Automatic Differentiation of Programs with Discrete Randomness

Gaurav Arya
Massachusetts Institute of Technology, USA
aryag@mit.edu

Moritz Schauer
Chalmers University of Technology, Sweden
University of Gothenburg, Sweden
smoritz@chalmers.se

Frank Schäfer
Massachusetts Institute of Technology, USA
University of Basel, Switzerland
franksch@mit.edu

Chris Rackauckas
Massachusetts Institute of Technology, USA
Julia Computing Inc., USA
Pumas-AI Inc., USA
crackauc@mit.edu

Abstract

Automatic differentiation (AD), a technique for constructing new programs which compute the derivative of an original program, has become ubiquitous throughout scientific computing and deep learning due to the improved performance afforded by gradient-based optimization. However, AD systems have been restricted to the subset of programs that have a continuous dependence on parameters. Programs that have discrete stochastic behaviors governed by distribution parameters, such as flipping a coin with probability p of being heads, pose a challenge to these systems because the connection between the result (heads vs tails) and the parameters (p) is fundamentally discrete. In this paper we develop a new reparameterization-based methodology that allows for generating programs whose expectation is the derivative of the expectation of the original program. We showcase how this method gives an unbiased and low-variance estimator which is as automated as traditional AD mechanisms. We demonstrate unbiased forward-mode AD of discrete-time Markov chains, agent-based models such as Conway's Game of Life, and unbiased reverse-mode AD of a particle filter.

1 Introduction

Automatic differentiation (AD) is a technique for taking a mathematical program $X(p)$ and generating a new program $\widetilde{X}(p) = \frac{\mathrm{d}X}{\mathrm{d}p}$ for computing the derivative [1, 2]. AD is widely used throughout machine learning and scientific computing due to the increased performance of gradient-based optimization techniques compared to derivative-free methods [3]. However, if $X(p)$ returns the flip of a coin with probability p of receiving a 1 and probability $1-p$ of receiving a 0, it is clear that $\frac{\mathrm{d}X}{\mathrm{d}p}$ is not defined in the classical sense. But when attempting to calibrate the parameter p to data, one may wish to fit the model using statistical quantities, e.g. find p such that the average of $X(p)$ is close to the average sum of N real-world coin flips. Given this use case, can one automatically construct a program $\widetilde{X}(p)$ that computes the derivative of the statistical quantities, i.e. $\mathbb{E}[\widetilde{X}(p)] = \frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p}$?

A naïve solution to this problem would be to use finite differences, i.e.:

$$\frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p} \approx \frac{\mathbb{E}[X(p+\varepsilon)] - \mathbb{E}[X(p)]}{\varepsilon}. \tag{1.1}$$

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

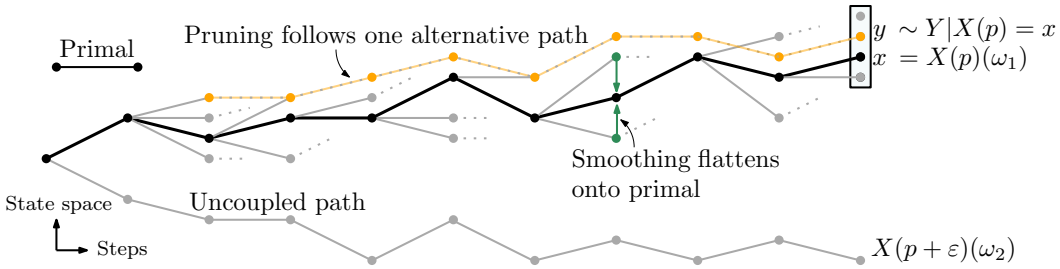


Figure 1: Qualitative sketch of our pruning and smoothing methods and comparison to finite differences, adopting the notation of Section 2. The primal computation (solid black line) samples $X(p)$ with random number sequence ω_1. Black-box finite differences samples the perturbed program $X(p+\varepsilon)$ with an *independent* random number sequence ω_2 (bottom) for some *finite* choice of ε. In contrast, the component Y of the stochastic derivative of the program considers the effect of the minimal possible perturbations (gray lines diverging from primal path) to the original program which could stem from a truly *infinitesimal* change in the input p.

Finite differences' major issue in the context of stochastic programs is that this calculation does not correlate the calculation of $\mathbb{E}[X(p+\varepsilon)]$ with the calculation of $\mathbb{E}[X(p)]$. This leads to a large variance in the finite-difference estimator [4] that goes to ∞ as $\varepsilon \to 0$. This variance issue could be solved if one could run the perturbed program with the same set of random numbers, estimating $\mathbb{E}[X(p+\varepsilon) - X(p)]$ by a method that is well-posed in the limit of $\varepsilon \to 0$, as shown in Fig. 1.

We demonstrate how to automatically construct a new stochastic program $\widetilde{X}(p)$ whose expectation satisfies $\mathbb{E}[\widetilde{X}(p)] = \frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p}$. We derive this through a technique which we term the *stochastic derivative*, propagating the proportional probability of differing event outcomes due to infinitesimal changes in p. Our technique has the following design goals:

- **Composability.** We differentiate stochastic programs $X(p)$ which are themselves composed of many "elementary" stochastic programs, including samples from discrete distributions such as the Bernoulli, Poisson, and Geometric, and samples from continuous distributions, which may be composed (chained, added, etc.) arbitrarily without affecting computational cost.
- **Unbiasedness.** The program's discrete structure is preserved, leading to a provably unbiased estimator that avoids continuous relaxations and tunable or learned internal parameters.
- **Low variance.** We consider correlated paths through the computation that are linked by the smallest possible perturbation (Fig. 1), generalizing the widely-used pathwise gradient estimator for continuous randomness [5] to the discrete case.

We show the utility of our technique by demonstrating forward-mode AD of stochastic simulations like inhomogeneous random walks and agent-based models such as the Game of Life. To achieve $\mathcal{O}(1)$ computational overhead for these applications, we combine stochastic derivatives with an online strategy we call *pruning*. We also demonstrate a straightforward way to perform reverse-mode AD from our approach via *smoothing*, allowing us to derive from first principles the biased but empirically successful straight-through gradient estimator [6] as a special case and to construct an unbiased end-to-end reverse-differentiable particle filter, recovering a technique discovered in [7]. We provide an open-source implementation of the method, `StochasticAD.jl`, for readers to explore the technique on their own applications.

1.1 Related work

Gradient estimators for stochastic functions can be divided into three different classes [5]: score-function [8, 9, 10], measure-valued [11, 12], and pathwise [13, 14] gradient estimators. When it comes to discrete randomness, the score-function method is a popular general-purpose choice because it is unbiased and can be composed through stochastic computation graphs [15]. However, the score-function method does not search for correlated paths through the computation and thus suffers from high variance, making gradient computation with discrete variables challenging. A number of

techniques (REBAR [16], RELAX [17]) have been introduced that rely on control variates for variance reduction [18]. Measure-valued derivatives also have no notion of intrinsic coupling, though coupling can be achieved using common random numbers for certain distributions [19]. As an alternative direction, several recent works such as Gumbel-Softmax [20] consider continuous relaxations of discrete programs so that a pathwise gradient estimator, based on the "reparameterization trick" [14], can then be applied. However, such methods face a variance-bias tradeoff and are inapplicable to discrete programs that cannot be continuously relaxed. Our stochastic derivatives also extend the pathwise gradient estimator to discrete programs but do so unbiasedly. The field of smoothed perturbation analysis [21, 22] develops a mathematically equivalent object to our smoothed stochastic derivative based on conditional expectations, which is a special case of our formalism, and has also considered a randomized approach similar to our pruning technique in the context of generalized semi-Markov processes [23]. However, these ideas have not previously been applied to construct a general-purpose AD method via rigorous composition rules and the algorithms and data structures to realize the approach automatically.

Although our method can be used to hand-derive a gradient estimator, the main feature is composability through user-written functions, enabling an automated mechanism. While mainstream AD frameworks do not support unbiased differentiation of discrete random programs, Storchastic [24] is a specialized framework for AD of stochastic computation graphs [15], where the user can specify which estimator to use at each node, as well as any tunable hyperparameters. Storchastic implements an exhaustive set of prior gradient estimation methods at each sampling step. However, the runtime of the derivative estimate is in general exponential in the length of the largest chain of stochastic nodes, an artifact of the way many prior gradient estimators compose [24, 25]. In Section 3 we demonstrate stochastic AD that matches the computational complexity of deterministic AD even when discrete random functions are chained together, alleviating these performance issues.

2 Composable derivatives of stochastic programs

In this section, we develop the notion of a *stochastic derivative* for programs containing discrete randomness. We shall motivate this object as a natural generalization of the pathwise gradient estimator for continuous random programs, and present the key ideas underpinning the formalism. When describing infinitesimal asymptotics, we say a function $g(\varepsilon)$ is $\mathcal{O}(\varepsilon)$ if $|g(\varepsilon)| \leq C|\varepsilon|$ for some real C and all sufficiently small ε. Colloquially, we describe quantities that are $\mathcal{O}(\varepsilon)$ as "infinitesimal".

2.1 Infinitesimally perturbing a stochastic program

We are interested in differentiating stochastic programs, formally defined below. A stochastic program $X(p)$ can be thought of as a map from an input p to a random variable $X(p)$. Here, $X(p)$ can represent either an "elementary" program such as a draw from a Bernoulli distribution, or the full user-provided stochastic program represented using many elementary programs, e.g. a simulation of a random walk.

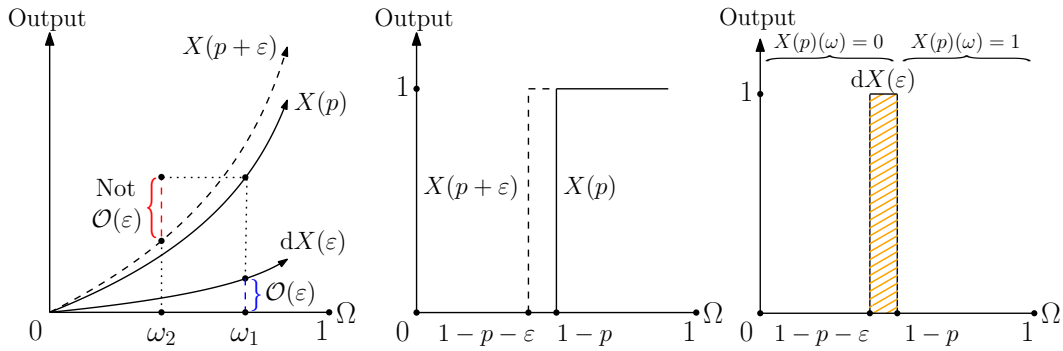
Definition 2.1. A *stochastic program* $X(p)$ is a stochastic process with values in a Euclidean space E, whose index set I is either an open subset of a Euclidean space or a closed real interval.

Let Ω be the sample space, equipped with a probability distribution $\mathbb{P}$. To sample $X(p)$ at input $p \in I$, which we call the *primal evaluation*, one should imagine a sample ω being randomly chosen from Ω according to $\mathbb{P}$ to produce an output $X(p)(\omega) \in E$. Note that $\mathbb{P}$ is independent of p and $X(p)$ is a map $\Omega \to E$; such a formulation has been called the "reparameterization trick" [14]. For example, a Bernoulli distribution $\mathrm{Ber}(p)$ can be represented by choosing a uniform random $\omega \in [0, 1]$ and defining $X(p)(\omega) = \mathbf{1}_{[1-p,1]}(\omega)$, where $\mathbf{1}_S$ is the indicator function for a set S (see Fig. 2b).

At fixed $p \in I$ we define the differential $\mathrm{d}X(\varepsilon)$, which is itself a stochastic program:

$$\mathrm{d}X(\varepsilon) = X(p + \varepsilon) - X(p). \tag{2.1}$$

Let us now restrict our attention to the case where I is a closed real interval, so that $p, \varepsilon \in \mathbb{R}$. The sensitivity of stochastic programs Z with more general index sets can be understood at an input $\mathbf{u}$ by considering at $p = 0$ the directional perturbation $X(p) = Z(\mathbf{u} + p\mathbf{v})$ in a direction $\mathbf{v}$.



(a) $X(p) \sim \mathrm{Exp}(p)$. (b) $X(p) \sim \mathrm{Ber}(p)$. (c) $X(p) \sim \mathrm{Ber}(p)$.

Figure 2: Illustrations of $\mathrm{d}X(\varepsilon)$ for a small finite ε, assuming $\varepsilon > 0$ for simplicity. **(a)** The $\mathcal{O}(\varepsilon)$ range of $\mathrm{d}X(\varepsilon)$ for continuous $X(p)$. The blue bracket indicates a sample from $\mathrm{d}X(\varepsilon)$, while the red bracket indicates a sample obtained by black-box finite differences. **(b)** The original program $X(p)$ and the perturbed program $X(p+\varepsilon)$ for discrete (Bernoulli) $X(p)$. **(c)** Only those samples z below the hatched area cause a non-zero change in output. The hatched area corresponds to $\mathbb{E}[\mathrm{d}X(\varepsilon)]$, the expected size of a change for randomly drawn ω.

