Semantics of Integrating and Differentiating Singularities

JESSE MICHEL, Massachusetts Institute of Technology, USA
WONYEOL LEE*, POSTECH, Republic of Korea
HONGSEOK YANG, KAIST, Republic of Korea

A *singular function* is a partial function such that at one or more points, the left and/or right limit diverge (e.g., the function $1/x$). Since programming languages typically support division, programs may denote singular functions. Although on its own, a singularity may be considered a bug, introducing a division-by-zero error, *singular integrals*—a version of the integral that is well-defined when the integrand is a singular function and the domain of integration contains a singularity—arise in science and engineering, including in physics, aerodynamics, mechanical engineering, and computer graphics.

In this paper, we present the first semantics of a programming language for singular integration. Our differentiable programming language, SingularFlow, supports the evaluation and differentiation of singular integrals. We formally define the denotational semantics of SingularFlow, deriving all the necessary mathematical machinery so that this work is rigorous and self-contained. We then define an operational semantics for SingularFlow that estimates integrals and their derivatives using Monte Carlo samples, and show that the operational semantics is a well-behaved estimator for the denotational semantics.

We implement SingularFlow in JAX and evaluate the implementation on a suite of benchmarks that perform the *finite Hilbert transform*, an integral transform related to the Fourier transform, which arises in domains such as physics and electrical engineering. We then use SingularFlow to approximate the solutions to four *singular integral equations*—equations where the unknown function is in the integrand of a singular integral—arising in aerodynamics and mechanical engineering.

CCS Concepts: • **Theory of computation** → *Denotational semantics*; *Operational semantics*; • **Mathematics of computing** → *Probabilistic algorithms*; *Automatic differentiation*; Integral equations; Functional analysis.

Additional Key Words and Phrases: Singular Integrals, Distribution Theory, Differentiable Programming, Probabilistic Programming, Machine Learning, Physically Informed Neural Networks, Science and Engineering

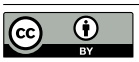
ACM Reference Format:
Jesse Michel, Wonyeol Lee, and Hongseok Yang. 2025. Semantics of Integrating and Differentiating Singularities. *Proc. ACM Program. Lang.* 9, PLDI, Article 164 (June 2025), 26 pages. https://doi.org/10.1145/3729263

1 Introduction

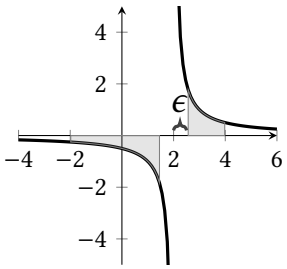
On its own, a program denoting a *singular function* can reasonably be thought of as having a bug. For example, a program may denote a singular function that takes in a real number u and returns $1/(u-2)$. An implementation of this program would typically return a division-by-zero error at $u = 2$, and the denoted function diverges as u approaches 2 (Figure 1a). More formally, a singular function is a partial function $f : \mathbb{R}^d \rightharpoonup \mathbb{R}$ such that there is a point $u_0 \in \mathbb{R}^d$ where $f(u)$ diverges

*Corresponding author.

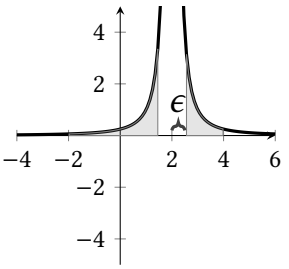
Authors' Contact Information: Jesse Michel, Massachusetts Institute of Technology, Cambridge, USA, jmmichel@mit.edu; Wonyeol Lee, POSTECH, Pohang, Republic of Korea, wonyeol.lee@postech.ac.kr; Hongseok Yang, KAIST, School of Computing, Daejeon, Republic of Korea, hongseok.yang@kaist.ac.kr.



ACM 2475-1421/2025/6-ART164
https://doi.org/10.1145/3729263



(a) The graph of $\frac{1}{u-2}$



(b) The graph of $\frac{1}{(u-2)^2}$

Fig. 1. The graphs above depict singular functions with a singularity at $s = 2$. The Cauchy principal value integral (a) is defined as the integral where the domain of integration approaches the singularity: $C\int_{-2}^{4} \frac{1}{u-s} du \triangleq \lim_{\epsilon \to 0^+} \left(\int_{-2}^{s-\epsilon} \frac{1}{u-s} du + \int_{s+\epsilon}^{4} \frac{1}{u-s} du \right)$. The Hadamard finite part integral (b) is defined in terms of the derivative of the Cauchy principal value integral: $\mathcal{H}\int_{-2}^{4} \frac{1}{(u-s)^2} du \triangleq \frac{d}{ds} \left(C\int_{-2}^{4} \frac{1}{u-s} du \right)$.

in the limit as $u \to u_0$ and $f(u_0)$ does not exist as a result (and the rate of divergence should not exceed $1/\|u - u_0\|_2^n$ for some integer $n > 0$).[1]

Many problems arising in diverse areas of science and engineering are specified using *singular integrals*—a version of the integral that is well-defined when the integrand is a singular function and the domain of integration contains one or more singularities. For instance, in mechanical engineering, the growth of a crack on a structure is modeled using a singular integral over the stress on the crack surface [64]. In aerodynamics, the upward force on a wing is modeled by a singular integral over pressure perpendicular to the wing [40]. In computer graphics, researchers can simulate a system on an infinite domain by transforming the problem to a finite domain with singularities and then computing singular integrals to perform the simulation [65].

The definite Riemann (or Lebesgue) integral over a region containing a singularity does not exist because the integrand diverges to $\pm\infty$ or an undefined value around the singularity. For example, Figure 1a depicts a singular function $g(u) = 1/(u-2)$, which has a singularity at $s = 2$. The Riemann integral from -2 to 4 of g is undefined because g approaches $-\infty$ as u approaches 2 from the left and $+\infty$ as u approaches 2 from the right. However, if $\epsilon > 0$ is a positive number approaching zero, then the integral from -2 to 4 excluding the ϵ ball around $s = 2$ (the shaded region in Figure 1a) approaches the *Cauchy principal value integral* from -2 to 4 of g. More generally, if s is closer to 4 than -2 (i.e., $1 < s < 4$), then only the region from -2 to $-4 + 2s$ remains after the positive and negative parts in a symmetric region around the singularity cancel each other out. Concretely,

$$C\int_{-2}^{4} \frac{1}{u-s} du \triangleq \lim_{\epsilon \to 0^+} \left(\int_{-2}^{s-\epsilon} \frac{1}{u-s} du + \int_{s+\epsilon}^{4} \frac{1}{u-s} du \right) = \int_{-2}^{-4+2s} \frac{1}{u-s} du = \ln\left(\frac{4-s}{s+2}\right),$$

where the notation C indicates that the integral is interpreted a Cauchy principal value integral. At $s = 2$, we find that $C\int_{-2}^{4} \frac{1}{u-s} du = -\ln 2$.

In the more general case, if $f : \mathbb{R} \to \mathbb{R}$ is a smooth function and $a < s < b$ bounds the location of the singularity, then the following Cauchy principal value integral is well-defined:

$$C\int_{a}^{b} \frac{f(u)}{u-s} du \triangleq \lim_{\epsilon \to 0^+} \left(\int_{a}^{s-\epsilon} \frac{f(u)}{u-s} du + \int_{s+\epsilon}^{b} \frac{f(u)}{u-s} du \right).$$

To better understand the singularity, suppose f is positive. Then, the integrand is a singular function with a singularity at s that diverges to $-\infty$ (or $+\infty$) as u approaches s from the left (or right).

