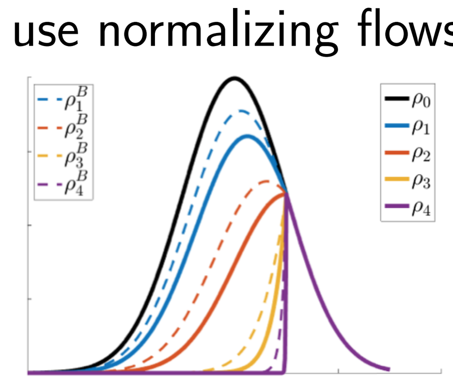
$$p_\gamma := \mathbb{P}_0(f(X) \le \gamma) = \mathbb{E}_{P_K}\left[\frac{Z_K \rho_\infty(X)}{Z_0\, \rho_K(X)}\right], \qquad \frac{Z_K}{Z_0} = \prod_{k=1}^{K} \frac{Z_k}{Z_{k-1}}$$

Algorithm

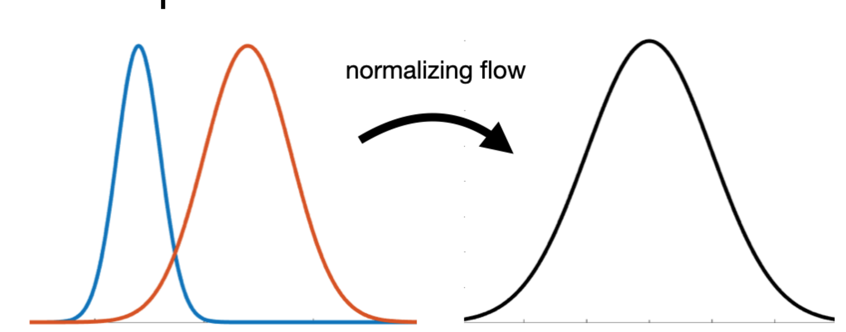
- **Exploit**: Determine the next β_{k+1} using samples from the k^{th} distribution
  - Efficient optimization problem solved via binary search
- **Explore + optimize**: Use a gradient-based based MCMC technique (Hamiltonian Monte Carlo) to sample from the updated distribution
  - Gradients of the intermediate densities automatically trade off between exploration and optimization

$$\nabla \log \rho_k(x) = \underbrace{\nabla \log \rho_0(x)}_{\text{exploration}} - \underbrace{\beta_k \nabla f(x) I\{f(x) > \gamma\}}_{\text{optimization}}$$

- **Estimate**: Estimate Z_{k+1}/Z_k via bridge sampling
  - Use samples form P_k and P_{k+1} to build an auxiliary "bridge distribution"
  - Error of bridge-sampling estimate grows with distance between P_k and P_{k+1} so use normalizing flows to "warp" the space between them