For the full user-provided stochastic program, we are interested in computing $\frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p}$, which is related to $\mathbb{E}[\mathrm{d}X(\varepsilon)]$ by

$$\frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p} = \lim_{\varepsilon \to 0} \frac{\mathbb{E}[\mathrm{d}X(\varepsilon)]}{\varepsilon}. \tag{2.2}$$

We thus expect $\mathrm{d}X(\varepsilon)$ to have "infinitesimal" $\mathcal{O}(\varepsilon)$ expectation as ε approaches 0. Let us consider the form of $\mathrm{d}X(\varepsilon)$ for two elementary programs: a program returning a sample from the continuous Exponential distribution $\mathrm{Exp}(p)$ with scale p and a program returning a sample from the discrete Bernoulli distribution $\mathrm{Ber}(p)$ with success probability p. In both cases, the program is parameterized via the inversion method [26] over the sample space $\Omega = [0, 1]$, where $X(p)(\omega)$ is chosen to be non-decreasing in ω (Fig. 2). For $X(p) \sim \mathrm{Exp}(p)$, $X(p)(\omega) = -p \log(1-\omega)$ is differentiable in p. Thus, the range of $\mathrm{d}X(\varepsilon)$ is $\mathcal{O}(\varepsilon)$ for any fixed ω, as illustrated in Fig. 2a. In contrast, consider $X(p) \sim \mathrm{Ber}(p)$, with $X(p)(\omega) = \mathbf{1}_{[1-p,1]}(\omega)$. As shown in Fig. 2b, as $\varepsilon \to 0$ the output of $X(p)$ at a random $\omega \in \Omega$ almost surely does not change in $X(p+\varepsilon)$. Specifically, $\mathrm{d}X(\varepsilon)$ is 0 with probability $1 - \varepsilon \approx 1$, and assumes the value 1 with probability ε (Fig. 2c). This is the fundamental challenge presented by discrete randomness: a $\mathcal{O}(\varepsilon)$ change in the input turns into a *finite* perturbation to the output, where "finite" means that it is non-vanishing as $\varepsilon \to 0$. The perturbation occurs with "infinitesimal" $\mathcal{O}(\varepsilon)$ probability, contributing to the $\mathcal{O}(\varepsilon)$ expectation of $\mathrm{d}X(\varepsilon)$.

2.2 Coupling to the primal

To produce low-variance estimates of the derivative of $X(p)$, it is important that the primal and derivative computations are coupled by a shared source of randomness. To see this, let us understand what happens when they are entirely uncoupled. A black-box finite difference approach with step size ε would independently sample ω_1 and ω_2 from Ω (Fig. 2a, red), computing the derivative estimate $(X(p+\varepsilon)(\omega_2) - X(p)(\omega_1))/\varepsilon$. However, since the samples are independent, the variance of the estimator is of order $1/\varepsilon^2$, so we are forced to pick a finite ε to balance a variance-bias tradeoff.

This motivates using the same random sample for the primal and derivative computations (Fig. 2a, blue). For continuous randomness, taking the limit of this approach as $\varepsilon \to 0$ leads to the widely-used pathwise gradient estimator δ, given as the almost sure limit of $\mathrm{d}X(\varepsilon)/\varepsilon$ as $\varepsilon \to 0$, so that

$$\frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p} = \lim_{\varepsilon \to 0} \frac{\mathbb{E}[\mathrm{d}X(\varepsilon)]}{\varepsilon} \stackrel{?}{=} \mathbb{E}\left[\lim_{\varepsilon \to 0} \frac{\mathrm{d}X(\varepsilon)}{\varepsilon}\right] = \mathbb{E}\left[\delta\right]. \tag{2.3}$$

But, considering a simple Bernoulli variable $X(p) \sim \mathrm{Ber}(p)$ as in Fig. 2c, we see how this approach is ill-suited for the discrete case! As ε approaches 0, the differential $\mathrm{d}X(\varepsilon)$ is non-zero with

infinitesimal $\mathcal{O}(\varepsilon)$ probability. This means that δ is almost surely 0, while the true derivative of $\mathbb{E}[X(p)] = p$ is 1. The finite perturbation is neglected.

Thus, the pathwise gradient estimator needs to be modified to handle discrete randomness. The issue with the interchange of limit and expectation in (2.3) is that $\mathrm{d}X(\varepsilon)/\varepsilon$ is unbounded in ε in the presence of a finite perturbation. This motivates explicitly considering the event of a large jump in $\mathrm{d}X(\varepsilon)$, as characterized by the event $A_B(\varepsilon) = \{|\mathrm{d}X(\varepsilon)| > B|\varepsilon|\}$. Here B is a random bound on the magnitude of $\mathrm{d}X(\varepsilon)/\varepsilon$. To deal with finite perturbations occurring with infinitesimal probability, we can restrict the probability space to $A_B(\varepsilon)$ and sample from this part *separately*.

Definition 2.2 (Stochastic derivative)**.** Suppose $X(p) \in E$ is a stochastic program with index set I a closed real interval. We say that the triple of random variables (δ, w, Y), with weight $w \in \mathbb{R}$ and random $Y \in E$, is a right (left) *stochastic derivative* of X at the input $p \in I$ if $\mathrm{d}X(\varepsilon)/\varepsilon \to \delta$ almost surely as $\varepsilon \to 0$, and there is an integrable random bound $B > |\delta|$ such that for all bounded functions with bounded derivative $f\colon E \to \mathbb{R}$, it holds almost surely that

$$\mathbb{E}\left[w\left(f(Y) - f(X(p))\right) \mid X(p)\right] = \lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon} \mathbf{1}_{A_B(\varepsilon)} \,\middle|\, X(p)\right] \tag{2.4}$$

with limit taken from above (below), where $\mathbb{P}\left(A_B(\varepsilon) \mid X(p)\right)/\varepsilon$ is dominated in ε by an integrable random variable.

The stochastic derivative augments the pathwise gradient estimator δ with a weight w and a random Y following a distribution of alternate values, which characterize the sensitivity of $X(p)$ given the event $A_B(\varepsilon)$. An unbiased estimator of the derivative of $\mathbb{E}[X(p)]$ may then be constructed.

Proposition 2.3 (Unbiasedness)**.** *If (δ, w, Y) is a stochastic derivative of $X(p)$ at p, it holds that*

$$\frac{\mathrm{d}\mathbb{E}\left[X(p)\right]}{\mathrm{d}p} = \mathbb{E}[\delta + w(Y - X(p))]. \tag{2.5}$$

Proof sketch. Applying Definition 2.2 with f as identity, the sensitivity of $X(p)$ over $A_B(\varepsilon)$ [i.e. the right-hand side of (2.4)] is given by $w(Y - X(p))$. When $A_B(\varepsilon)$ does not hold, $|\mathrm{d}X(\varepsilon)/\varepsilon| \leq B$: a dominated convergence argument implies that the sensitivity of $X(p)$ is then given by its almost-sure derivative δ. □

Theorem 2.4 shows the existence of the stochastic derivative for a stochastic program, subject to technical assumptions given in our formal treatment (Appendix B).

Theorem 2.4 (Existence, simplified)**.** *Given a sufficiently regular stochastic program $X(p)$ with index set I a closed interval $[a, b] \subset \mathbb{R}$, there exists a right stochastic derivative (δ, w_R, Y_R) with $w_R \geq 0$ at any $p < b$ and a left stochastic derivative (δ, w_L, Y_L) with $w_L \leq 0$ at any $p > a$.*

Proof sketch. The proof is by construction, where Y has distribution given as the limit as $\varepsilon \to 0$ of the conditional distribution of $X(p+\varepsilon) = X(p) + \mathrm{d}X(\varepsilon)$ over the event $A_B(\varepsilon)$ and the outcome of $X(p)$. In words, Y represents the distribution of possible alternate values of $X(p)$, conditioned on the primal output and on the fact that some finite perturbation occurs. The weight w is then given as the derivative w.r.t. ε of the probability $\mathbb{P}\left(A_B(\varepsilon) \mid X(p)\right)$ that $\mathrm{d}X(\varepsilon)$ jumps by a non-infinitesimal amount conditional on $X(p)$. Essentially, since $\mathbb{P}\left(A_B(\varepsilon) \mid X(p)\right) \approx w\varepsilon$, multiplying by w bridges the gap between *conditioning* on $A_B(\varepsilon)$ and simply restricting the probability space to $A_B(\varepsilon)$ [i.e. multiplying by $\mathbf{1}_{A_B(\varepsilon)}$ as in (2.4)]. □

Example 2.5 (Right stochastic derivative of Bernoulli variable)**.** Suppose $X(p) \sim \mathrm{Ber}(p)$, parameterized via the inversion method, and take $\varepsilon > 0$. As shown in Fig. 2c, $\mathrm{d}X(\varepsilon)$ is parameterized as,

$$\mathrm{d}X(\varepsilon)(\omega) = \begin{cases} 1 & \text{if } 1 - p - \varepsilon \leq \omega \leq 1 - p, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

Given the event $X(p) = 1$, we have that $\omega > 1 - p$ and $\mathrm{d}X(\varepsilon)$ is deterministically 0. On the other hand, given $X(p) = 0$, we have $\omega < 1 - p$, so with probability $\varepsilon/(1-p)$ the differential $\mathrm{d}X(\varepsilon)$ assumes a value of 1, i.e. the Bernoulli variable flips from 0 to 1. Thus, we may construct a right

stochastic derivative $(0, w_R, Y_R)$ of $X(p)$ by letting $w_R = 1/(1-p), Y_R = 1$ conditionally on $X(p) = 0$ and $w_R = Y_R = 0$ conditionally on $X(p) = 1$. For a concrete example, with $p = 0.6$, a Bernoulli variable that is 0 on the primal evaluation has $w_R = 2.5$; `StochasticAD.jl` prints this as `"0 + (1 with probability 2.5ε)"` (as explained further in Section 3.1).

We give a number of examples of stochastic derivatives in Appendix A. It is of crucial importance that w and Y depend conditionally on the output of $X(p)$. In particular, given a primal evaluation $X(p) = x$, the forms of w and Y depend only on the distribution of the perturbed program on the set $\{\omega : X(p)(\omega) = x\} \subset \Omega$, which elegantly generalizes the coupling achieved by the pathwise gradient estimator δ at each fixed $\omega \in \Omega$. Intuitively, this coupling allows us to consider only the smallest possible perturbations to the program in the derivative computation (Fig. 1), and thereby achieve variance reduction without resorting to continuous relaxations. For example, for a binomial variable $X(p) \sim \mathrm{Bin}(n, p)$ parameterized via the inversion method (a natural parameterization for maximizing coupling), it holds that $Y \in \{X(p) - 1, X(p) + 1\}$. We show in Example A.1 that the variance of our gradient estimator for such a binomial variable is $\mathcal{O}(n)$, whereas the score function estimator has variance $\mathcal{O}(n^3)$, justifying this intuition.

2.3 Composition of stochastic derivatives

The key is *composition*, i.e. a stochastic derivative "chain rule". While Definition 2.2 requires composition through deterministic test functions f to enforce a sufficiently strict definition, Theorem 2.6 provides a general-purpose composition result through any possibly random and/or discrete program which has a stochastic derivative, as well as multidimensional programs with directional stochastic derivatives (i.e. stochastic derivatives of directional perturbations of the program).

Theorem 2.6 (Chain rule, simplified)**.** *Consider independent stochastic programs X_1 and X_2 and their composition $X_2 \circ X_1$. Suppose that X_1 has a right (left) stochastic derivative at $p \in \mathbb{R}$ given by (δ_1, w_1, Y_1), and X_2 has a right stochastic derivative (δ_2, w_2, Y_2) in the direction $\hat{\delta}_1 = \delta_1/|\delta_1|$ given conditionally on its input $X_1(p)$. Then, under assumptions to ensure integrability, the stacked program $[X_1; X_2 \circ X_1]$ has a right (left) stochastic derivative at p given by (δ, w, Y) where $\delta = [\delta_1; |\delta_1|\delta_2]$,*

$$Y = \begin{cases} [Y_1; X_2(Y_1)] & \textit{with probability} \quad \dfrac{w_1}{w_1 + |\delta_1| \cdot w_2}, \\ [X_1(p); Y_2] & \textit{with probability} \quad \dfrac{|\delta_1| \cdot w_2}{w_1 + |\delta_1| \cdot w_2}, \end{cases} \tag{2.7}$$

and $w = w_1 + |\delta_1| \cdot w_2$.

Proof sketch. Let $A_1(\varepsilon)$ be the event of a jump in X_1 and $A_2(\varepsilon)$ be the event of a jump in X_2 when its input is perturbed by $\varepsilon\delta_1$. Given $A_1^c(\varepsilon) \cap A_2^c(\varepsilon)$ (a jump in neither), a dominated convergence argument implies that $X_2 \circ X_1$ does not jump either. Given $A_1(\varepsilon)$, Definition 2.2 yields that the sensitivity of $X_2 \circ X_1$ is described by alternate values Y_1 with weight w_1, while given $A_2(\varepsilon)$ the sensitivity of $X_2 \circ X_1$ is described by alternate values Y_2 with weight $|\delta_1|w_2$. We may then form w as the sum of these weights and Y as a weighted distribution over the two cases. Crucially, we may neglect the sensitivity of $X_2 \circ X_1$ given $A_1(\varepsilon) \cap A_2(\varepsilon)$ (a jump in both), as this event has probability $\mathcal{O}(\varepsilon^2)$. □

2.4 Smoothed stochastic derivatives

The reparameterization trick neglects finite perturbations, while stochastic derivatives precisely capture all possible finite perturbations. There exists a middle ground between these methods: one may take a conditional expectation on $X(p)$ so that finite perturbations have been "smoothed" into infinitesimal ones. (E.g. `"0 + (1 with probability 2.5ε)"` becomes `"0 + 2.5ε"`.)