[1] Gelfand and Shilov [29, Chapter 3] call this an *algebraic singularity*. Distribution theoretic techniques apply to this case, while other techniques do not (e.g., dimensional regularization).

Analogously, the Riemann integral from -2 to 4 of the function $g(u) = 1/(u-2)^2$ depicted in Figure 1b diverges to ∞, but the *Hadamard finite part integral* is finite and equals the derivative with respect to s of the Cauchy principal value integral of $1/(u-s)$ evaluated at $s = 2$. Concretely,

$$\mathcal{H}\int_{-2}^{4} \frac{1}{(u-s)^2} du \triangleq \frac{d}{ds}\left(C\int_{-2}^{4} \frac{1}{u-s} du\right) = \frac{d}{ds}\left(\ln\left(\frac{4-s}{s+2}\right)\right) = -\frac{1}{4-s} - \frac{1}{s+2}.$$

At $s = 2$, we find that $\mathcal{H}\int_{-2}^{4} \frac{1}{(u-s)^2} du = -3/4$. More generally, the Hadamard finite part integral is defined in terms of the kth-order derivative of the Cauchy principal value integral:

$$\mathcal{H}\int_{a}^{b} \frac{f(u)}{(u-s)^{k+1}} du \triangleq \frac{1}{k!}\frac{d^k}{ds^k}\left(C\int_{a}^{b} \frac{f(u)}{u-s} du\right).$$

At a high-level, singular integrals are a modeling tool in the same way that derivatives, integrals, and polynomials are all tools for modeling the world. Singular integrals live within a well-established area of mathematics called *distribution theory* [29, Chapter 3], which we do not use in this paper to make the presentation more accessible.

State of the Art. Programming languages do not typically support integration as a primitive, let alone singular integration. Moreover, many programs (e.g., arising in mechanical engineering, aerodynamics, and computer graphics) are implemented in *differentiable programming languages*, such as PyTorch [69] and JAX [11], which take in a program representing a real-valued function and efficiently compute its derivative using *automatic differentiation* [32].

Since commonly used differentiable programming languages do not have integration in the language [11, 69], users typically implement estimators for the integral of a function using a for-loop over points in the domain of integration. However, doing so is challenging when programs use singular integrals. Estimating these integrals with a standard approach such as simple Monte Carlo integration produces incorrect results (e.g., unbounded bias and infinite variance). Differentiating these programs with singular integrals presents a significant challenge, as naively differentiating an estimator often yields an incorrect derivative, even when the original estimator is correct.

Our Approach. In this paper, we define the first differentiable programming language with support for singular integrals. We present a differentiable programming language, SingularFlow, which uses a Monte Carlo method with unbiased, strongly consistent, and finite variance estimators, unlike a standard Monte Carlo integration for which the bias and variance are unbounded.

We use SingularFlow to find approximate solutions to *singular integral equations*, equations where the unknown function is in the integrand of a singular integral, such as the following:

$$C\int_{a}^{b} \frac{f(u)}{u-s} du = g(u) \qquad \text{for all } s \in (a,b), \tag{1}$$

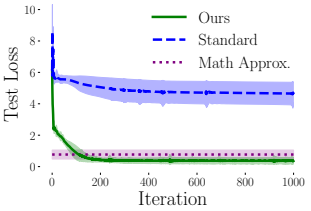
where $a, b \in \mathbb{R}$, $f : \mathbb{R} \to \mathbb{R}$ is unknown, and $g : \mathbb{R} \to \mathbb{R}$ is given. Our solver uses *physically-informed neural networks (PINNs)* [73] to solve problems in fracture mechanics [64] and aerodynamics [24, 40].

We also implement and differentiate a fundamental operation called the *Hilbert transform*, which arises in signal processing [15, 68] and many areas of physics (e.g., optics, scattering, wave mechanics) [19, 27, 28, 42, 81]. Previously, researchers hand-implemented application-specific techniques to handle singular integrals [33, 54, 80], while SingularFlow automatically produces correct results.

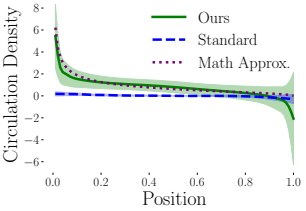
Our contributions are as follows:

- We identify and formally specify the problem of evaluating and differentiating singular integrals in a programming language. This includes a self-contained, rigorous description

(a) NACA 6412 airfoil with the chord line (dashed in purple).



(b) Loss over iterations.



(c) Learned solution.

Fig. 2. In aerodynamics, the flow of air around a wing can be modeled by a singular integral equation. Figure 2a shows a type of wing called a thin airfoil (model NACA 6612). We compare a Monte Carlo method that accounts for the singularity (Ours) to standard Monte Carlo integration (Standard) and a by-hand derivation of a mathematical approximation (Math Approx.). Figure 2b shows the loss over iterations, and Figure 2c shows the function predicted by the three methods. Our method converges to a loss (squared error) that is less than the other methods and the predicted function is close to the math approximation.

of singular integrals and their derivatives using only elementary tools from calculus (Appendix 3); we are unaware of such description in the literature. This will benefit future researchers in the programming languages community working on singular integrals.

- We introduce SingularFlow—a differentiable programming language with a primitive for singular integration. It is the first language to support singular integrals and their derivatives. We provide a denotational semantics for SingularFlow (Appendix 4) and identify conditions under which programs denote smooth functions (i.e., a law of composition, as in Theorem 4.2).
- Our operational semantics for SingularFlow performs Monte Carlo integration and supports automatic differentiation (Appendix 5). We prove that the operational semantics provides a well-behaved estimator of the denotational semantics (Theorems 5.5 and 5.7).
- We implement SingularFlow[2] in JAX [11] and evaluate it on the finite Hilbert transform (used in signal processing and physics [15, 19]) and its derivative (Appendix 6.1). We use SingularFlow to numerically estimate solutions to singular integral equations including problems from aerodynamics (Sections 2 and 6.2) and problems from mechanical engineering (Appendix 6.3).

In the future, we hope that support for singular integrals in differentiable programming languages will accelerate practitioners in domains such as science and engineering by providing a higher level of abstraction (e.g., from loops to integrals), numerical robustness, and automatic differentiation.

2 Motivating Example: Aerodynamics

In this section, we introduce SingularFlow by example. We implement a model of the flow of air around a wing that uses a neural network to solve a singular integral equation.

Circulation Density on a Wing. To make a plane fly, engineers design wings that produce enough *lift* to keep the plane in the air. Lift is the force created by the movement of a fluid around an object, acting perpendicularly to the flow direction. Lift can be calculated from *circulation density*, which is a scalar that quantifies the rate of rotation of a vector field (e.g., wind) about a point.

A *thin airfoil* is a model for a 2D slice along the length of a 3D wing, simplifying the analysis to modeling the effect of a two-dimensional *air stream* (wind). This simplifying assumption introduces a singularity at the *leading edge*, the part of the wing that first contacts the air stream (the leftmost point in Figure 2a), due to the difference in the speed of the air stream above and below the wing.

[2]The implementation is available at https://github.com/martinjm97/singularflow.