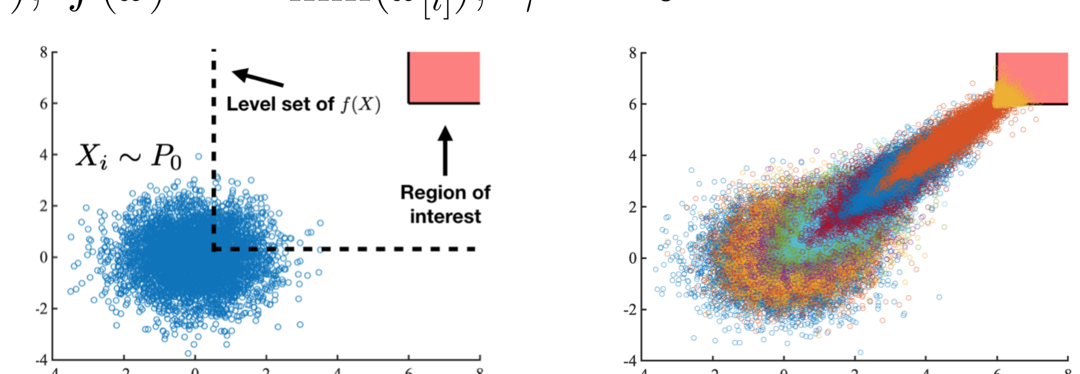
Bridge distributions



Neural warping

Example

- $P_0 = \mathcal{N}(0, I),\ f(x) = -\min(x_{[i]}),\ \gamma = -6$



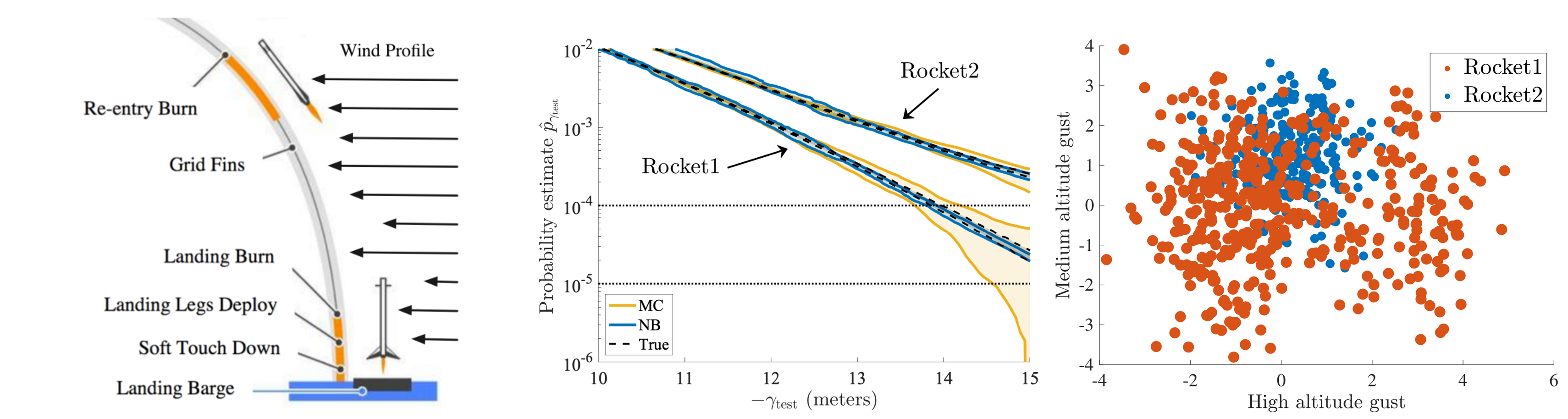
Performance guarantees

- **Theorem**: overall efficiency gain of $O\left(\frac{1}{p_\gamma \log(p_\gamma)^2}\right)$ over naive Monte Carlo
- Relative advantage scales with rarity

Experiments

Rocket landing

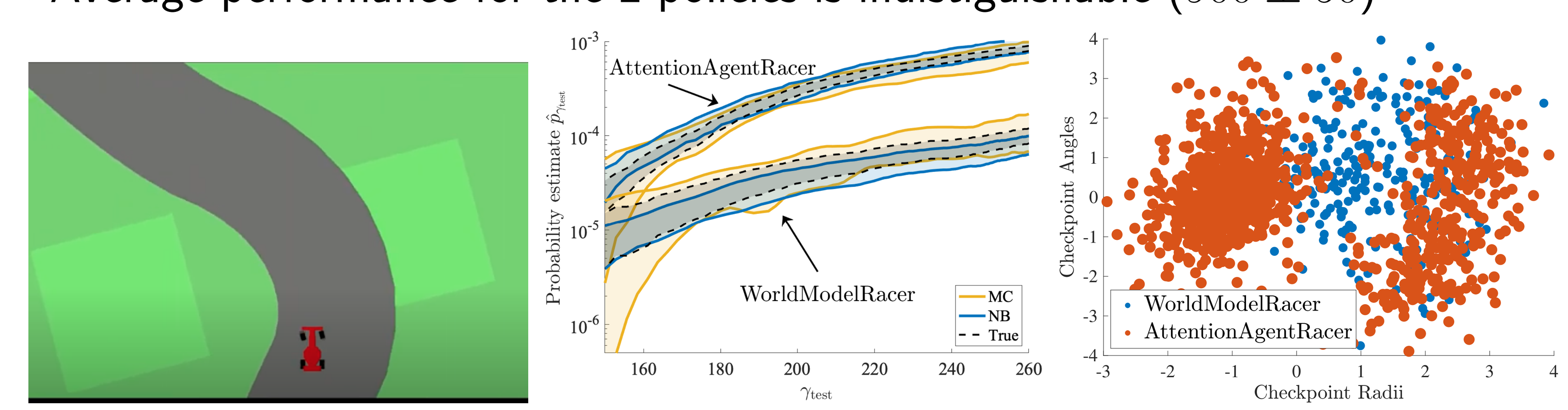
- Comparing 2 engine designs for vertical landing of an orbital-class rocket
- P_0 models the wind gusts throughout the rocket's flight, $f(x)$ measures distance from landing pad's center at touchdown



Environment   Failure rates   Failure modes

CarRacing

- Comparing 2 SOTA policies on the Open Gym CarRacing environment
- P_0 models the racetrack geometry, $f(x)$ measures lap score
- Average performance for the 2 policies is indistiguishable (900 ± 50)



Environment   Failure rates   Failure modes

Quantitative comparisons

- Our approach (NB) outperforms Monte Carlo and other similar techniques
- We achieve tighter estimates of risk for the same number of samples.

Relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ over 10 trials

	Synthetic	MountainCar	Rocket1	Rocket2	AttentionAgentRacer	WorldModelRacer
MC	1.1821	0.2410	1.1039	0.0865	1.0866	0.9508
AMS	0.0162	0.5424	0.0325	0.0151	1.0211	0.8177
B	0.0514	0.3856	0.0129	0.0323	0.9030	0.7837
NB	**0.0051**	**0.0945**	**0.0102**	**0.0078**	**0.2285**	**0.1218**
p_γ	$3.6 \cdot 10^{-6}$	$1.6 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$\approx 2.5 \cdot 10^{-5}$	$\approx 9.5 \cdot 10^{-6}$