Definition 2.7 (Smoothed stochastic derivative)**.** For a stochastic program $X(p)$ with a right (left) stochastic derivative (δ, w, Y) at input p, a right (left) smoothed stochastic derivative $\widetilde{\delta}$ of X at input p is given as

$$\widetilde{\delta} = \mathbb{E}\left[\delta + w(Y - X(p)) \mid X(p)\right]. \tag{2.8}$$

```julia
struct StochasticTriple
    value # primal evaluation
    δ # "infinitesimal" component
    Δs # component of discrete change
       # with "infinitesimal"
       # probability
end
```

```julia
using Distributions
function X(p)  p = 0.6 + ε
    a = p^2  0.36 + 1.2ε
    b = rand(Binomial(10, p))
     6 + (1 with probability 10.0ε)
    c = 2 * b + 3 * rand(Bernoulli(p))
     12 + (3 with probability 12.5ε)
    return a * c * rand(Normal(b, a))
end
```

```julia
julia> using StochasticAD
julia> st = stochastic_triple(X, 0.6) # sample a single stochastic triple at p = 0.6
 27.11 + 94.32ε + (6.78 with probability 12.5ε)
julia> derivative_contribution(st) # which produces a single derivative estimate...
 179.04
julia> samples = [derivative_estimate(X, 0.6) for i in 1:1000] # take many estimates!
julia> println("d/dp of E[X(p)]: $(mean(samples)) ± $(std(samples) / sqrt(1000))")
 d/dp of E[X(p)]: 204.63 ± 1.25
```

Figure 3: **Left:** Stochastic triple structure (simplified). **Right:** A toy program $X(p)$, using discrete distributions $\mathrm{Bin}(n)$, $\mathrm{Ber}(p)$, and the continuous normal distribution $\mathcal{N}(\mu, \sigma)$; the used stochastic derivatives are given in Appendix A. Highlights show intermediate values during a single derivative estimate. **Bottom:** Differentiating $\mathbb{E}[X(p)]$; printout float precision reduced for clarity.

Smoothed stochastic derivatives immediately permit reverse-mode AD instead of forward-mode, as they have the same form as the traditional derivative. However, they are in general biased: they propagate through smooth functions f that are linear over the conditional distribution of Y given $X(p)$ via the standard chain rule, as we prove in Appendix B.6. Due to the coupling of Y and $X(p)$, this is a much weaker requirement than global linearity, which can lead to low-bias estimates: for example, for the program $X(p) = \mathrm{Geo}(p)^3$ smoothed stochastic derivatives give a derivative estimate with $< 0.5\%$ bias at $p = 0.01$, even though the cube function is highly non-linear on the inter-quartile range $[28, 137]$ of $\mathrm{Geo}(p)$. Smoothed stochastic derivatives recover the widely-used straight-through gradient estimator [6] as a special case, as we work out in Example A.8.

3 Automatic differentiation of stochastic programs

We develop `StochasticAD.jl`, a prototype package for stochastic AD based on our theory of stochastic derivatives. As discussed in Section 2.4, *smoothed* stochastic derivatives obey the usual chain rule, and thus can be used with existing AD infrastructure by supplying custom rules for discrete random constructs, and we do so for a particle filter in Section 3.4. However, performing automatic differentiation with unsmoothed stochastic derivatives, which are unbiased in all cases, requires new innovation. We develop a novel computational object called a *stochastic triple*, introduced in Section 3.1 and showcased in Section 3.2 and Section 3.3.

3.1 Educational toy example of stochastic triples

Forward-mode AD is often implemented with *dual numbers* [2], which pair the primal evaluation of a deterministic function $f(p)$ with its derivative $\frac{\mathrm{d}}{\mathrm{d}p} f(p)$. Dual numbers can be propagated through a program using the chain rule. A useful alternative perspective of dual numbers is that they propagate an "infinitesimal" perturbation ε to the input through the program, where $\frac{\mathrm{d}}{\mathrm{d}p} f(p)$ is the coefficient of the "dual" element ε. Stochastic triples generalize dual numbers by including a third component Δs to describe finite perturbations with infinitesimal *probability* (Fig. 3, left).

In Fig. 3, we consider a toy program $X(p)$ including discrete randomness. We are interested in the sensitivity of $X(p)$ at $p = 0.6$, and hence we provide the stochastic triple printed as `"0.6 + ε"` as input. First, the triple is squared and becomes `"0.36 + 1.2ε"`: this is the familiar way that dual numbers propagate, via the chain rule. But what happens when the triple is propagated

(a)

```
...
X = 0
for step in 1:n
  i = rand(Categorical(probs(X)))
  X += steps[i]
end
return f(X)
```

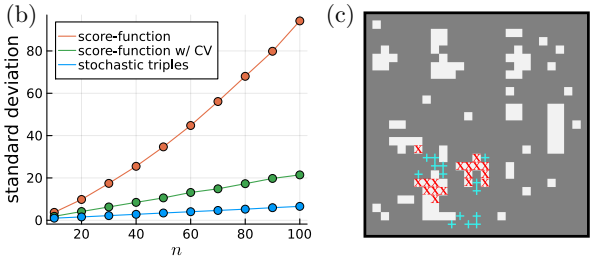


Figure 4: Automatic differentiation of discrete-time Markov processes. **(a)** Code snipped for a 1D random walk, which can be automatically differentiated by `StochasticAD.jl`; `probs(X)` gives the transition probabilities at value `X` and `steps[i]` gives the step size for the `i`th transition. **(b)** The variance of unbiased gradient estimates of the random walk program using stochastic triples and the score function, which is applied both without a control variate (CV) and with a pre-computed batch-average CV. **(c)** The final board for one run of the stochastic game of life with $N = 25$ and $T = 10$, where the "+" signs represent additional living cells (white) in the stochastic alternative path, and the "X" signs represent additional dead cells (grey).

through the discrete and random Binomial variable? The resultant stochastic triple `"6 + (1 with probability 10.0ε)"` is an integer with a component of *discrete* change, reflecting an infinitesimal probability of one more success in the Binomial $\mathrm{Bin}(10, 0.6)$. We can in fact understand why the probability is `"10ε"` by representing the Binomial as the sum of 10 Bernoulli variables, each with probability 0.6. Since 4 of the Bernoulli's have an output of 0, they each have a probability `"2.5ε"` of switching to 1 (recall Example 2.5), and thus there is in total a `"10ε"` probability that the output of the Binomial increases by 1 (rigorously, we have applied Theorem 2.6). Formally, we can interpret a printout `"x + δε + (Δ with probability wε)"` as follows: `x` is a sample of the random variable $X(p)$ describing the primal evaluation, and `δ`, `w`, and `x+Δ` are samples of the components δ, w, and Y, respectively, of the stochastic derivative of X at p.

Using our chain rule for stochastic derivatives (Theorem 2.6), we write rules for propagating stochastic triples through functions via operator overloading [27], exploiting Julia's multiple dispatch feature [28]. When multiple discrete changes are possible, we pick one probabilistically: we call this strategy *pruning* (recall Fig. 1) and show its unbiasedness in Appendix B.5. For example, in line 6 of Fig. 3, right, we probabilistically choose between the perturbation to the Binomial and the perturbation to the Bernoulli, in this case picking the latter. To handle causal relationships between perturbations as in the first case of Eq. (2.7), we associate each perturbation with a tag to avoid erroneously pruning between two perturbations that occur simultaneously. Thus, stochastic triples can efficiently propagate through the full toy function written in Fig. 3. The function `derivative_estimate` creates a stochastic triple, propagates it, and collapses it into the derivative estimate `δ + wΔ`, forming an unbiased estimate of the derivative via Definition 2.2 (Fig. 3, bottom).

3.2 Inhomogeneous random walk

We consider a Markovian random walk $x_0, \ldots, x_n$ on $\mathbb{Z}_{\geq 0}$, with transition behavior dependent on a parameter p as follows,

$$x_n = \begin{cases} x_{n-1} + 1 \text{ with probability } \exp\left(-\frac{x_{n-1}}{p}\right) \\ x_{n-1} - 1 \text{ with probability } 1 - \exp\left(-\frac{x_{n-1}}{p}\right) \end{cases}, \quad x_0 = 0. \tag{3.1}$$

We consider a program that stochastically simulates this walk and applies an arbitrary non-linear function f to the output x_n. In practice, f may represent a loss or a likelihood estimate; in this toy setting, we take $f(x) = x^2$. We are interested in studying the asymptotic behavior of the variance of our automatically derived gradient estimator, and so set $p = n$ so that the transition function varies appreciably over the range of the walk for all n. We find that the stochastic triple estimator has asymptotically lower variance than the score function estimator (Fig. 4b). Crucially, stochastic triples achieve this variance reduction while remaining entirely in discrete space, automatically producing a gradient estimate that is provably unbiased.

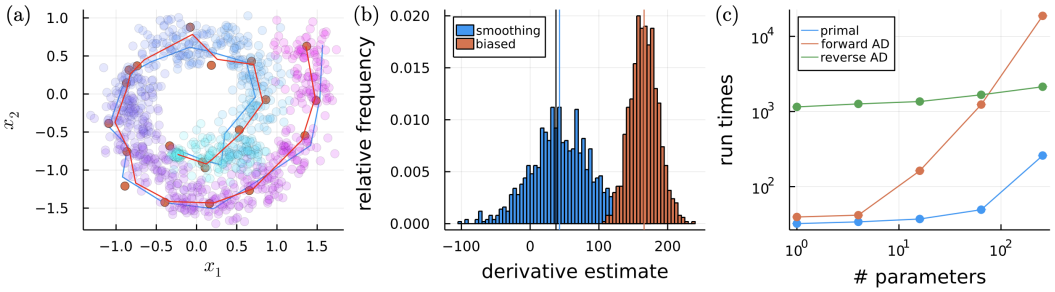


Figure 5: Unbiased differentiable particle sampler using smoothed stochastic derivatives. **(a)** Latent path (blue) and observations (orange) for a particular realization of the process given by the hidden Markov model (see Appendix C for details) in $d = 2$ dimensions, including the recursively applied Kalman filter estimator (red) and the particles of the particle filter (going from blue at $n = 1$ to pink at $n = 20$). **(b)** The value of the derivative of $\log \mathcal{L}$ with respect to the first parameter for $d = 2$ calculated by differentiating the Kalman filter is marked by a black line. The blue distribution corresponds to the derivative computed with smoothed stochastic derivatives. The orange distribution represents the biased derivative approach, where the resampling step is not differentiated. The means of the two distributions are highlighted by a line in the respective color. In both cases, we use 1000 samples. **(c)** Averaged run times in ms comparing forward- and reverse-mode AD for increasing numbers of parameters.

3.3 Stochastic game of life

For our next example, we differentiate a stochastic version of John Conway's game of life, played on a two-dimensional board. In the traditional game of life, a dead cell becomes alive when 3 of its neighbors are alive, while a living cell survives when 2 or 3 of its neighbors are alive. In our stochastic version, each of these events instead has probability 95%, while their complementary events have probability 5%. Such types of discrete stochastic examples arise in many applications. For example, mixing machine learning with agent-based models found in epidemiology and sociological contexts [29] or rule-based models and discrete stochastic (Gillespie) simulations of discrete chemicals in systems biology [30, 31] requires similar program constructs.

Consider a program that populates each cell of an $N \times N$ board with probability p, runs the stochastic game of life for T time steps, and counts the number of living cells n_{living}. We perform a sensitivity analysis of the final living population with respect to the initial living population, i.e. differentiate the expectation of n_{living} with respect to p. Stochastic triples propagate fully through the program, leading to an unbiased estimate of the derivative, as we verify with black-box finite differences. The final board is depicted in Fig. 4c, along with the difference to the alternative final board chosen by pruning. This is a high-dimensional example (the state space has dimension $N^2 = 625$) with fundamentally discrete structure, providing support for the algorithmic correctness and generality of stochastic triples. In particular, the program cannot directly be continuously relaxed since it includes array indexing; in the approach of stochastic triples, integer-valued quantities stay integers.

3.4 Particle filter

As a final example, we consider a hidden Markov model with random latent states $X_1, \ldots, X_n$, observations $y_1 \sim Y_1, \ldots, y_n \sim Y_n$, and parameters θ. The likelihood $\mathcal{L} = p(y_1, \ldots, y_n \mid \theta)$ is in general not tractable, but a particle filter can be used to compute an estimate of $\mathcal{L}$. Here, we assume familiarity with particle filters [32, 33] but provide a short exposition with focus on the resampling step and experimental details in Appendix C. We assume the latent states to be continuous random, but discete randomness enters through the resampling step.

To differentiate the particle filter resampling step, we provide a stochastic derivative formulation of the returned particles and importance weights (Appendix C). Importantly, $\mathcal{L}$ can be expressed by the sum of the weights at the last step, and the weight assigned to each particle in the resampling steps

is used in a *purely linear* way. Smoothed stochastic derivatives permit *unbiased* reverse-mode AD in this case. Our formulated approach, though derived very differently, is equivalent to the particle filter AD scheme developed in [7] implementing the first estimator derived in [34], as we show in Appendix C. Our particular choice of system allows the calculation of a ground-truth gradient of $\mathcal{L}$ by differentiating the Kalman filter algorithm [35]. Fig. 5a visualizes the latent process and observations, and the Kalman and particle filter trajectories. Our estimator agrees with the Kalman filter derivative, unlike biased estimators [36, 37, 38] that neglect the contribution of the resampling step or perform it with entropy-regularized optimal transport (Fig. 5b). For our program, we observe reverse-mode AD to perform better for more than ≈ 100 parameters (Fig. 5c). However, we find that the variance increases more rapidly with the number of steps and dimension as compared to biased estimators, suggesting that there is room for improvement in the coupling approach [7, 34].