Moreover, it corresponds to a phenomenon observed in practice called leading-edge thrust [13]. Figure 2a depicts an example of a thin airfoil called the NACA 6412 [1].[3]

The Airfoil Equation. The *airfoil equation* is a singular integral equation that models the circulation density around a thin airfoil [40, Section 11.14]. The airfoil equation is expressed in terms of the Cauchy principal value integral, which we identify with a C, and is defined as the limit of the Riemann integral as the domain of integration approaches the singularity:

$$C\int_a^b \frac{f(u)}{u-s}du \triangleq \lim_{\epsilon\to 0^+}\left(\int_a^{s-\epsilon}\frac{f(u)}{u-s}du + \int_{s+\epsilon}^b \frac{f(u)}{u-s}du\right).$$

The Cauchy principal value integral is well-defined for every smooth function $f : \mathbb{R} \to \mathbb{R}$ and for real numbers $a < s < b$. The airfoil equation is given by

$$-\frac{1}{2\pi}\cdot C\int_0^c \frac{\gamma(u)}{u-u_0}du = \alpha(u_0)V_\infty \qquad \text{for all } u_0 \in (0,c), \tag{2}$$

where $\gamma : \mathbb{R} \to \mathbb{R}$ is the unknown *circulation density*, $c > 0$ is the *chord length*, $u_0 \in (0, c)$ is the location of the point of interest on the *chord line*, which is the imaginary straight line from one end of the wing to the other (depicted as the dotted purple line in Figure 2a), $\alpha : \mathbb{R} \to \mathbb{R}$ is the *angle of attack*, and $V_\infty > 0$ is the speed of the air stream. The chord length is defined as the length of the chord line. The angle of attack is the angle between the chord line and the air stream.

Numerical Integration of the Cauchy Principal Value Integral. Researchers have designed many numerical integration techniques for singular integrals [53, 63, 71, 78]. We call a known Monte Carlo integrator for the Cauchy principal value integral the *symmetric sampling estimator* [53]. The idea behind this estimator is to sample u from the domain of integration and consider its reflection across s, which is $2s - u$; if both of u and $2s - u$ are in the domain of integration, we evaluate the integrand at both points and sum them up; otherwise, we evaluate the integrand only at u. The symmetric sampling estimator cancels out the singularity and results in an unbiased estimator.

Solving Singular Integral Equations with Neural Networks. The solution to Equation (2) is a function γ that satisfies the airfoil equation for all $u_0 \in (0, c)$. Ideally, there would be an integral inversion formula that could reverse the effect of integration by expressing the integrand in terms of the integral of other terms (e.g., we would want to transform Equation (2) to a formula of the form $\frac{\gamma(u)}{u-u_0} = \cdots$, where the ellipses stand in for a function). However, King [40, Section 11.4] shows that there is no integral inversion formula for this equation, justifying approximation.

Following recent work [33, 54, 80], we use a neural network $\tilde{\gamma}_\theta : \mathbb{R} \to \mathbb{R}$ to approximate γ in Equation (2). Solving the integral equation then reduces to finding the p parameters $\theta \in \mathbb{R}^p$ of the neural network that minimize the squared error between the left- and right-hand sides of Equation (2) for a fixed set of *collocation points* $u_0^{(1)}, \ldots, u_0^{(n)} \in (0, c)$ along the chord line:

$$\min_\theta \sum_{i=1}^n \left(\frac{-1}{2\pi}\cdot C\int_0^c \frac{\tilde{\gamma}_\theta(u)}{u-u_0^{(i)}}du - \alpha(u_0^{(i)})V_\infty\right)^2. \tag{3}$$

SingularFlow allows us to express this loss function directly as a program and to estimate the singular integral using a singularity-aware Monte Carlo integration algorithm:

[3] NACA (National Advisory Committee for Aeronautics) is the government aeronautics research organization that was the precursor to NASA (National Aeronautics and Space Administration). The 4-series NACA airfoils written as NACA XXXX were studied both theoretically and empirically by NACA after World War I and during World War II [1]. The four digits determine the dimensions of the airfoil.

```
1  loss, step_size, theta = 0, 0.01, Param("theta", init())
2  c, V_inf = 1, 1  # Given function alpha depends on airfoil shape
3  for uo in uos:  # uos is a list of points between 0 and c
4    est = integral (0,c) (nn(theta, u)/(u-uo)) du
5    loss += (-1/(2 * pi) * est - alpha(u0) * Vinf)^2
6  theta -= deriv(loss, theta) * step_size
```

Line 1 initializes the variable `loss` to zero, the step size to 0.01, and the parameters to random values specified by an initialization function `init`. Line 2 sets the constants $c = 1$, $V_\infty = 1$, and then we assume that the function $\alpha(u_0)$ is defined. Line 3 iterates over each of the collocation points (which we write in the core language by iterating over a range). Line 4 uses the integral primitive of SingularFlow to estimate the Cauchy principal value integral. The first argument to the integral primitive is a pair representing the domain of integration, the second argument represents the integrand, and the third argument declares that `u` is a variable of integration. Line 5 accumulates the squared error into the variable `loss`. Finally, on Line 6, we use the `deriv` primitive to compute the derivative of the loss with respect to the parameter `theta` (which we implement by extracting the derivative part in the forward-mode operational semantics in Figure 8) and then update the parameters using a single step of gradient descent.

The syntax above is a simplified version of the actual JAX [11] implementation. In practice, we use the *vmap* command to vectorize the computation over the collocation points, the *jit* command to compile the code for faster execution (effectively unrolling the above loop), and we use a library to implement the Adam optimizer rather than using gradient descent.

Methodology. We use a multilayer perceptron with one input, two hidden layers of 100 units each with the smooth GeLU activation function, and one output.[4] We use the Adam optimizer with a learning rate of 0.01 and train for 1000 iterations using 50 samples to estimate the singular integral. We use 50 uniformly spaced collocation points from 0 to 1 and then remove the endpoints.

Results. In Figure 2, we compare symmetric sampling (Ours) to standard Monte Carlo integration (Standard) and a by-hand derivation of a mathematical approximation (Math Approx.).

Figure 2b shows the loss over iterations for training neural networks to solve the airfoil equation. We train each neural network with 5 different random seeds and plot the mean (darker line) and standard deviation (lighter line) of the test losses across runs. We use different random samples for the test loss than the train loss. We use 1000 test samples instead of 50 training samples. We provide the training loss curves in Appendix D. The dotted purple line represents the loss of Math Approx., which serves as both a baseline and a sanity check for the results. We see that Ours converges to a squared error loss that is slightly less than Math Approx. The loss is the squared error, which is a typical measure of end-application performance (as in Harold Page Starr [35, Tables 2.1-2.13]). As a result, the low loss indicates that Ours provides a good solution to the airfoil equation.

Figure 2c depicts the function predicted by Ours, Math Approx., and Standard. We see that Ours is close to the Math Approx., while the Standard is essentially a flat line at zero.

Training takes about 13 seconds for Standard and about 18 seconds for Ours.

3 Theory of Singular Integrals and Their Derivatives

To define the semantics of SingularFlow, we need a mathematical foundation for singular integrals and their derivatives. We present a self-contained introduction to singular integrals, specifically the Cauchy principal value integral and the Hadamard finite part integral. We then show how to differentiate these singular integrals. Although it is not one of our core contributions, we believe our work is the first to rigorously state and prove that a singular integral with a parameter-dependent

[4] We use a smooth activation function so that the theoretical results from Appendix 3 apply.

integrand is smooth in those parameters, and it admits nice differentiation formulas under the integral sign (Propositions 3.12–3.15).[5] The proofs of the results in this section are in Appendix A.

Throughout the section, we will use the convention that a, b, and s are real numbers such that $a < s < b$, where the interval from a to b represents the domain of integration. The variable s is thus implicitly quantified over the interval (a, b) in this section. Also, we use the notation $B_\epsilon(c) \triangleq (c - \epsilon, c + \epsilon)$ to represent the open ball of radius $\epsilon > 0$ centered at $c \in \mathbb{R}$.

3.1 Cauchy Principal Value Integral

The plot in Figure 1a depicts the function $\frac{1}{u-2}$, which has a singularity at $u = 2$. In calculus, if $a < 2 < b$ then the integral $\int_a^b \frac{1}{u-2} du$ does not exist (i.e., it does not have a well-defined finite value) due to the singularity at $u = 2$. More generally, for every $s \in (a, b)$ and smooth function $f : \mathbb{R} \to \mathbb{R}$, the integral $\int_a^b \frac{f(u)}{u-s} du$ exists if and only if $f(s) = 0$, as we show below.

Proposition 3.1. *For all smooth f, the integral $\int_a^b \frac{f(u)}{u-s} du$ exists if and only if $f(s) = 0$.* «

The *Cauchy principal value integral*, written as $C\int_a^b f(u)/(u-s)du$, is a generalization of the Riemann integral that exists for every smooth function f (even when $f(s) \neq 0$).[6] Informally, the Cauchy principal value integral is defined so that the two instances of infinity of the integrand $f(u)/(u-s)$ at $u = s$ cancel each other out. For example, in Figure 1a the integrand approaches $-\infty$ as u approaches 2 from the left and $+\infty$ as u approaches 2 from the right. We can imagine approaching the singularity $s = 2$ from the left and right at the same rate, so that the two infinities cancel each other out. The Cauchy principal value integral is inspired by this idea of cancellation, and defines the result of integration as what remains after the cancellation. We now provide the formal definition.

Definition 3.2 ([64]). The *Cauchy principal value integral* is defined as

$$C\int_a^b \frac{f(u)}{u-s} du \triangleq \lim_{\epsilon \to 0^+} \int_{[a,b] \setminus B_\epsilon(s)} \frac{f(u)}{u-s} du$$

for every smooth function $f : \mathbb{R} \to \mathbb{R}$. △

The Cauchy principal value integral exists for every smooth function and can be written in terms of Riemann integrals. Moreover, the Cauchy principal value integral is a linear operator on smooth functions and is a (strict) generalization of the corresponding Riemann integral in that they agree with each other whenever the Riemann integral exists.

Proposition 3.3. *For every smooth $f : \mathbb{R} \to \mathbb{R}$, the following hold:*

(a) $C\int_a^b \frac{f(u)}{u-s} du$ is well-defined and

$$C\int_a^b \frac{f(u)}{u-s} du = \int_a^s \frac{f(u) - f(2s-u)}{u-s} du - \int_{2s-b}^a \frac{f(2s-u)}{u-s} du, \tag{4}$$

[5] For example, [56] references a proof in [57], which states that the proof of smoothness is implicit in [34] and references a proof in [55, Pages 8-12]. However, (i) [55] does not give a formal theorem statement, (ii) its proof implicitly relies on the numerator of an integrand being analytic, and (iii) the proof is not mathematically rigorous in many steps (e.g., does not formally justify the exchange of limits, does not prove the smoothness of a singular integral, etc). In contrast, (i) we provide a formal theorem statement, (ii) our proof requires only smoothness (a weaker condition than analyticity) of the numerator, and (iii) the proof is rigorous in each step. To achieve (iii), our proof relies on several results such as Lemmas A.4-A.6, which prove that the integral of a jointly continuous (or smooth) function is continuous (or smooth) and certain singular integrands can be formally extended to a smooth function.

[6] We present all results in terms of the Riemann integral since it is more elementary and should be more familiar to the reader. The theoretical development in this paper holds immediately for Lebesgue integration as well, because our proofs do not use any properties that differ between the two notions of integration.

where the two Riemann integrals in Equation (4) are well-defined.

(b) The Cauchy principal value integral is a linear operator on smooth functions:

$$C\int_a^b \frac{c_1 f_1(u) + c_2 f_2(u)}{u-s}du = c_1 \cdot \left(C\int_a^b \frac{f_1(u)}{u-s}du\right) + c_2 \cdot \left(C\int_a^b \frac{f_2(u)}{u-s}du\right)$$

for all $c_1, c_2 \in \mathbb{R}$ and smooth $f_1, f_2 : \mathbb{R} \to \mathbb{R}$.

(c) If $\int_a^b \frac{f(u)}{u-s}du$ is well-defined, then $C\int_a^b \frac{f(u)}{u-s}du = \int_a^b \frac{f(u)}{u-s}du$. «

Example 3.4. Let $a = -1$, $b = 1$, $s = 0$, and $f(u) = 1$. Then by Proposition 3.3(a),

$$C\int_{-1}^{1} \frac{1}{u}du = \int_{-1}^{0} \frac{1-1}{u}du - \int_{0-1}^{-1} \frac{1}{u}du = 0.$$

The result makes sense because the integrand is an odd function (a function $h : \mathbb{R} \to \mathbb{R}$ such that $h(-x) = -h(x)$ for all $x \in \mathbb{R}$) and the domain of integration is symmetric so that the positive and negative parts of the integrand cancel. △

Example 3.5. Let $a = -1$, $b = 2$, $s = 1$, and $f(u) = u$. Then by Proposition 3.3(a),

$$C\int_{-1}^{2} \frac{u}{u-1}du = \int_{-1}^{1} \frac{u-(2-u)}{u-1}du - \int_{2-2}^{-1} \frac{2-u}{u-1}du.$$

The first integral simplifies to $\int_{-1}^{1} 2du = 4$, and the second integral simplifies to $\int_0^{-1} \frac{2-u}{u-1}du = \int_0^{-1}(-1 + \frac{1}{u-1})du = 1 + \ln 2$. Hence, the result is $4 - (1 + \ln 2) = 3 - \ln 2$. △

We now present an alternative formula that expresses the Cauchy principal value integral in terms of the Riemann integral, which we use to prove the operational semantics correct.

Proposition 3.6. *For every smooth function $f : \mathbb{R} \to \mathbb{R}$,*

$$C\int_a^b \frac{f(u)}{u-s}du = \begin{cases} \int_a^s \frac{f(u)-f(2s-u)}{u-s}du + \int_{2s-a}^b \frac{f(u)}{u-s}du & \text{if } s \in \left(a, \frac{a+b}{2}\right] \\ \int_s^b \frac{f(u)-f(2s-u)}{u-s}du + \int_a^{2s-b} \frac{f(u)}{u-s}du & \text{if } s \in \left[\frac{a+b}{2}, b\right). \end{cases} \tag{5}$$

«

3.2 Hadamard Finite Part Integral

The plot in Figure 1b depicts the function $1/(u-2)^2$, which has a singularity at $u = 2$. In calculus, if $a < 2 < b$, then the integral $\int_a^b \frac{1}{(u-2)^{k+1}}du$ does not exist for all positive integers k (as in the previous subsection) because the integrand has a singularity at $u = 2$. More generally, for every $s \in (a, b)$ and smooth $f : \mathbb{R} \to \mathbb{R}$, the integral $\int_a^b \frac{f(u)}{(u-s)^{k+1}}du$ exists if and only if $f^{(i)}(s) = 0$ for all $i \in \{0, \ldots, k\}$, where $f^{(i)}$ denotes the ith order derivative of f.