4 Limitations and outlook

We have presented a method for unbiased AD of programs with discrete randomness, which, we have argued, is a natural generalization of pathwise gradient estimators based on the reparameterization trick to the discrete case. However, more work will need to be done to turn our software demonstration `StochasticAD.jl` into an AD system capable of handling the full complexity of applications such as machine learning. A useful improvement would be better support for discrete constructs such as `if` statements with discrete random input; currently, such branches need to be rewritten using array indexing, which is supported. Further, an interesting direction for future work is to automatically handle functions which deterministically turn a continuous random quantity into a discrete one, such as an inequality comparison `X(p) > 0` or a Bernoulli variable implicitly implemented as `rand() < p`, and also to handle constructs such as while loops based on such functions.

We also expect future work to focus on further variance reduction. Our method's variance depends on the degree to which the primal and derivative computations can be *coupled*: while we present a natural method of coupling for a number of elementary stochastic programs and their compositions, the design space is rich when it comes to challenging examples such as the game of life or the resampling step of a particle filter. (This design space is reflected in the fact that the form of the stochastic derivative depends not only on a program's probability distribution, but also on the way it is parameterized.) Furthermore, the pruning operation can introduce additional variance: for example, the derivative estimate automatically produced for the program $B_1 + 2B_2$ with i.i.d. $B_i \sim \mathrm{Bin}(n, p)$ has variance $\mathcal{O}(n^2)$ due to the pruning between the `+1` and `+2` perturbations, even though $\mathcal{O}(n)$ variance can be achieved for $\mathrm{Bin}(3n, p)$. Smoothing does not face this issue but accrues a bias through non-linear functions. Pruning and smoothing may be thought of as the simplest ways to construct an AD algorithm from stochastic derivatives, lying on opposite ends of the design space: we anticipate future work to address their suboptimalities and ideally form unbiased estimators for discrete random programs that fully close the variance gap to their continuous counterparts [14, 39]. Finally, going beyond smoothing for reverse-mode AD, and ideally achieving unbiasedness while remaining coupled to the primal, is an important open problem for large-scale applications.

Acknowledgments and disclosure of funding

We thank Alan Edelman, Guillaume Dalle, and the anonymous reviewers for their feedback, and Emile van Krieken for helpful discussions regarding composing gradient estimators. We acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the research results reported within this paper. This material is based upon work supported by the National Science Foundation OAC-1835443, SII-2029670, ECCS-2029670, OAC-2103804, and PHY-2021825; the Advanced Research Projects Agency-Energy DE-AR0001211 and DE-AR0001222; the Defense Advanced Research Projects Agency (DARPA) HR00112290091; the United States Artificial Intelligence Accelerator FA8750-19-2-1000; the Chalmers AI Research Centre; and the Swiss National Science Foundation 51NF40-185902. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] Andreas Griewank et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989.

[2] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18:1–43, 2018.

[3] George Corliss, Christele Faure, Andreas Griewank, Laurent Hascoet, and Uwe Naumann. *Automatic differentiation of algorithms: from simulation to optimization*. Springer Science & Business Media, 2002.

[4] Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*, volume 57. Springer, 2007.

[5] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62, 2020.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013.

[7] Adam Ścibior and Frank Wood. Differentiable particle filtering without modifying the forward pass. *arXiv:2106.10314*, 2021.

[8] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[9] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[10] Jack PC Kleijnen and Reuven Y Rubinstein. Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3):413–427, 1996.

[11] G Ch Pflug. Sampling derivatives of probabilities. *Computing*, 42(4):315–328, 1989.

[12] Bernd Heidergott and FJ Vázquez-Abad. Measure-valued differentiation for Markov chains. *Journal of Optimization Theory and Applications*, 136(2):187–209, 2008.

[13] Paul Glasserman and Yu-Chi Ho. *Gradient estimation via perturbation analysis*, volume 116. Springer Science & Business Media, 1991.

[14] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv:1312.6114*, 2013.

[15] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in Neural Information Processing Systems*, 28, 2015.

[16] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

[17] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv:1711.00123*, 2017.

[18] Peter W Glynn and Roberto Szechtman. Some new perspectives on the method of control variates. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 27–49. Springer, 2002.

[19] Bernd Heidergott, Felisa J Vázquez-Abad, and Warren Volk-Makarewicz. Sensitivity estimation for Gaussian systems. *European Journal of Operational Research*, 187(1):193–207, 2008.

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *arXiv:1611.01144*, 2016.

[21] Wei-Bo Gong and Yu-Chi Ho. Smoothed (conditional) perturbation analysis of discrete event dynamical systems. *IEEE Transactions on Automatic Control*, 32(10):858–866, 1987.

[22] Paul Glasserman and W-B Gong. Smoothed perturbation analysis for a class of discrete-event systems. *IEEE Transactions on Automatic Control*, 35(11):1218–1230, 1990.

[23] Michael Fu and Jian-Qiang Hu. *Conditional Monte Carlo Gradient Estimation*, pages 73–131. Springer US, Boston, MA, 1997.

[24] Emile Krieken, Jakub Tomczak, and Annette Ten Teije. Storchastic: A framework for general stochastic automatic differentiation. *Advances in Neural Information Processing Systems*, 34, 2021.

[25] Emile Krieken, Jakub Tomczak, and Annette Ten Teije. Openreview: Storchastic: A framework for general stochastic automatic differentiation. URL https://openreview.net/forum?id=KAFyFabsK88.

[26] Luc Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006.

[27] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[28] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[29] Eric Silverman, Umberto Gostoli, Stefano Picascia, Jonatan Almagor, Mark McCann, Richard Shaw, and Claudio Angione. Situating agent-based modelling in population health research. *Emerging Themes in Epidemiology*, 18(1):1–15, 2021.

[30] Lily A Chylek, Leonard A Harris, Chang-Shung Tung, James R Faeder, Carlos F Lopez, and William S Hlavacek. Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6(1):13–36, 2014.

[31] James R Faeder, Michael L Blinov, and William S Hlavacek. Rule-based modeling of biochemical systems with bionetgen. In *Systems biology*, pages 113–167. Springer, 2009.

[32] Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*. Springer, 2020.

[33] Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[34] George Poyiadjis, Arnaud Doucet, and Sumeetpal S Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

[35] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82, 1960.

[36] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv:1805.11122*, 2018.

[37] Adrien Corenflos, James Thornton, George Deligiannidis, and Arnaud Doucet. Differentiable particle filtering via entropy-regularized optimal transport. In *International Conference on Machine Learning*, pages 2100–2111. PMLR, 2021.

[38] Conor Rosato, Paul Horridge, Thomas B Schön, and Simon Maskell. Efficient learning of the parameters of non-linear models using differentiable resampling in particle filters. *arXiv:2111.01409*, 2021.

[39] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv:1611.00712*, 2016.

[40] Fernando Mazzone. A characterization of almost everywhere continuous functions. *Real Anal. Exchange*, 21(1):317–319, 1995/96.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 3 and Section 4 for discussions on potentially large variance of the estimator and language-wide AD support. In addition, we provide a continuously updated list about the current limitations of the `StochasticAD.jl` package in its README.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix D and the README of `StochasticAD.jl`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] For the differentiable particle sampler, we have included the error on the mean value of the run times as error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the Julia packages on which this work is based.
   (b) Did you mention the license of the assets? [No] We only use open-source software packages.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We have included our package in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We do not use any previously collected data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Table 1: Probability mass functions and cumulative distributions of integer-valued stochastic programs, where x is assumed in the domain of $X(p)$.

$X(p)$	Parameter	$\mathbb{P}(X(p) = x)$	$\mathbb{P}(X(p) \leq x)$	$\partial_p \mathbb{P}(X(p) \leq x)$
$\text{Ber}(p)$	Probability	$px + (1-p)(1-x)$	$(1-p) + px$	$x - 1$
$\text{Geo}(p)$	Probability	$p(1-p)^x$	$1 - (1-p)^x$	$x(1-p)^{x-1}$
$\text{Pois}(p)$	Rate	$p^x/(e^p x!)$	$\sum_{k=0}^{x} e^{-p} p^k / k!$	$-p^{x+1}/(e^p(x+1)!)$

In the supplement, we:

- provide a number of examples of stochastic derivatives and present hand-worked examples of how they compose (Appendix A),
- prove our formal statements regarding stochastic derivatives, and state and prove formal statements regarding smoothed stochastic derivatives (Appendix B),
- provide an introduction to the particle filter methodology, and show how smoothed stochastic derivatives leads to unbiased differentiation of the resampling step (Appendix C),
- provide details about the hardware used for the experiments of the main text and the software dependencies of `StochasticAD.jl` (Appendix D).

A Examples of stochastic derivatives

In this section, we present examples of stochastic derivatives for a number of stochastic programs, recalling Definition 2.2.

Example A.1 (Integer-valued stochastic programs)**.** The Bernoulli, Binomial, Geometric, and Poisson distributions are all discrete, and in fact (nonnegative) integer-valued. Suppose $X(p)$ follows one of these distributions, parameterized using the inversion method [26] over the interval $[0, 1]$. We present a worked derivation of the stochastic derivative of $X(p)$.

We construct a stochastic derivative (δ, w, Y) at input p satisfying Definition 2.2. Since $X(p)$ is parameterized via the inversion method, $\mathrm{d}X(\varepsilon) \in \{-1, 0, 1\}$ for small enough ε. Thus, for small enough ε, the conditional distribution of $\mathrm{d}X(\varepsilon)$ given $X(p) = x$ is:

$$\mathbb{P}\left(\mathrm{d}X(\varepsilon) = 1 \mid X(p) = x\right) = \frac{\mathbb{P}(X(p+\varepsilon) = x + 1 \text{ and } X(p) = x)}{\mathbb{P}(X(p) = x)}, \tag{A.1}$$

$$\mathbb{P}\left(\mathrm{d}X(\varepsilon) = -1 \mid X(p) = x\right) = \frac{\mathbb{P}(X(p+\varepsilon) = x - 1 \text{ and } X(p) = x)}{\mathbb{P}(X(p) = x)}, \tag{A.2}$$

and $\mathrm{d}X(\varepsilon) = 0$ otherwise. According to the inversion method, $X(p)(\omega) = x$ for ω between $\mathbb{P}(X(p) \leq x - 1)$ and $\mathbb{P}(X(p) \leq x)$. Taking the the derivative of the above quantities as $\varepsilon \to 0$ therefore yields:

$$w_+ := \lim_{\varepsilon \to 0} \frac{\mathbb{P}\left(\mathrm{d}X(\varepsilon) = 1 \mid X(p) = x\right)}{\varepsilon}, \tag{A.3}$$

$$= \frac{\frac{\mathrm{d}}{\mathrm{d}p}\mathbb{P}(X(p) \leq x)}{\mathbb{P}(X(p) = x)} \mathbf{1}\{\varepsilon \cdot \partial_p \mathbb{P}(X(p) \leq x) < 0\}, \tag{A.4}$$

and

$$w_- := \lim_{\varepsilon \to 0} \frac{\mathbb{P}\left(\mathrm{d}X(\varepsilon) = -1 \mid X(p) = x\right)}{\varepsilon}, \tag{A.5}$$

$$= \frac{\frac{\mathrm{d}}{\mathrm{d}p}\mathbb{P}(X(p) \leq x - 1)}{\mathbb{P}(X(p) = x)} \mathbf{1}\{\varepsilon \cdot \partial_p \mathbb{P}(X(p) \leq x - 1) > 0\}, \tag{A.6}$$

where the ε included on the right hand sides is an abuse of notation to express their dependence on the direction of the limit, and $\mathbf{1}\{C\} = \mathbf{1}_{\{C\}}$ for a condition C. By the differentiability of

Table 2: Stochastic derivatives of integer-valued stochastic programs

	Conditionally on $X(p) = x$	
$X(p)$	w_-	w_+
$\mathrm{Ber}(p)$	$-1/p \cdot \mathbf{1}\{x = 1 \text{ and } \varepsilon < 0\}$	$1/(1-p) \cdot \mathbf{1}\{x = 0 \text{ and } \varepsilon > 0\}$
$\mathrm{Bin}(n, p)$	$-x/p \cdot \mathbf{1}\{x > 0 \text{ and } \varepsilon < 0\}$	$(n-x)/(1-p) \cdot \mathbf{1}\{x < n \text{ and } \varepsilon > 0\}$
$\mathrm{Geo}(p)$	$x/\left(p(1-p)\right) \cdot \mathbf{1}\{x > 0 \text{ and } \varepsilon > 0\}$	$-(x+1)/p \cdot \mathbf{1}\{\varepsilon < 0\}$
$\mathrm{Pois}(p)$	$-x/p \cdot \mathbf{1}\{x > 0 \text{ and } \varepsilon < 0\}$	$\mathbf{1}\{\varepsilon > 0\}$

$\mathbb{P}(\mathrm{d}X(\varepsilon) \neq 0)$, $\mathrm{d}X(\varepsilon)/\varepsilon$ almost surely approaches 0, so $\delta = 0$. A simple choice for the bound B is 1: then $|\mathrm{d}X(\varepsilon)| > B\varepsilon = \varepsilon \iff \mathrm{d}X(\varepsilon) \in \{-1, 1\}$. We may now form w as the almost sure limit of $\mathbb{P}\left(A_B(\varepsilon) \mid X(p)\right)/\varepsilon = \mathbb{P}\left(\mathrm{d}X(\varepsilon) \neq 0 \mid X(p)\right)$ and Y as the limit of the conditional distribution of $X(p) + \mathrm{d}X(\varepsilon)$ given $X(p)$ and $\mathrm{d}X(\varepsilon) \neq 0$. By the above, w is given conditionally on $X(p) = x$ as

$$w = w_+ + w_-, \tag{A.7}$$

while the probability distribution of Y is given conditionally on $X(p) = x$ as

$$\mathbb{P}\left(Y = x + 1 \mid X(p) = x\right) = \frac{w_+}{w_+ + w_-}, \tag{A.8}$$

$$\mathbb{P}\left(Y = x - 1 \mid X(p) = x\right) = \frac{w_-}{w_+ + w_-}. \tag{A.9}$$

Table 2 lists the weights w_+ and w_- for the Bernoulli, Binomial, Poisson, and Geometric distributions, derived using the parameterizations given in Table 1. Using w_+ and w_-, we can easily express w and Y as above. Note that w_+ and w_- for a Binomial variable may be derived via representation as the sum of Bernoulli variables, with the primal evaluation $X(p) = x$ corresponding to the sum of x Bernoulli successes and $n - x$ Bernoulli failures.