Proposition 3.7. *For all smooth $f : \mathbb{R} \to \mathbb{R}$ and integers $k \geq 0$, the integral $\int_a^b \frac{f(u)}{(u-s)^{k+1}}du$ exists if and only if $f^{(i)}(s) = 0$ for all $i \in \{0, \ldots, k\}$.* «

The *Hadamard finite part integral* handles cases when f or some of its ith derivatives are nonzero. For every smooth f, it is well-defined (even when $f^{(i)}(s) \neq 0$ for some $0 \leq i \leq k$) and equals the Riemann integral when both are defined. Concretely, the Hadamard finite part integral is defined as a constant times the kth derivative of the corresponding Cauchy principal value integral.

Definition 3.8. For all smooth functions $f : \mathbb{R} \to \mathbb{R}$ and integers $k \geq 1$, the *Hadamard finite part integral* is defined as the kth derivative of the Cauchy principal value integral:

$$\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du \triangleq \frac{1}{k!}\frac{d^k}{ds^k}\left(C\int_a^b \frac{f(u)}{u-s}du\right).$$

△

While our definition differs from Monegato [63, Definition 2.3] and Monegato [64], it is interchangeable (e.g., Monegato [63, Property 2.5] matches our definition). A way to arrive at our definition is by splitting the integral into two at the singularity and subtracting out terms that diverge in the limit, yielding only the *finite part* of the integral [34]. This informal procedure agrees with Definition 3.8.

For example, the informal calculation of $\mathcal{H}\int_{-1}^{1} \frac{e^{u^2}}{u^2}du$ would split the integral into $\int_{-1}^{-\epsilon} \frac{e^{u^2}}{u^2}du$ and $\int_{\epsilon}^{1} \frac{e^{u^2}}{u^2}du$. Then by integration by parts, $\int_{\epsilon}^{1} \frac{e^{u^2}}{u^2}du = \int_{\epsilon}^{1} \frac{2ue^{u^2}}{u}du - \left.\frac{e^{u^2}}{u}\right|_{\epsilon}^{1} = \int_{\epsilon}^{1} 2e^{u^2}du - \left(e - \frac{e^{\epsilon^2}}{\epsilon}\right)$, and similarly, $\int_{-1}^{-\epsilon} \frac{e^{u^2}}{u}du = \int_{-1}^{-\epsilon} 2e^{u^2}du - \left(e - \frac{e^{\epsilon^2}}{\epsilon}\right)$. In the limit as ϵ goes to 0, the term $\frac{e^{\epsilon^2}}{\epsilon}$ diverges. Subtracting $\frac{e^{\epsilon^2}}{\epsilon}$ from both expressions makes the integral converge in the limit. Only the *finite part* remains. Specifically, $\int_{-1}^{1} 2e^{u^2}du - 2e \approx 0.414$. At the end of this section, we will confirm our formal definition is the same as in this informal calculation.

The following theorem states that the Hadamard finite part integral is well-defined, is a linear operator on smooth functions, and agrees with the Riemann integral when both are well-defined. We were not able to find rigorous proofs of the following results in the literature (see Footnote 5).

Proposition 3.9. *For every smooth $f : \mathbb{R} \to \mathbb{R}$ and integer $k \geq 1$, the following hold:*

(a) $\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du$ is well-defined and satisfies integration by parts:

$$\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du = \begin{cases} -\left.\frac{f(u)}{u-s}\right|_{u=a}^{b} + C\int_a^b \frac{f'(u)}{u-s}du & \text{if } k = 1 \\ -\left.\frac{1}{k}\frac{f(u)}{(u-s)^k}\right|_{u=a}^{b} + \frac{1}{k}\mathcal{H}\int_a^b \frac{f'(u)}{(u-s)^k}du & \text{if } k \geq 2. \end{cases} \tag{6}$$

(b) The Hadamard finite part value integral is a linear operator on smooth functions:

$$\mathcal{H}\int_a^b \frac{c_1 f_1(u) + c_2 f_2(u)}{(u-s)^{k+1}}du = c_1 \cdot \left(\mathcal{H}\int_a^b \frac{f_1(u)}{(u-s)^{k+1}}du\right) + c_2 \cdot \left(\mathcal{H}\int_a^b \frac{f_2(u)}{(u-s)^{k+1}}du\right)$$

for all $c_1, c_2 \in \mathbb{R}$ and smooth $f_1, f_2 : \mathbb{R} \to \mathbb{R}$.

(c) If $\int_a^b \frac{f(u)}{(u-s)^{k+1}}du$ is well-defined, then $\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du = \int_a^b \frac{f(u)}{(u-s)^{k+1}}du$. «

We can write the Hadamard finite part integral in terms of the Cauchy principal value integral plus some boundary terms.

Proposition 3.10. *For all integers $k \geq 1$ and smooth functions $f : \mathbb{R} \to \mathbb{R}$,*

$$\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du = \frac{1}{k!}C\int_a^b \frac{f^{(k)}(u)}{u-s}du - \sum_{i=1}^{k} \frac{(i-1)!}{k!}\left.\frac{f^{(k-i)}(u)}{(u-s)^i}\right|_{u=a}^{b}. \tag{7}$$

«

We now apply the above formula and confirm the previous informal calculation.

Example 3.11. Calculate $\mathcal{H}\int_{-1}^{1} \frac{e^{u^2}}{u^2}du$. By Proposition 3.10, $\mathcal{H}\int_a^b \frac{f(u)}{u^2}du = C\int_a^b \frac{f'(u)}{u}du - \left.\frac{f(u)}{u}\right|_{u=a}^{b}$. At $a = -1$, $b = 1$, $f(u) = e^{u^2}$, and $s = 0$, we have

$$\mathcal{H}\int_{-1}^{1} \frac{e^{u^2}}{u^2}du = C\int_{-1}^{1} \frac{2ue^{u^2}}{u}du - \left.\frac{e^{u^2}}{u}\right|_{u=-1}^{1} = \int_{-1}^{1} 2e^{u^2}du - (e - (-e)) = 2\int_{-1}^{1} e^{u^2}du - 2e \approx 0.414.$$

△

3.3 Differentiating Singular Integrals

In this section, we provide derivative rules for singular integrals, which enables us to build a differentiable programming language that supports singular integration.

We show that the derivative and integral commute when differentiating a singular integral with respect to parameters that occur in the integrand but do not correspond to the singularity. These results imply that the smooth neural network that models the unknown function in Equation (1) can be trained by differentiating the numerical integrator for the singular integral.

Proposition 3.12. *For every integer $m \geq 0$ and smooth function $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$, the function $g(s, v_1, \ldots, v_m) = C\int_a^b \frac{f(u,v_1,\ldots,v_m)}{u-s} du$ is smooth on $(a, b) \times \mathbb{R}^m$. Moreover, for every integer $n \geq 0$ and $i_1, \ldots, i_n \in \{1, \ldots, m\}$,*

$$\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} \left(C \int_a^b \frac{f(u, v_1, \ldots, v_m)}{u - s} du \right) = C \int_a^b \frac{\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} f(u, v_1, \ldots, v_m)}{u - s} du. \tag{8}$$

«

Proposition 3.13. *For every integer $m \geq 0$, smooth function $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$, and integer $k \geq 1$, the function $g(s, v_1, \ldots, v_m) = \mathcal{H}\int_a^b \frac{f(u,v_1,\ldots,v_m)}{(u-s)^{k+1}} du$ is smooth on $(a, b) \times \mathbb{R}^m$. Moreover, for every integer $n \geq 0$ and $i_1, \ldots, i_n \in \{1, \ldots, m\}$,*

$$\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} \left(\mathcal{H} \int_a^b \frac{f(u, v_1, \ldots, v_m)}{(u - s)^{k+1}} du \right) = \mathcal{H} \int_a^b \frac{\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} f(u, v_1, \ldots, v_m)}{(u - s)^{k+1}} du. \tag{9}$$

«

Next, we give a formula for the derivative of a singular integral with respect to the singularity.

Proposition 3.14. *For every smooth function $f : \mathbb{R} \to \mathbb{R}$ and integer $n \geq 1$,*

$$\frac{d^n}{ds^n} \left(C \int_a^b \frac{f(u)}{u - s} du \right) = n! \cdot \mathcal{H} \int_a^b \frac{f(u)}{(u - s)^{n+1}} du.$$

«

Proposition 3.15. *For every smooth function $f : \mathbb{R} \to \mathbb{R}$ and integers $n, k \geq 1$,*

$$\frac{d^n}{ds^n} \left(\mathcal{H} \int_a^b \frac{f(u)}{(u - s)^{k+1}} du \right) = \frac{(n + k)!}{k!} \cdot \mathcal{H} \int_a^b \frac{f(u)}{(u - s)^{n+k+1}} du.$$

«

The derivative with respect to the singularity s may not commute with the singular integral even when the numerator $f(u)$ in the integrand is smooth as we show in the following example.

Example 3.16. Using Proposition 3.14: $\frac{d}{ds}\left(C\int_{-1}^{1} \frac{e^{u^2}}{u-s} du\right)\Big|_{s=0} = \mathcal{H}\int_{-1}^{1} \frac{e^{u^2}}{(u-s)^2} du\Big|_{s=0} \approx 0.414$, where the second approximate equality is by Example 3.11. Note the change from the Cauchy principal value integral to the Hadamard finite part integral here. If we instead commute the derivative and the integral by differentiating the integrand at $s = 0$ and integrating the outcome as in the definition of the Cauchy principal value integral, we get $\lim_{\epsilon \to 0^+} \left(\int_{-1}^{-\epsilon} \frac{e^{u^2}}{u^2} du + \int_{\epsilon}^{1} \frac{e^{u^2}}{u^2} du \right) = \infty$ because as u approaches 0 from the left and from the right, the integrand $\frac{e^{u^2}}{u^2}$ diverges to $+\infty$. Thus, commuting the derivative and the Cauchy principal value integral does not work in this case. △

4 SingularFlow: A Differentiable Language with Singular Integrals

In this section, we define the syntax and denotational semantics of SingularFlow. SingularFlow has a singular integral primitive, which no prior programming language to date has supported as a language primitive. We provide proofs of the results in this section in Appendix B, and a discussion of the boundaries of available theory and their implications to programming is in Appendix F.

4.1 Syntax

The syntax of SingularFlow is defined by the following grammar:

$$e ::= c \mid x \mid h(e, \ldots, e) \qquad p ::= x = e \mid y = \textbf{integral}\ (a, b)\ e/(x-s)^k\ \mathbf{d}x \mid p; p \mid \textbf{ifpos}\ e\ \textbf{then}\ p\ \textbf{else}\ p \mid \textbf{for}\ x\ \textbf{in}\ \textbf{range}(a, b, c) : p$$

$a, b, c \in \mathbb{R},\ k \in \mathbb{N},\ x, y, s \in \mathit{Var},\ h \in \mathit{Op}$.

Here Var is a finite set of variable symbols, and Op is a countable set of function symbols that includes + and × and is closed under taking partial derivatives. Expressions e represent smooth functions of variables, written as x, y, and s. We write constant reals as a, b, and c, and integers as k. Each function symbol $h \in \mathit{Op}$ represents a computable smooth function from $\mathbb{R}^m$ to $\mathbb{R}$ for $m \in \mathbb{N}$.

Programs p are commands in an imperative programming language. The two assignment statements set a variable to the value of either an expression or a singular integral. In the latter case, the integrand has the form e divided by $(x-s)^k$, where the variable x is bound in the declaration of the integral and the domain of integration is specified by constants for simplicity. The conditional evaluates the true branch if its condition e is positive, and the false branch if e is negative. When e is zero, the conditional raises an error, signaling a possible issue with differentiation. SingularFlow supports sequential composition and a bounded for-loop ranging from a to b with increment c.

For simplicity, we consider only the assignments $y = \textbf{integral}\ (a, b)\ e/(x-s)^k\ \mathbf{d}x$, where the domain of integration satisfies $a \le b$ and the singular variable s is different from x and all the free variables of e. We suppress the exponent of the denominator in the integrand when it is 1: $\textbf{integral}\ (a, b)\ e/(x-s)\ \mathbf{d}x \triangleq \textbf{integral}\ (a, b)\ e/(x-s)^1\ \mathbf{d}x$.

4.2 Denotational Semantics

Figure 3 depicts the denotational semantics of expressions in SingularFlow. A context γ is a map from variables to real numbers, and the set of all contexts is denoted by $\mathit{Ctx} \triangleq [\mathit{Var} \to \mathbb{R}]$. Since Var is finite, we often view Ctx as $\mathbb{R}^{|\mathit{Var}|}$ equipped with the standard topology.

An expression e denotes a smooth function $[\![e]\!]$ that maps a context γ to a real number. The definition of $[\![e]\!]$ is standard. Note that each function symbol $h \in \mathit{Op}$ denotes a smooth function $[\![h]\!] : \mathbb{R}^m \to \mathbb{R}$, where m is the arity of h.

Figure 4 describes the denotational semantics of programs in SingularFlow. A program p denotes a function $[\![p]\!]$ from a context $\gamma \in \mathit{Ctx}$ to either an updated context or an error, written as err. The denotation of the assignment of a variable to an expression $[\![x = e]\!](\gamma)$ updates the context with x mapped to the denotation of e in γ.

The semantics of the singular-integral assignment varies depending on the real constants $a, b \in \mathbb{R}$ and the integer $k \in \mathbb{N}$. If $\gamma(s) \in (-\infty, a) \cup (b, \infty)$, the domain of integration does not contain the singularity $\gamma(s)$, and the program denotes a standard integral. If $\gamma(s) \in (a, b)$, the program denotes a singular integral that is a Cauchy principal value integral when $k = 1$ and a Hadamard finite part integral when $k > 1$. The denotation returns an error if the singularity is exactly at the boundary of the domain of integration.

The denotation of $p_1; p_2$ is defined so that if $[\![p_1]\!](\gamma)$ returns an error, it propagates the error, and otherwise, it returns the composition the denotations of p_2 and p_1 at γ. The denotation of a conditional $[\![\textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2]\!](\gamma)$ is $[\![p_1]\!](\gamma)$ if $[\![e]\!]\gamma$ is positive, and $[\![p_2]\!](\gamma)$ if $[\![e]\!]\gamma$ is negative. If $[\![e]\!]\gamma$ is zero, the denotation returns an error. This way of handling the $[\![e]\!]\gamma = 0$ case ensures differentiability for non-error computations, because $\mathrm{sgn} \circ [\![e]\!]$ is constant in a ball around γ.