The stochastic derivative for a Binomial variable $\mathrm{Bin}(n, p)$ has weight w of order n and Y deterministically either 1 or -1 depending on the chosen direction of the derivative. Therefore, derivative estimates produced by the derivative estimator $w(Y - X(p))$ have variance of order n. In contrast, the score method gives an estimator

$$\frac{X(p)(X(p) - np)}{p(1-p)} \tag{A.10}$$

which is also unbiased, but its variance

$$n^3 \frac{p}{1-p} + \mathcal{O}(n^2) \tag{A.11}$$

is growing with cubic rate in n.

Example A.2 (Differentiably parameterized continuous stochastic programs)**.** For continuous stochastic programs, our method reduces to the standard reparameterization trick. Specifically, for a differentiably parameterized continuous random variable, the stochastic derivative w.r.t a parameter p of the distribution has the form $(\delta, 0, 0)$. To satisfy Definition 2.2, one only has to select a bound B such that $|\mathrm{d}X(\varepsilon)| < B|\varepsilon|$ is satisfied almost surely: a natural choice is $B = |\delta| + 1$, so that $A_B^c(\varepsilon) = \{|\mathrm{d}X(\varepsilon)| \leq \varepsilon|\delta| + \varepsilon\}$ simply requires that the higher-order correction to the derivative linearization be smaller than ε when applied at a point at distance ε from p.

For instance, the stochastic derivative of $X(p) \sim \mathrm{Exp}(p)$ w.r.t. p, the scale of the Exponential distribution, is

$$\left(\frac{x}{p}, 0, 0\right) \tag{A.12}$$

conditionally on $X(p) = x$. Such rules are already implemented in the Julia AD ecosystem in `Distributions.jl` [1] and `DistributionsAD.jl` [2], and may be used without modification as the continuous case is a special case of our formalism.

Example A.3 (Categorical variable)**.** Consider a categorical variable $\text{Categorical}_{\mathbf{a}}(p_1, p_2, p_3, \ldots, p_n)$ assuming fixed outputs $\mathbf{a} = a_1, a_2, \ldots, a_n$ with probabilities $p_1, p_2, p_3, \ldots, p_n > 0$. Assume that $p_1, p_2, p_3, \ldots, p_n$ implicitly depend on a single parameter p, to fit our definition of stochastic derivative w.r.t. a single parameter, and let $X(p) \sim \text{Categorical}_{\mathbf{a}}(p_1, p_2, p_3, \ldots, p_n)$.

The categorical variable is univariate and its discrete outputs do not depend on p, and hence its stochastic derivative is of the form $(0, w, Y)$ with $Y \in \{a_{x-1}, a_{x+1}\}$. Similar logic to Example A.1 yields

$$w \cdot \mathbb{P}(Y = a_{x+1}) = w_+ = \frac{|\sum_{i=1}^{x} \partial_p p_i|}{p_x} \cdot \mathbf{1}\left\{\varepsilon \cdot \sum_{i=1}^{x} \partial_p p_i < 0\right\}, \tag{A.13}$$

$$w \cdot \mathbb{P}(Y = a_{x-1}) = w_- = \frac{\left|\sum_{i=1}^{x-1} \partial_p p_i\right|}{p_x} \cdot \mathbf{1}\left\{\varepsilon \cdot \sum_{i=1}^{x-1} \partial_p p_i > 0\right\}. \tag{A.14}$$

In theory, $a_1, a_2, \ldots, a_n$ can be arbitrary discrete objects, such as arrays or strings, as the categorical variable may represent an intermediate value of the program that is ultimately converted into a number. (Interpreted strictly, our formalism requires an embedding into Euclidean space; one can imagine a trivial embedding $a_i \mapsto i$ which is then provided to a learned embedding.) In handling such cases, one must be careful to speak in terms of *alternate* outputs rather than perturbations to the output. Our formalism of stochastic derivatives can already accommodate this idea, as Y takes on the alternate values rather than the values of the perturbations.

We now consider some general stochastic programs. Their stochastic derivatives arise automatically from composition and need not be hand-derived, but we do so to illustrate Theorem 2.6.

Example A.4 (Bernoulli plus Exponential)**.** Consider the program,

$$X_1(p) \sim \text{Ber}(p) \tag{A.15}$$

$$X_2(p) \sim \text{Exp}(p) \tag{A.16}$$

$$X(p) \sim X_1(p) + X_2(p) \sim \text{Ber}(p) + \text{Exp}(p). \tag{A.17}$$

Then, $X(p)$ has right (left) stochastic derivative,

$$\left(\frac{x_2}{p}, w_1, Y_1\right) \tag{A.18}$$

conditionally on $X_1(p) = x_1$ and $X_2(p) = x_2$, where w_1 and Y_1 are given conditionally on the Bernoulli output $X_1(p) = x_1$ for $\varepsilon > 0$ ($\varepsilon < 0$) by Table 2.

Example A.5 (Cubing a geometric)**.** Consider the program,

$$X_1(p) = \text{Geo}(p), \tag{A.19}$$

$$X(p) = X_1(p)^3. \tag{A.20}$$

We know from Example A.1 that $X_1(p)$ has right (left) stochastic derivative $(0, w_1, Y_1)$ with w_1 and Y_1 given by Table 2 for $\varepsilon > 0$ ($\varepsilon < 0$). In particular, $Y_1 \in \{x_1 - 1, x_1 + 1\}$ conditionally on $X_1(p) = x_1$. Thus, intuitively, when propagating through the cube function we only care about the discretely spaced alternate values $(x_1 - 1)^3$ and $(x_1 + 1)^3$; the conventional derivative $3x_1^2$ is irrelevant as the input to the cube function is integer-valued. Indeed, by Theorem 2.6 the stochastic derivative of $X(p)$ reads

$$(0, w_1, Y_1^3), \tag{A.21}$$

where $Y_1^3 \in \{(x_1 - 1)^3, (x_1 + 1)^3\}$.

Example A.6 (Parameter-scaled Bernoulli)**.** Consider the program,

$$X_1(p) = \text{Ber}(p), \tag{A.22}$$

$$X(p) = p \cdot X_1(p). \tag{A.23}$$

By Theorem 2.6, the stochastic derivative of $X(p)$ reads,

$$(X(p), w_1, pY_1), \tag{A.24}$$

where w_1 and Y_1 are given for $X_1(p)$ by Table 2.

Example A.7 (Two-step random walk)**.** Consider the program,

$$X_1(p) \sim \mathrm{Ber}(p), \tag{A.25}$$

$$X_2(x_1) \sim \begin{cases} \mathrm{Ber}(p) & \text{if } x_1 = 0 \\ \mathrm{Ber}(2p) & \text{if } x_1 = 1. \end{cases} \tag{A.26}$$

$$X(p) \sim X_1(p) + (X_2 \circ X_1)(p). \tag{A.27}$$

This represents a two-step random walk with steps given by $X_1(p)$ and $X_2(p)$, where there is a transition $0 \to 1$ with probability p, a transition $1 \to 2$ with probability $2p$, and self loops otherwise. In this example, let us focus on a particular primal evaluation where $X_1(p) = 0$ and $X_2(p) = 0$, so that $X(p) = 0$, and consider only the right stochastic derivative for simplicity. Conditionally on $X_1(p) = 0$, $X_1(p)$ has right stochastic derivative

$$\left(0, \frac{1}{1-p}, 1\right), \tag{A.28}$$

as given by Table 2. Now, conditionally on $X_2(p) = 0$, $X_2(0)$ has right stochastic derivative w.r.t. x_1,

$$\left(0, \frac{1}{1-p}, 1\right). \tag{A.29}$$

Let us now turn our attention to the stacked program $[X_1(p); (X_2 \circ X_1)(p)]$. By Theorem 2.6, the stacked program has right stochastic derivative $(0, w, Y_{12})$ where conditionally on $X_1(p) = 0$ and $X_2(0) = 0$,

$$Y_{12} \quad = \quad \begin{cases} [1; X_2(1)] & \text{with probability} \quad 1/2, \\ [0; 1] & \text{with probability} \quad 1/2, \end{cases} \tag{A.30}$$

and $w = \frac{2}{1-p}$. Note that conditionally on $X_2(0) = 0$, $X_2(1)$ has a chance $\frac{1-2p}{1-p}$ of also being 0, but a $\frac{p}{1-p}$ chance of flipping to 1, so that the first case expands into two events

$$Y_{12} \quad = \quad \begin{cases} [1; 1] & \text{with probability} \quad \frac{p}{2(1-p)}, \\ [1; 0] & \text{with probability} \quad \frac{1-2p}{2(1-p)}, \\ [0; 1] & \text{with probability} \quad 1/2. \end{cases} \tag{A.31}$$

Finally, the expression for $X(p)$ may be thought of as a unary sum function operating on this stacked program, with stochastic derivative $(0, w, Y)$ where conditionally on $X_1(p) = 0$ and $X_2(0) = 0$

$$Y \quad = \quad \begin{cases} 2 & \text{with probability} \quad \frac{p}{2(1-p)}, \\ 1 & \text{with probability} \quad \frac{1-2p}{2(1-p)} + 1/2. \end{cases} \tag{A.32}$$

Finally, we present an example using smoothed stochastic derivatives to rederive a popular gradient estimator.

Example A.8 (Recovering the straight-through gradient estimator of [6])**.** The straight-through gradient estimator formally assigns a derivative of 1 to the hard-thresholding function $\mathrm{HT} = \mathbf{1}_{[0,\infty)}$, such that a Bernoulli variable $\mathrm{Ber}(p) \sim \mathrm{HT}(p - U[0,1])$ is assigned a derivative of 1. Now, let $\widetilde{\delta}_L$ and $\widetilde{\delta}_R$ be the left and right smoothed stochastic derivatives of $\mathrm{Ber}(p)$. By Table 2 and Definition 2.7, for the right-sided ($\varepsilon > 0$) case,

$$\widetilde{\delta_R} = 1/(1-p) \cdot \mathbf{1}\{x = 0\}, \tag{A.33}$$

and for the left-sided ($\varepsilon < 0$) case,

$$\widetilde{\delta_L} = 1/p \cdot \mathbf{1}\{x = 1\}. \tag{A.34}$$

By linearity of expectation applied to Definition 2.7, any affine combination of $\widetilde{\delta_L}$ and $\widetilde{\delta_R}$ is also a valid smoothed stochastic derivative. In particular,

$$1 = (1-p) \cdot \widetilde{\delta}_L + p \cdot \widetilde{\delta}_R, \tag{A.35}$$

is a valid smoothed stochastic derivative of a Bernoulli variable, which explains why the straight-through estimator provides a low-bias estimate. Note that a constant-valued affine combination of the left and right smoothed stochastic derivatives is not possible in general, e.g. it is not possible for Geometric and Poisson random variables, in which cases smoothed stochastic derivatives generalize the straight-through estimator (mixing $\widetilde{\delta}_R$ and $\widetilde{\delta}_L$ may still be useful in these cases to reduce bias.)

B Proofs

B.1 Preliminaries

Recall,

Definition 2.1. A *stochastic program* $X(p)$ is a stochastic process with values in a Euclidean space E, whose index set I is either an open subset of a Euclidean space or a closed real interval.

Throughout the formalism, we let $X(p)$ denote such a stochastic program, where $I = [a, b] \subset \mathbb{R}$ is a closed interval. As noted in the main text, it is sufficient to consider this case because sensitivities of stochastic programs Z with more general index sets can be understood for an input $\mathbf{u}$ by studying at $p = 0$ the directional perturbation $X(p) = Z(\mathbf{u} + p\mathbf{v})$ in a direction $\mathbf{v}$, where $X(p)$ is then a stochastic program with index set a closed interval containing 0.

As in the main text, we use the shorthand

$$\mathrm{d}X(\varepsilon) = X(p + \varepsilon) - X(p). \tag{B.1}$$

Due to the asymmetry of stochastic derivatives, a number of statements and propositions have equivalent forms for right and left stochastic derivatives. In this case, we often use the notation $\varepsilon \to 0^{+/-}$ to indicate that statements and proofs for right and left stochastic derivatives can both be read off by choosing the appropriate side of the limit for all expressions, where objects such as w have different definitions depending on the chosen reading $\varepsilon \to 0^+$ or $\varepsilon \to 0^-$.

For clarity, we remark upon the (standard) notation,

$$W = \mathbb{E}\left[U \mid V\right], \tag{B.2}$$

where U and V are both random variables defined on a sample space Ω. In this setting, W is itself also a random variable defined on Ω. In particular, for a particular sample $\omega \in \Omega$, $W(\omega)$ is the conditional expectation $\mathbb{E}\left[U \mid V = V(\omega)\right]$. In the language of σ-algebras, $\mathbb{E}\left[U \mid V\right]$ is equivalent to $\mathbb{E}\left[U \mid \sigma(V)\right]$, where $\sigma(V)$ is the sub-σ-algebra generated by V.

B.2 Unbiasedness of stochastic derivatives

We first prove a general result regarding when a class of events $A(\varepsilon)$ allows for the pathwise gradient estimator δ to be applied when the probability space is restricted to $A^c(\varepsilon)$.

Proposition B.1. *Suppose $\mathrm{d}X(\varepsilon)/\varepsilon \to \delta$ almost surely, and $|\mathrm{d}X(\varepsilon)| \leq B\varepsilon$ holds given the event $A^c(\varepsilon)$, where B is integrable. Further, suppose $A(\varepsilon)$ almost surely does not hold as $\varepsilon \to 0^{+/-}$. Then, for any function $f\colon E \to \mathbb{R}$ with bounded derivative,*

$$\lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{A^c(\varepsilon)} \;\middle|\; X(p)\right] = \mathbb{E}\left[f'(X(p))\delta \mid X(p)\right]. \tag{B.3}$$

Proof. Since $\mathbf{1}_{A^c(\varepsilon)} \to 1$ and $\mathrm{d}X(\varepsilon)/\varepsilon \to \delta$ almost surely, it holds by the chain rule that

$$\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{A^c(\varepsilon)} \to f'(X(p))\delta. \tag{B.4}$$

almost surely as $\varepsilon \to 0^{+/-}$. Furthermore,

$$\left|\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{A^c(\varepsilon)}\right| \leq \frac{1}{\varepsilon}\|f'\|_\infty |\mathrm{d}X(\varepsilon)|\mathbf{1}_{A^c(\varepsilon)} \tag{B.5}$$

$$\leq B\|f'\|_\infty =: G \tag{B.6}$$

By assumption $\mathbb{E}\, G < \infty$. The proposition follows by the dominated convergence theorem applied to conditional expectations. □

As in the main text, define

$$A_B(\varepsilon) = \{|\mathrm{d}X(\varepsilon)| > B|\varepsilon|\}. \tag{B.7}$$

and recall

Definition 2.2 (Stochastic derivative)**.** Suppose $X(p) \in E$ is a stochastic program with index set I a closed real interval. We say that the triple of random variables (δ, w, Y), with weight $w \in \mathbb{R}$ and random $Y \in E$, is a right (left) *stochastic derivative* of X at the input $p \in I$ if $\mathrm{d}X(\varepsilon)/\varepsilon \to \delta$ almost surely as $\varepsilon \to 0$, and there is an integrable random bound $B > |\delta|$ such that for all bounded functions with bounded derivative $f\colon E \to \mathbb{R}$, it holds almost surely that

$$\mathbb{E}\left[w\left(f(Y) - f(X(p))\right) \mid X(p)\right] = \lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon} \mathbf{1}_{A_B(\varepsilon)} \,\middle|\, X(p)\right] \tag{2.4}$$

with limit taken from above (below), where $\mathbb{P}\left(A_B(\varepsilon) \mid X(p)\right)/\varepsilon$ is dominated in ε by an integrable random variable.

Using Proposition B.1, we may show that outside the event $A_B(\varepsilon)$, the sensitivity is well-described by the pathwise gradient estimator, allowing us to prove Proposition 2.3.

Proposition 2.3 (Unbiasedness)**.** If (δ, w, Y) is a stochastic derivative of $X(p)$ at p, it holds that

$$\frac{\mathrm{d}\mathbb{E}\left[X(p)\right]}{\mathrm{d}p} = \mathbb{E}[\delta + w(Y - X(p))]. \tag{2.5}$$

Proof. Let B be the associated bound. Since $A_B^c(\varepsilon)$ holds almost surely as $\varepsilon \to 0^{+/-}$ and $\mathrm{d}X(\varepsilon)/\varepsilon \leq B\varepsilon$ given $A_B^c(\varepsilon)$, Proposition B.1 applied to $A_B^c(\varepsilon)$ with f as identity implies

$$\lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{X(p+\varepsilon) - X(p)}{\varepsilon} \mathbf{1}_{A_B^c(\varepsilon)} \,\middle|\, X(p)\right] = \mathbb{E}\left[\delta \mid X(p)\right], \tag{B.8}$$

while Definition 2.2 also applied with f as identity gives

$$\lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{X(p+\varepsilon) - X(p)}{\varepsilon} \mathbf{1}_{A_B(\varepsilon)} \,\middle|\, X(p)\right] = \mathbb{E}\left[w(Y - X(p)) \mid X(p)\right]. \tag{B.9}$$

Summing Eq. (B.8) and Eq. (B.9), we have

$$\lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{X(p+\varepsilon) - X(p)}{\varepsilon} \,\middle|\, X(p)\right] = \mathbb{E}\left[\delta + w(Y - X(p)) \mid X(p)\right], \tag{B.10}$$

and applying the tower property of conditional expectations, we obtain

$$\frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p} = \lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{X(p+\varepsilon) - X(p)}{\varepsilon}\right] = \mathbb{E}[\delta + w(Y - X(p))] = \mathbb{E}[\widetilde{X}(p)], \tag{B.11}$$

as desired. □

B.3 Construction of stochastic derivatives for elementary programs

We now give a technical condition under which a stochastic derivative can be constructed for an elementary stochastic program, in which we characterize w as the derivative of the conditional probability of $A_B(\varepsilon)$ given $X(p)$, and Y as realization of the weak limit of the conditional distribution of $X(p+\varepsilon)$ given $X(p)$ and $A_B(\varepsilon)$.

Assumption B.2. We assume

- there exists an integrable δ such that $\mathrm{d}X(\varepsilon)/\varepsilon \to \delta$ almost surely as $\varepsilon \to 0$,

and that we may find an integrable random bound $B > |\delta|$ such that

- for the quantity
$$w(\varepsilon) = \mathbb{P}\left(A_B(\varepsilon) \mid X(p)\right), \tag{B.12}$$
$w(\varepsilon)/\varepsilon$ is dominated in ε by an integrable random variable and converges almost surely to a random variable w as $\varepsilon \to 0^{+/-}$.

- the conditional distribution of $X(p+\varepsilon)$ given $X(p)$ and $A_B(\varepsilon)$ converges in distribution to the law of a random variable Y as $\varepsilon \to 0^{+/-}$. Specifically, Y must satisfy

$$\mathbb{E}\left[w \cdot f(Y) \mid X(p)\right] = \lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[w \cdot f(X(p+\varepsilon)) \mid X(p), A_B(\varepsilon)\right]. \tag{B.13}$$

for all bounded continuous functions $f\colon E \to \mathbb{R}$.

Given Assumption B.2, we may construct a stochastic derivative for $X(p)$.

Theorem 2.4. Given a stochastic program $X(p)$ satisfying Assumption B.2 for all $p \in I = [a,b] \subset \mathbb{R}$, there exists a right stochastic derivative (δ, w_R, Y_R) with $w_R \geq 0$ at any $p < b$ and a left stochastic derivative (δ, w_L, Y_L) with $w_L \leq 0$ at any $p > a$.

Proof. Construct (δ, w, Y) via Assumption B.2 (taking the limit from above for the case of right stochastic derivatives, and the limit from below for the case of left stochastic derivatives). For bounded f with bounded derivative, write

$$\mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{A_B(\varepsilon)} \,\middle|\, X(p)\right] \tag{B.14}$$

$$= \mathbb{E}\left[w \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right)\frac{\mathbf{1}_{A_B(\varepsilon)}}{w(\varepsilon)} \,\middle|\, X(p)\right]$$

$$+ \mathbb{E}\left[\left(\frac{w(\varepsilon)}{\varepsilon} - w\right) \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right)\frac{\mathbf{1}_{A_B(\varepsilon)}}{w(\varepsilon)} \,\middle|\, X(p)\right]. \tag{B.15}$$

Note that $\mathbb{E}[\mathbf{1}_{A_B(\varepsilon)} \mid X(p)] = \mathbb{P}(A_B(\varepsilon) \mid X(p)) = w(\varepsilon)$. Thus, as $\varepsilon \to 0^{+/-}$ the first term approaches

$$\mathbb{E}\left[w \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right)\frac{\mathbf{1}_{A_B(\varepsilon)}}{w(\varepsilon)} \,\middle|\, X(p)\right] \tag{B.16}$$

$$= \mathbb{E}\left[w \cdot \left(f\left(X(p+\varepsilon)\right) - f(X(p))\right) \mid X(p), A_B(\varepsilon)\right] \tag{B.17}$$

$$\to \mathbb{E}\left[w \cdot \left(f(Y) - f(X(p))\right) \mid X(p)\right] \tag{B.18}$$

by Assumption B.2. Now, using the $X(p)$-measurability of $w(\varepsilon)$ and w, we may bound the magnitude of the second term as

$$\left|\mathbb{E}\left[\left(\frac{w(\varepsilon)}{\varepsilon} - w\right) \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right)\frac{\mathbf{1}_{A_B(\varepsilon)}}{w(\varepsilon)} \,\middle|\, X(p)\right]\right| \tag{B.19}$$

$$\leq 2\|f\|_\infty \cdot \left|\frac{w(\varepsilon)}{\varepsilon} - w\right| \cdot \mathbb{E}\left[\frac{\mathbf{1}_{A_B(\varepsilon)}}{w(\varepsilon)} \,\middle|\, X(p)\right] \tag{B.20}$$

$$= 2\|f\|_\infty \cdot \left|\frac{w(\varepsilon)}{\varepsilon} - w\right|, \tag{B.21}$$

which approaches 0 almost surely as $\varepsilon \to 0^{+/-}$ by the almost sure convergence of $w(\varepsilon)/\varepsilon$ to w. Thus,

$$\mathbb{E}\left[w \cdot \left(f(Y) - f(X(p))\right) \mid X(p)\right] = \lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{A_B(\varepsilon)} \,\middle|\, X(p)\right]. \tag{B.22}$$

Additionally, $\mathbb{P}\left(A_B(\varepsilon) \mid X(p)\right)/\varepsilon$ is dominated in ε by an integrable random variable and $\mathrm{d}X(\varepsilon)/\varepsilon \to \delta$ almost surely by assumption. So (δ, w, Y) is a stochastic derivative of X at p. Finally, note that $w(\varepsilon)/\varepsilon \geq 0$ for $\varepsilon \geq 0$ while $w(\varepsilon)/\varepsilon \leq 0$ for $\varepsilon \leq 0$, by the non-negativity of $w(\varepsilon)$. So applying Assumption B.2 with limits from above indeed produces a right stochastic derivative, while applying Assumption B.2 with limits from below produces a left stochastic derivative. □

B.4 Composition of stochastic derivatives

We now provide a proof of the chain rule for stochastic derivatives.

Theorem 2.6 (Chain rule)**.** Consider independent stochastic programs X_1 and X_2 and their composition $X_2 \circ X_1$. Suppose that X_1 has a right (left) stochastic derivative at $p \in \mathbb{R}$ given by (δ_1, w_1, Y_1) with bound B_1, and X_2 has a right stochastic derivative (δ_2, w_2, Y_2) in direction $\hat{\delta}_1 = \delta_1/|\delta_1|$ with bound B_2 given conditionally on its input $X_1(p)$, where with $\mathrm{d}X_2(\mathbf{v}) = X_2(X_1(p)+\mathbf{v}) - X_2(X_1(p))$ the event

$$\widetilde{A}_2(\varepsilon) = \{|\mathrm{d}X_2(\mathbf{v})| > B_2|\mathbf{v}| \text{ for some } \mathbf{v} \text{ satisfying } |\mathbf{v}| \leq \varepsilon\} \tag{B.23}$$

of a jump in some input direction $\mathbf{v}$ has $\mathbb{P}\left(\widetilde{A}_2(\varepsilon) \;\middle|\; X_1(p), X_2(p)\right)/\varepsilon$ dominated in ε by an integrable random variable W. Then, if $B_1(B_2 + W)$ is integrable, the stacked program $[X_1; X_2 \circ X_1]$ has a right (left) stochastic derivative at p given by (δ, w, Y) where $\delta = [\delta_1; |\delta_1|\delta_2]$,

$$Y \quad = \quad \begin{cases} [Y_1; X_2(Y_1)] & \text{with probability} \quad \dfrac{w_1}{w_1 + |\delta_1| \cdot w_2}, \\ [X_1(p); Y_2] & \text{with probability} \quad \dfrac{|\delta_1| \cdot w_2}{w_1 + |\delta_1| \cdot w_2}, \end{cases} \tag{2.7}$$

and $w = w_1 + |\delta_1| \cdot w_2$, with associated bound $B = B_1 + B_1 B_2$.

Proof. We prove the case of right stochastic derivatives, as the case of left stochastic derivatives is analogous. Let $X = X_2 \circ X_1$ as a shorthand. Now, for $\varepsilon \geq 0$ consider the events

$$A_1(\varepsilon) = \{|\mathrm{d}X_1(\varepsilon)| > B_1\varepsilon\}, \tag{B.24}$$
$$A_2(\varepsilon) = \{|\mathrm{d}X_2(\varepsilon\delta_1)| > B_2\varepsilon|\delta_1|\}, \tag{B.25}$$
$$A(\varepsilon) = \{|\mathrm{d}X(\varepsilon)| > B_1 B_2 \varepsilon\}. \tag{B.26}$$

Intuitively, the events $A_1(\varepsilon)$, $A_2(\varepsilon)$, and $A(\varepsilon)$ represent large jumps in X_1, X_2, and X respectively. Throughout the proof, we let f denote a bounded function with bounded derivative.