The denotation of a for-loop $[\![\textbf{for}\ x\ \textbf{in}\ \textbf{range}(a, b, c) : p]\!](\gamma)$ is standard. It does not update the context when $a \ge b$, and it propagates the error if $[\![p]\!](\gamma[x \mapsto a])$ is err. If $[\![p]\!](\gamma[x \mapsto a])$ is an updated context γ', it recurses on the incremented counter with the context γ'.

$$[\![e]\!] : Ctx \to \mathbb{R}, \quad [\![c]\!]\gamma \triangleq c, \quad [\![x]\!]\gamma \triangleq \gamma(x), \quad [\![h(e_1, \ldots, e_m)]\!]\gamma \triangleq [\![h]\!]([\![e_1]\!]\gamma, \ldots, [\![e_m]\!]\gamma).$$

Fig. 3. Denotational semantics of expressions in SingularFlow.

$$[\![p]\!] : Ctx \to (Ctx \cup \{\text{err}\}),$$
$$[\![x = e]\!](\gamma) \triangleq \gamma[x \mapsto [\![e]\!]\gamma],$$
$$\left[\!\!\left[y = \begin{pmatrix} \textbf{integral } (a,b) \\ e/(x-s)^k \ \mathbf{d}x \end{pmatrix} \right]\!\!\right](\gamma) \triangleq \text{if } (c = \text{err}) \text{ then err else } \gamma[y \mapsto c],$$

$$\text{where } g(u) = [\![e]\!](\gamma[x \mapsto u]) \text{ in } c = \begin{cases} \text{err} & \text{if } \gamma(s) \in \{a, b\} \\ \displaystyle\int_a^b \frac{g(u)}{(u-\gamma(s))^k} du & \text{if } \gamma(s) \in (-\infty, a) \cup (b, \infty) \\ C\displaystyle\int_a^b \frac{g(u)}{u-\gamma(s)} du & \text{if } \gamma(s) \in (a,b) \wedge k = 1 \\ \mathcal{H}\displaystyle\int_a^b \frac{g(u)}{(u-\gamma(s))^k} du & \text{if } \gamma(s) \in (a,b) \wedge k > 1, \end{cases}$$

$$[\![p_1; p_2]\!](\gamma) \triangleq \text{if } (\gamma' = \text{err}) \text{ then err else } [\![p_2]\!](\gamma'), \quad \text{where } \gamma' = [\![p_1]\!]\gamma,$$

$$[\![\textbf{ifpos } e \textbf{ then } p_1 \textbf{ else } p_2]\!](\gamma) \triangleq \begin{cases} \text{err} & \text{if } [\![e]\!]\gamma = 0 \\ [\![p_1]\!](\gamma) & \text{if } [\![e]\!]\gamma > 0 \\ [\![p_2]\!](\gamma) & \text{if } [\![e]\!]\gamma < 0, \end{cases}$$

$$[\![\textbf{for } x \textbf{ in range}(a,b,c) : p]\!](\gamma) \triangleq \begin{cases} \gamma & \text{if } a \geq b \\ [\![\textbf{for } x \textbf{ in range}(a+c,b,c) : p]\!](\gamma') & \text{if } a < b \wedge c > 0 \wedge \gamma' \neq \text{err} \\ \text{err} & \text{otherwise,} \end{cases}$$
$$\text{where } \gamma' = [\![p]\!](\gamma[x \mapsto a]).$$

Fig. 4. Denotational semantics of programs in SingularFlow.

We next present smoothness results on the denotation of expressions and programs: $[\![e]\!]$ is smooth unconditionally, and $[\![p]\!]$ is smooth at γ whenever p does not return an error at γ.

Lemma 4.1. *For every expression e, its semantics $[\![e]\!]$ is smooth on Ctx.* «

Theorem 4.2. *For every program p and $\gamma \in Ctx$, if $[\![p]\!]\gamma \neq \text{err}$, then $[\![p]\!]$ is smooth at γ.* «

We now provide two simple examples of the denotational semantics of SingularFlow.

Example 4.3. The program $y = \textbf{integral } (0,1)\ 1/(x-s)\ \mathbf{d}x$ errors if the singularity is at the boundary of the domain of integration ($\gamma(s) \in \{0,1\}$). It denotes a Riemann integral if the singularity is outside the domain of integration ($\gamma(s) \notin [0,1]$), and a Cauchy principal value integral if the singularity lies within the domain of integration ($\gamma(s) \in (0,1)$):

$$[\![y = \textbf{integral } (0,1)\ 1/(x-s)\ \mathbf{d}x]\!]\gamma = \text{if } \gamma(s) \in \{0,1\} \text{ then err else } \gamma\left[y \mapsto \begin{cases} \int_0^1 \frac{1}{u-\gamma(s)} du & \text{if } \gamma(s) \notin [0,1] \\ C\int_0^1 \frac{1}{u-\gamma(s)} du & \text{if } \gamma(s) \in (0,1) \end{cases}\right]. \quad \triangle$$

Example 4.4. The program $y = \textbf{integral } (0,1)\ 1/(x-s)^2\ \mathbf{d}x$ errors if the singularity is at the boundary of the domain of integration ($\gamma(s) \in \{0,1\}$). It denotes a Riemann integral if the singularity is outside the domain of integration ($\gamma(s) \notin [0,1]$), and a Hadamard finite part integral if the singularity lies within the domain of integration ($\gamma(s) \in (0,1)$):

$$[\![y = \textbf{integral } (0,1)\ 1/(x-s)^2\ \mathbf{d}x]\!]\gamma = \text{if } \gamma(s) \in \{0,1\} \text{ then err else } \gamma\left[y \mapsto \begin{cases} \int_0^1 \frac{1}{(u-\gamma(s))^2} du & \text{if } \gamma(s) \notin [0,1] \\ \mathcal{H}\int_0^1 \frac{1}{(u-\gamma(s))^2} du & \text{if } \gamma(s) \in (0,1) \end{cases}\right]. \quad \triangle$$

$$\frac{}{(\gamma, c) \Downarrow c} \qquad \frac{}{(\gamma, x) \Downarrow \gamma(x)} \qquad \frac{(\gamma, e_1) \Downarrow c_1 \quad \cdots \quad (\gamma, e_m) \Downarrow c_m \qquad [\![h]\!](c_1, \ldots, c_m) = c}{(\gamma, h(e_1, \ldots, e_m)) \Downarrow c}$$

Fig. 5. Standard operational semantics $(\gamma, e) \Downarrow c$ for expressions.

For $n = 0$:
$$D[e](z, 0) \triangleq e.$$

For $n \geq 1$:
$$D[c](z, n) \triangleq 0, \qquad D[x](z, n) \triangleq \mathbf{1}[n = 1 \wedge x = z],$$
$$D[h(e_1, \ldots, e_m)](z, n) \triangleq D[h'_1 \cdot e'_1 + \cdots + h'_m \cdot e'_m](z, n - 1),$$
where $h'_i = (\mathcal{D}_i h)(e_1, \ldots, e_m)$, $e'_i = D[e_i](z, 1)$, $\mathcal{D}_i h \in Op$ is the ith partial derivative of h.

Fig. 6. The source-to-source nth derivative $D[e](z, n)$ of an expression e with respect to a variable z, where $n \in \mathbb{N}_0$. In the rules, $\mathbf{1}[\varphi]$ is 1 if φ is true, and 0 otherwise.

5 Operational Semantics for SingularFlow

In this section, we present a stochastic operational semantics for SingularFlow that estimates singular integrals and provides a forward-mode derivative. We present a theorem relating the operational semantics to the denotational semantics. Proofs of results in this section are in Appendix C.