<u>Step 1: No large jump in X_1 or X_2 implies no large jump in X.</u>

Given $A_1^c(\varepsilon)$ and $\widetilde{A}_2^c(\varepsilon|\delta_1|)$, we have

$$|\mathrm{d}X(\varepsilon)| = |\mathrm{d}X_2(\mathrm{d}X_1(\varepsilon))| \tag{B.27}$$
$$\leq B_2 B_1 \varepsilon, \tag{B.28}$$

which implies $A^c(\varepsilon)$ (no jump in output). We now remark that $\widetilde{A}_2(\varepsilon|\delta_1|)$ and $A_2(\varepsilon)$ may be interchanged when considering sensitivities. Specifically,

$$\lim_{\varepsilon \to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \varepsilon\delta_1)) - f(X(p))}{\varepsilon} \mathbf{1}_{A_2^c(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.29}$$
$$= \lim_{\varepsilon \to 0^+} \mathbb{E}\left[f'(X_2(X_1(p)))\delta_1 \mid X_1(p), X(p)\right] \tag{B.30}$$
$$= \lim_{\varepsilon \to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \varepsilon\delta_1))) - f(X(p))}{\varepsilon} \mathbf{1}_{\widetilde{A}_2^c(\varepsilon|\delta_1|)} \;\middle|\; X_1(p), X(p)\right] \tag{B.31}$$

by applying the dominated convergence argument of Proposition B.1 for the directional perturbation $\varepsilon \mapsto X_2(X_1(p) + \varepsilon\delta_1)$ to both of the event classes $A_2(\varepsilon)$ and $\widetilde{A}_2(\varepsilon|\delta_1|)$, noting that both $A_2(\varepsilon)$ and $\widetilde{A}_2(\varepsilon|\delta_1|)$ almost surely do not hold as $\varepsilon \to 0$ and that $\widetilde{A}_2^c(\varepsilon|\delta_1|) \implies A_2^c(\varepsilon)$.

<u>Step 2: A large jump in both X_1 and X_2 has negligible probability.</u>

Note that $A_1(\varepsilon)$ and $\widetilde{A}_2(\varepsilon)$ are independent conditional on $X_1(p)$. Therefore, in the limit, we have

$$\frac{\mathbb{P}\left(A_1(\varepsilon) \cap \widetilde{A}_2(\varepsilon) \;\middle|\; X_1(p), X(p)\right)}{\varepsilon} \tag{B.32}$$
$$= \frac{\mathbb{P}\left(A_1(\varepsilon) \mid X_1(p)\right)}{\varepsilon} \mathbb{P}\left(\widetilde{A}_2(\varepsilon) \;\middle|\; X_1(p), X(p)\right) \tag{B.33}$$
$$\to 0 \tag{B.34}$$

almost surely as $\varepsilon \to 0^+$ by the domination of $\mathbb{P}\left(A_1(\varepsilon) \mid X_1(p)\right)/\varepsilon$ by an integrable random variable and that $\mathbb{P}\left(\widetilde{A}_2(\varepsilon) \;\middle|\; X_1(p), X(p)\right) \to 0$. By boundedness of f this implies

$$\mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{A_1(\varepsilon)\cap A_2(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] = 0, \tag{B.35}$$

so the sensitivity of X is negligible under $A_1(\varepsilon) \cap A_2(\varepsilon)$.

Step 3: Sensitivity of $X(p)$ given a large jump in X_1 is described by w_1, Y_1.

The random variable

$$Z(y) = \mathbb{E}[f(X_2(y)) \mid X_1(p), X(p)] \tag{B.36}$$

is $X_1(p)$, $X(p)$ measurable and almost surely continuous, and the function f in the characterization Eq. (2.4) can be taken as bounded and almost surely continuous in $X_1(p)$ by an extension of the Portmanteau theorem [40]. It follows, as $X(p)$ and $X_1(p+\alpha)$ are independent given $X_1(p)$, that

$$\lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{Z(X_1(p+\varepsilon)) - Z(X_1(p))}{\varepsilon}\mathbf{1}_{A_1(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.37}$$

$$= \mathbb{E}\left[w_1 \cdot (Z(Y_1) - Z(X_1(p)) \mid X_1(p), X_2(p)\right], \tag{B.38}$$

so that

$$\lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{A_1(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.39}$$

$$= \mathbb{E}\left[w_1 \cdot f(X_2(Y_1)) - f(X(p)) \mid X_1(p), X(p)\right]. \tag{B.40}$$

Step 4: Sensitivity of $X(p)$ given a large jump in X_2 is described by w_2, Y_2.

Using Eq. (B.27), given $\widetilde{A}_2^c(\varepsilon|\delta_1|) \cap A_1^c(\varepsilon)$ it holds that

$$\left|\frac{f\left(X_2(X_1(p) + \mathrm{d}X_1(\varepsilon))\right) - f(X_2(p))}{\varepsilon}\right| \le \|f'\|_\infty \frac{|\mathrm{d}X(\varepsilon)|}{\varepsilon} \le \|f'\|_\infty B_1 B_2 \tag{B.41}$$

is dominated by an integrable random variable. Therefore, since $\mathrm{d}X_1(\varepsilon)/\varepsilon \to \delta_1$ almost surely,

$$\lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \mathrm{d}X(\varepsilon))) - f(X(p))}{\varepsilon}\mathbf{1}_{\widetilde{A}_2^c(\varepsilon|\delta_1|)\cap A_1^c(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.42}$$

$$= \lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \varepsilon\delta_1)) - f(X(p))}{\varepsilon}\mathbf{1}_{\widetilde{A}_2^c(\varepsilon|\delta_1|)\cap A_1^c(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.43}$$

by dominated convergence. Putting this together with Eq. (B.31), by definition of w_2 and Y_2, for any bounded f with bounded derivative we have

$$\mathbb{E}\left[|\delta_1| w_2 \left(f(Y_2) - f\big(X(p)\big)\right) \mid X_1(p), X(p)\right] \tag{B.44}$$

$$= \lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \varepsilon\delta_1)) - f(X(p))}{\varepsilon}\mathbf{1}_{A_2(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.45}$$

$$= \lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \varepsilon\delta_1)) - f(X(p))}{\varepsilon}\mathbf{1}_{A_2(\varepsilon)\cap A_1^c(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.46}$$

$$= \lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \varepsilon\delta_1))) - f(X(p))}{\varepsilon}\mathbf{1}_{\widetilde{A}_2(\varepsilon|\delta_1|)\cap A_1^c(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.47}$$

$$= \lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X_2(X_1(p) + \mathrm{d}X_1(\varepsilon))) - f(X(p))}{\varepsilon}\mathbf{1}_{\widetilde{A}_2(\varepsilon|\delta_1|)\cap A_1^c(\varepsilon)} \;\middle|\; X_1(p), X(p)\right] \tag{B.48}$$

$$= \lim_{\varepsilon\to 0^+} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\mathbf{1}_{\widetilde{A}_2(\varepsilon|\delta_1|)} \;\middle|\; X_1(p), X(p)\right], \tag{B.49}$$

where we freely neglect (or add back in) $\mathbf{1}_{A_1(\varepsilon)\cap A_2(\varepsilon)}$ inside the conditional expectation by Eq. (B.35).

Step 5: (δ, w, Y) is a stochastic derivative of the stacked program $[X_1(p); X(p)]$.

By the almost sure convergences $\mathrm{d}X_1(\varepsilon)/\varepsilon \to \delta_1$ and $\mathrm{d}\widehat{X}_2(\varepsilon)/\varepsilon \to \delta_2$, the chain rule implies $\mathrm{d}X(\varepsilon)/\varepsilon \to |\delta_1|\delta_2$ almost surely, where $|\delta_1|\delta_2 \leq B_1 B_2$ is integrable by assumption. So $\delta = [\delta_1; |\delta_1|\delta_2]$ is an almost sure derivative of $[X_1(p); X(p)]$.

Given $A(\varepsilon)$, $|\mathrm{d}X(\varepsilon)| > B_1 B_2 \varepsilon$, and given $A_1(\varepsilon)$, $|\mathrm{d}X_1(\varepsilon)| > B_1\varepsilon$. So we can choose the bound $B = B_1 B_2 + B_1$ for the stacked program, so that $A_B(\varepsilon) \subseteq A_1(\varepsilon) \cup A(\varepsilon)$.

Now, since $A(\varepsilon) \subseteq A_1(\varepsilon) \cup \widetilde{A}_2(\varepsilon|\delta_1|)$,

$$\frac{\mathbb{P}\left(A(\varepsilon) \mid X_1(p), X(p)\right)}{\varepsilon} \leq \frac{\mathbb{P}\left(A_1(\varepsilon) \mid X_1(p), X(p)\right)}{\varepsilon} + \frac{\mathbb{P}\left(A_2(\varepsilon|\delta_1|) \mid X_1(p), X(p)\right)}{\varepsilon} \tag{B.50}$$

$$\leq \frac{\mathbb{P}\left(A_1(\varepsilon) \mid X_1(p), X(p)\right)}{\varepsilon} + |\delta_1| W, \tag{B.51}$$

where $\mathbb{P}\left(A_1(\varepsilon) \mid X_1(p), X(p)\right)/\varepsilon$ and $|\delta_1|W$ are integrable by assumption. Therefore, $\mathbb{P}\left(A_B(\varepsilon) \mid X_1(p), X(p)\right)$ is also dominated by an integrable random variable, as desired.

Now, let f be operating on the stacked program, and for convenience write $f(x_1; x_2) = f([x_1; x_2])$. As a shorthand, let $\overline{X}(p) = [X_1(p); X(p)]$. With w and Y as given in the statement,

$$\mathbb{E}\left[w\big(f(Y) - f(X_1(p); X(p))\big) \;\middle|\; X_1(p), X(p)\right] \tag{B.52}$$

$$= \mathbb{E}\left[w_1\big(f\left(X_1(p); X_2(Y_1)\right) - f(X_1(p); X(p))\big) \;\middle|\; X_1(p), X(p)\right]$$

$$+ \mathbb{E}\left[|\delta_1| w_2\big(f(Y_1; X_2(Y_1)) - f(X_1(p); X(p))\big) \;\middle|\; X_1(p), X(p)\right] \tag{B.53}$$

$$= \lim_{\varepsilon \to 0^+} \mathbb{E}\left[\frac{f(\overline{X}(p+\varepsilon)) - f(\overline{X}(p))}{\varepsilon}\left(\mathbf{1}_{A_1(\varepsilon)} + \mathbf{1}_{\widetilde{A}_2(\varepsilon|\delta_1|)}\right) \;\middle|\; X_1(p), X(p)\right] \tag{B.54}$$

$$= \lim_{\varepsilon \to 0^+} \mathbb{E}\left[\frac{f(\overline{X}(p+\varepsilon)) - f(\overline{X}(p))}{\varepsilon}\mathbf{1}_{A_1(\varepsilon) \cup \widetilde{A}_2(\varepsilon|\delta_1|)} \;\middle|\; X_1(p), X(p)\right] \tag{B.55}$$

$$= \lim_{\varepsilon \to 0^+} \mathbb{E}\left[\frac{f(\overline{X}(p+\varepsilon)) - f(\overline{X}(p))}{\varepsilon}\mathbf{1}_{A(\varepsilon)} \;\middle|\; X_1(p), X(p)\right]. \tag{B.56}$$

We conclude that (δ, w, Y) is a valid stochastic derivative of $\overline{X}(p) = [X_1(p); X(p)]$.

□

B.5 Unbiasedness of pruning strategy

In the main text, we note that we can employ a pruning strategy so that we only ever track one alternative path (i.e. one sample from the stochastic derivative component Y) and yet still obtain an unbiased estimate. The following shows that the pruning method, whereby one chooses between two samples of Y by picking one with probability proportional to its weight, is indeed unbiased.

We prove by induction that the currently tracked alternative path is an unbiased choice amongst all possible alternative paths seen so far. The base case, where the first alternative path observed is followed, is trivial as it is the only choice. Now, suppose that n alternative branches have been observed so far, where $w = w_1 + w_2 + \cdots + w_n$ is the summed weight so far. Suppose we observe an $(n+1)$th branch, with weight w_{n+1}. By our pruning strategy it is chosen with probability

$$\frac{w_{n+1}}{w_{n+1} + w} = \frac{w_{n+1}}{w_1 + w_2 + \cdots + w_{n+1}}, \tag{B.57}$$

while the jth path for some $j \leq n$ will have been chosen after this step with probability

$$\frac{w_j}{w_1 + w_2 + \cdots + w_n} \cdot \frac{w_1 + w_2 + \cdots + w_n}{w_1 + w_2 + \cdots + w_{n+1}} = \frac{w_j}{w_1 + w_2 + \cdots + w_{n+1}}, \tag{B.58}$$

as desired for unbiasedness. Note that this enables us to prune *online*, i.e. without knowing the full structure of the computation a priori, which we exploit in `StochasticAD.jl`.

B.6 Smoothing

Recall,

Definition 2.7 (Smoothed stochastic derivative)**.** For a stochastic program $X(p)$ with a right (left) stochastic derivative (δ, w, Y) at input p, a right (left) smoothed stochastic derivative $\widetilde{\delta}$ of X at input p is given as

$$\widetilde{\delta} = \mathbb{E}\left[\delta + w(Y - X(p)) \mid X(p)\right]. \tag{2.8}$$

Using Definition 2.2, we can write the above in an alternative form that does not rely on the definition of the stochastic derivative,

$$\widetilde{\delta} = \lim_{\varepsilon \to 0^{+/-}} \frac{\mathbb{E}\left[\mathrm{d}X(\varepsilon) \mid X(p)\right]}{\varepsilon}, \tag{B.59}$$

as given in [21, 22]. Smoothed stochastic derivatives propagate through functions that are *locally* linear over the range of Y conditionally on $X(p)$, as we now formalize. Note that this is a much weaker requirement than global linearity over the full range of $X(p)$, as the range of Y conditional on $X(p)$ is generally more restricted (e.g. $Y \in \{X(p) - 1, X(p) + 1\}$ for a binomial variable).

Proposition B.3. *Suppose that f is linear over the range of Y conditionally on $X(p) = x$, for all x. Then,*

$$\frac{\mathrm{d}}{\mathrm{d}p}\mathbb{E}f(X(p)) = \mathbb{E}\left[f'(X(p))\widetilde{\delta}\right]. \tag{B.60}$$

Proof. Note that $(f'(X(p)) \cdot \delta, w, f(Y))$ is a stochastic derivative of $f \circ X$ by Theorem 2.6. Now, by Proposition 2.3, for such an f we have the simplification,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}p}\mathbb{E}\left[f(X(p))\right] &= \mathbb{E}\left[f'(X(p)) \cdot \delta + w(f(Y) - f(X(p)))\right] && \text{(B.61)}\\
&= \mathbb{E}\left[f'(X(p))\left(\delta + w \cdot (Y - X(p))\right)\right] && \text{(B.62)}\\
&= \mathbb{E}\left[f'(X(p))\widetilde{\delta}\right]. && \text{(B.63)}
\end{aligned}$$

□

In most cases, local linearity will only hold approximately, leading to bias in the estimate produced by propagating smoothed stochastic derivatives. However, in the case of a particle filter resampling step an exact estimate is produced, as we show in the following.

C Formalism of the particle filter

C.1 Hidden Markov model

Let us consider a hidden Markov model with random states $X_1, \ldots, X_n$ as specified by a stochastic program $X_1(\theta)$ giving the starting value depending on parameters θ and consecutive states given by pointwise differentiable stochastic programs $X_i(x_{i-1}, \theta)$ depending on the previous state $x_{i-1} \sim X_{i-1}$ (Markov property) and θ. In general, we allow continuous probability densities $p(x_1 \mid \theta)$ of X_1 and continuous transition probability densities $p(x_i \mid x_{i-1}, \theta)$ to depend arbitrarily on θ. This latent process $X_1, \ldots, X_n$ is indirectly observed as the process $y_1 \sim Y_1, \ldots, y_n \sim Y_n$ with n observations $Y_i(x_i)$ depending on $x_i \sim X_i$ which are specified to have smooth conditional probability densities $p(y_i \mid x_i, \theta)$ depending only on x_i and θ. As a concrete, special case, we consider the following linear Gaussian state-space model with a d-dimensional latent process,

$$X_i = \Phi X_{i-1} + \mathrm{Normal}(0, Q), \tag{C.1}$$

$$Y_i = X_i + \mathrm{Normal}(0, R), \tag{C.2}$$

where $Q = 0.02 \cdot \mathbb{1}_{d\times d}$, $R = 0.01 \cdot \mathbb{1}_{d\times d}$, $x_1 \sim \mathrm{Normal}(\mu, 0.001 \cdot \mathbb{1}_{d\times d})$, $\mu \sim \mathrm{Normal}(0, \mathbb{1}_{d\times d})$ is a random initial position, and Φ is a d-dimensional rotation matrix. Here, the parameters θ are defined by the entries of Φ, i.e. $\theta = \mathrm{vec}(\Phi)$. We use a particle filter to compute an estimate of the likelihood $\mathcal{L} = p(y_1, \ldots, y_n \mid \theta)$.

C.2 Differentiating a particle filter with a resampling step

Given n observations $y_1 \sim Y_1, \ldots, y_n \sim Y_n$ of the hidden Markov model defined via Eqs. (C.1) and (C.2), a bootstrap particle filter allows us to approximate the posterior distributions of the states and the likelihood $\mathcal{L}$ of these observations by propagating a cloud of K *weighted* particles. We denote the kth particle by $x_i^{(k)}$. Each particle evolves independently according to the stochastic program [Eqs. (C.1) and (C.2)] and carries a weight $\varpi_i^{(k)}$ measuring how well the trajectory of the particle so far is matching the observations. This importance weight is updated by $\varpi_i^{(k)} = p(y_i \mid x_i^{(k)}, \theta)\varpi_{i-1}^{(k)}$, such that the empirical measure $\sum_k \varpi_i^{(k)} \delta_{x_i^{(k)}} / \sum_k \varpi_i^{(k)}$ of the weighted particles approximates the filtering distribution $p(x_i \mid y_1, \ldots, y_i, \theta)$.

So far, the particles and weight trajectories are differentiable with respect to the parameter [3]. However, weight degeneracy, the collapse of all but a few weights, is a common problem. Our goal is to discard unlikely particles, so that numerical resources are not wasted on particles with vanishing weight. The strategy to accomplish this goal is to include resampling steps, where we pick the particles that best match the observations. However, such resampling steps present discrete randomness, where the particle population is resampled according to the particle weights $\varpi_i^{(k)}$ to form a new population $(x')_i^{(k)}$ with equal weight $\varpi_i' = 1/K \cdot \sum_{k=1}^K \varpi_i^{(k)}$. To differentiate the resampling step, we need to describe how perturbing the weight distribution $\varpi_i^{(k)}$ provided to the resampling procedure affects the resampled particles and weights. Importantly, the marginal likelihood of a parameter θ can be read off the weights at the last step

$$p(y_1, \ldots, y_n \mid \theta) \approx \sum_{k=1}^{K} \varpi_n^{(k)}. \tag{C.3}$$

The strategies for resampling vary, but all strategies have in common that the *marginal* distribution of each resampled particle $x_i'^{(k)}$ is a multinomial distribution over the original particles with weights given by the normalized weight vector $(\varpi_i^{(1)}/\varpi_i, \ldots, \varpi_i^{(K)}/\varpi_i)$ with $\varpi_i = \sum_{k=1}^K \varpi_i^{(k)}$. Thus, a weighted sample of the marginal distribution of each resampled particle can be obtained by repeating the following procedure until obtaining a weight of ϖ_i:

1. Sample an integer k uniformly from 1 to K.
2. Return the particle $x_i^{(k)}$ with assigned weight $\varpi_i \operatorname{Ber}(\varpi_i^{(k)}/\varpi_i)$.

Conditional on the assigned weight being ϖ_i, the returned particle obeys the multinomial distribution. The key insight is that the alternate possibility of a resampled particle $x_i^{(k)}$ not being chosen can be written instead as the alternate possibility of its weight $\varpi_i^{(k)}$ changing to 0.

Recall from Example 2.5 that there is an asymmetry between the left and right stochastic derivative of the Bernoulli distribution $\operatorname{Ber}(\varpi_i^{(k)}/\varpi)$ conditioned on the output 0 or 1 being chosen. In this case, it is convenient to take the *left* stochastic derivative $(0, w_i^{(k)}, Y_i^{(k)})$ with $Y_i^{(k)} = 0$ and $w_i^{(k)} = -\varpi_i^k/\varpi_i$ of the particle's weight ϖ_i because it is 0 when the assigned weight is 0, meaning that this case has no influence on the derivative estimate. Therefore, particles that are not resampled [and thus do not contribute to the primal value, cf. Eq. (C.3)] do (also) not contribute to the derivative computation. The fact that only the weights have a stochastic derivative, but not the particles imposes the following setting:

Let $X(p)$ be a stochastic program approximated by the program $\widetilde{X}(p)$. Assume we can sample from $\widetilde{X}(p)$. Assuming absolute continuity, we may write expectations as $\mathbb{E}\varpi(p) f(\widetilde{X}(p)) = \mathbb{E} f(X(p))$ where $\varpi(p)$ is the Radon-Nikodym derivative [4] of the law of $\widetilde{X}(p)$ with respect to the law of $X(p)$, evaluated in $\widetilde{X}(p)$. We thus consider the program which returns the pair of weight and value

$$(\varpi(p), \widetilde{X}(p)). \tag{C.4}$$

Proposition C.1. *If $\widetilde{X}(p)$ is a continuous program, reparameterized such that it is differentiable pointwise, and $\varpi(p)$ has a smoothed stochastic derivative $\widetilde{\delta}$,*

$$\frac{\mathrm{d}}{\mathrm{d}p}\mathbb{E}f(X(p)) = \mathbb{E}\left[\varpi(p)f'(\widetilde{X}(p))\widetilde{X}'(p) + \widetilde{\delta}f(\widetilde{X}(p))\right]. \tag{C.5}$$

Proof. By assumption,

$$\mathbb{E}f(X(p)) = \mathbb{E}\varpi(p)f(\widetilde{X}(p)). \tag{C.6}$$

As the function $(\varpi, x) \mapsto \varpi f(x)$ is linear in ϖ, the statement follows from Proposition B.3. □

Therefore, we can replace the stochastic derivative of $\varpi\,\mathrm{Ber}(\varpi_i^{(k)}/\varpi)$ with its smoothed stochastic derivative $w_i^{(k)}(1 - Y_i^{(k)}) = \varpi_i^k/\varpi_i$ obtained using Table 2. This is convenient, as smoothed stochastic derivatives permit forward- and reverse-mode whereas reverse-mode AD becomes superior than forward-mode AD for functions f from $\mathbb{R}^n$ to $\mathbb{R}^m$ with $m \gg n$ [2]. Since the weights are used in a purely linear fashion, Proposition B.3 and Proposition C.1 guarantee that the derivative estimator is unbiased.

Numerically, we accomplish this in our code with a formally differentiable weight function `new_weight(p)` whose primal value is always 1 but whose derivative is the left smoothed stochastic derivative of $\mathrm{Ber}(p)$ for primal output 1,

$$\widetilde{\delta_L} = \frac{1}{p}, \tag{C.7}$$

so that a particle $x_i^{(k)}$ has weight given as

$$\varpi_i \cdot \texttt{new_weight}(\varpi_i^{(k)}/\varpi_i). \tag{C.8}$$

In [7], an equivalent expression is derived by different means, using the stop-gradient operator $\perp$:

$$\varpi_i \cdot \frac{(\varpi_i^{(k)}/\varpi_i)}{\perp(\varpi_i^{(k)}/\varpi_i)}, \tag{C.9}$$

where $\perp$ is formally assigned a derivative of 0.

D Implementation details

D.1 Experiment details

All computation times in Fig. 5 were measured on an Intel Xeon Platinum 8260 CPU and Julia version 1.6. Garbage collection times are included in the total run time. We provide code and instructions to run the examples in the tutorials folder of `StochasticAD.jl`.

D.2 Software dependencies

`StochasticAD.jl` is implemented in the Julia Language [4] and uses internally `Distributions.jl` and `DistributionsAD.jl` [2, 1], `ChainRulesCore.jl` [5], and `ForwardDiff.jl` [6].

The presented examples are using the additional packages: `BenchmarkTools.jl` [7], `Zygote.jl` [8], `GaussianDistributions.jl` [9], and `Plots.jl` [10].

We refer the reader to the documentation of `StochasticAD.jl` for more detailed information on the package implementation.

E Funding details

This material is based upon work supported by the National Science Foundation under grant no. OAC-1835443, grant no. SII-2029670, grant no. ECCS-2029670, grant no. OAC-2103804, and grant no. PHY-2021825. We also gratefully acknowledge the U.S. Agency for International Development through Penn State for grant no. S002283-USAID. The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001211 and DE-AR0001222. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No HR00112290091. We also gratefully acknowledge the U.S. Agency for International Development through Penn State for grant no. S002283-USAID. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. This material was supported by The Research Council of Norway and Equinor ASA through Research Council project "308817 - Digital wells for optimal production and drainage". Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. We also acknowledge financial support from the NCCR QSIT funded by the Swiss National Science Foundation (Grant No. 51NF40-185902), as well as from the Swiss National Science Foundation individual grant (Grant No. 200020_200481).

Supplemental References

[SI1] Dahua Lin, John Myles White, Simon Byrne, Douglas Bates, Andreas Noack, John Pearson, Alex Arslan, Kevin Squire, David Anthoff, Theodore Papamarkou, Mathieu Besançon, Jan Drugowitsch, Moritz Schauer, and contributors. JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions, July 2019. URL https://doi.org/10.5281/zenodo.2647458.

[SI2] Mathieu Besançon, Theodore Papamarkou, David Anthoff, Alex Arslan, Simon Byrne, Dahua Lin, and John Pearson. Distributions.jl: Definition and modeling of probability distributions in the juliastats ecosystem. *Journal of Statistical Software*, 98(16):1–30, 2021.

[SI3] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv:1805.11122*, 2018.

[SI4] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[SI5] Frames Catherine White, Michael Abbott, Miha Zgubic, Jarrett Revels, Alex Arslan, Seth Axen, Simeon Schaub, Nick Robinson, Yingbo Ma, Gaurav Dhingra, Will Tebbutt, Niklas Heim, David Widmann, Andrew David Werner Rosemberg, Niklas Schmitz, Christopher Rackauckas, Rainer Heintzmann, Frank Schäfer, Carlo Lucibello, Keno Fischer, Alex Robson, Jerry Ling, Matt Brzezinski, Andrei Zhabinski, Daniel Wennberg, Mathieu Besançon, Pietro Vertechi, Shashi Gowda, and Andrew Fitzgibbon. JuliaDiff/ChainRules.jl: v1.35.0, May 2022. URL https://doi.org/10.5281/zenodo.6574605.

[SI6] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[SI7] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments. *arXiv:1608.04295*, 2016.

[SI8] Michael Innes. Don't unroll adjoint: Differentiating SSA-form programs. *arXiv:1810.07951*, 2018.

[SI9] Moritz Schauer and contributors. GaussianDistributions.jl, 2018. URL https://github.com/mschauer/GaussianDistributions.jl.

[SI10] Simon Christ, Daniel Schwabeneder, and Christopher Rackauckas. Plots.jl–a user extendable plotting api for the julia programming language. *arXiv:2204.08775*, 2022.