5.1 Operational Semantics

We organize an operational semantics for SingularFlow as follows. We first define the operational semantics of an expression as a deterministic function mapping a context to a real. We then define the operational semantics of a program as a stochastic function mapping a pair of contexts to another pair of contexts, where the first (or second) component of the output stores Monte Carlo estimates of all variables (or their derivatives). Along the way, we define two procedures: one for taking higher-order derivatives of expressions, and the other for estimating singular integrals.

Operational Semantics for Expressions. Figure 5 defines a standard operational semantics for expressions. The big-step relation $(\gamma, e) \Downarrow c$ takes as input a context γ and an expression e, and returns a real value c. The rule for $h(e_1, \ldots, e_m)$ evaluates each of the arguments e_i to c_i and then evaluates the denoted function $[\![h]\!]$ at $c_1, \ldots, c_m$.

Figure 6 defines $D[e](z, n)$, a source-to-source transformation that computes the nth derivative of an expression e with respect to a variable z, where $n \in \mathbb{N}_0$. For instance, $D[c](x, 1) = 0$ and $D[x \cdot x \cdot x](x, 2) = 6 \cdot x$ (after simplification). In the rules, $\mathcal{D}_i$ denotes the syntactic derivative of an operation $h \in Op$ with respect to the ith parameter; that is, $[\![\mathcal{D}_i h]\!](c_1, \ldots, c_m) = D_i[\![h]\!](c_1, \ldots, c_m)$ for all $c \in \mathbb{R}^m$, where m is the arity of h, $i \in \{1, \ldots, m\}$, and D_i denotes the mathematical differentiation operator with respect to the ith parameter. As mentioned in Appendix 4.1, we assume that Op is closed under taking partial derivatives: $\mathcal{D}_i h \in Op$ for all $h \in Op$ and $i \in \{1, \ldots, m\}$. For simplicity, we choose to use the source-to-source derivative to implement the nth derivative rather than generalizing forward-mode automatic differentiation [22, 75].

Monte Carlo Integration Accounting for Singularities. Figure 7 depicts a procedure for estimating singular integrals. The Standard rule applies standard Monte Carlo integration when the singularity is outside the domain of integration. By transforming a sample u from $\mathcal{U}(0, 1)$ (i.e., uniform distribution over $[0, 1]$), it generates a uniformly-sampled point t at the domain of integration, evaluates the integrand at t to compute the average value, and scales the result by the size of the domain of integration to estimate the area under the curve.

The Cauchy PV1 and Cauchy PV2 rules apply a Monte Carlo algorithm that we call *symmetrical sampling* to $\mathbf{integral}\ (a, b)\ e/(x - s)\ \mathbf{d}x$. This algorithm is different from the one used in standard Monte Carlo integration. The idea of symmetrical sampling is to sample points symmetrically around the singularity s. This is a way to implement the limit to the singularity in Definition 3.2.

$$\textsc{Standard}\quad \frac{\gamma(s) \notin [a,b] \qquad t = u \cdot (b-a) + a \qquad (\gamma[x \mapsto t], e) \Downarrow c}{(\gamma, u, u', \textbf{integral}\ (a,b)\ e/(x-s)^k\ \mathbf{d}x) \Downarrow_h (b-a) \cdot \frac{c}{(t-\gamma(s))^k}}$$

$$\textsc{Cauchy PV1}\quad \frac{\gamma(s) \in (a, \frac{a+b}{2}] \quad t = u' \cdot (\gamma(s)-a) + a \quad (\gamma[x \mapsto t], e) \Downarrow c^+ \quad b' = 2\gamma(s) - a \quad (\gamma, u, u', \textbf{integral}\ (b', b)\ e/(x-s)\ \mathbf{d}x) \Downarrow_h c' \quad (\gamma[x \mapsto 2\gamma(s)-t], e) \Downarrow c^-}{(\gamma, u, u', \textbf{integral}\ (a,b)\ e/(x-s)\ \mathbf{d}x) \Downarrow_h c' + (\gamma(s)-a) \cdot \frac{c^+ - c^-}{t - \gamma(s)}}$$

$$\textsc{Cauchy PV2}\quad \frac{\gamma(s) \in [\frac{a+b}{2}, b) \quad t = u' \cdot (b-\gamma(s)) + \gamma(s) \quad (\gamma[x \mapsto t], e) \Downarrow c^+ \quad a' = 2\gamma(s) - b \quad (\gamma, u, u', \textbf{integral}\ (a, a')\ e/(x-s)\ \mathbf{d}x) \Downarrow_h c' \quad (\gamma[x \mapsto 2\gamma(s)-t], e) \Downarrow c^-}{(\gamma, u, u', \textbf{integral}\ (a,b)\ e/(x-s)\ \mathbf{d}x) \Downarrow_h c' + (b-\gamma(s)) \cdot \frac{c^+ - c^-}{t - \gamma(s)}}$$

$$\textsc{Hadamard FP}\quad \frac{k \geq 2 \quad \forall i \in \{0,\dots,k-1\}.\ e_i = D[e](x,i) \quad \forall i \in \{0,\dots,k-2\}.\ (\gamma[x \mapsto a], e_i) \Downarrow c_i^- \quad \gamma(s) \in (a,b) \quad (\gamma, u, u', \textbf{integral}\ (a,b)\ e_{k-1}/(x-s)\ \mathbf{d}x) \Downarrow_h c' \quad \forall i \in \{0,\dots,k-2\}.\ (\gamma[x \mapsto b], e_i) \Downarrow c_i^+}{(\gamma, u, u', \textbf{integral}\ (a,b)\ e/(x-s)^k\ \mathbf{d}x) \Downarrow_h \frac{c'}{(k-1)!} - \sum_{i=1}^{k-1} \frac{(i-1)!}{(k-1)!} \left(\frac{c^+_{k-1-i}}{(b-\gamma(s))^i} - \frac{c^-_{k-1-i}}{(a-\gamma(s))^i} \right)}$$

Fig. 7. The helper function $(\gamma, u, u',\ \cdot\ ) \Downarrow_h c$ that takes in a context, two independent samples from $\mathcal{U}(0,1)$, and the syntax of an integral, and returns a constant, representing a single-point Monte Carlo estimate of the integral. We use a, b, and c with subscripts/superscripts to denote real values.

$$\textsc{Expr-Assign}\quad \frac{(\gamma, e) \Downarrow c \qquad \forall y \in \mathrm{FV}(e).\ (\gamma, D[e](y,1)) \Downarrow c'_y}{(\gamma, \gamma', x = e) \Downarrow_o \gamma[x \mapsto c], \gamma'[x \mapsto \textstyle\sum_y c'_y \cdot \gamma'(y)]}$$

$$\textsc{Integral-Assign}\quad \frac{u \leftarrow \mathcal{U}(0,1) \quad u' \leftarrow \mathcal{U}(0,1) \quad (\gamma, u, u', \textbf{integral}\ (a,b)\ e/(x-s)^{k+1}\ \mathbf{d}x) \Downarrow_h c' \quad (\gamma, u, u', \textbf{integral}\ (a,b)\ e/(x-s)^k\ \mathbf{d}x) \Downarrow_h c \quad \forall z \in \mathrm{FV}(e) \setminus \{x,s\}.\ (\gamma, u, u', \textbf{integral}\ (a,b)\ D[e](z,1)/(x-s)^k\ \mathbf{d}x) \Downarrow_h c'_z}{(\gamma, \gamma', y = \textbf{integral}\ (a,b)\ e/(x-s)^k\ \mathbf{d}x) \Downarrow_o \gamma[y \mapsto c], \gamma'[y \mapsto k \cdot c' \cdot \gamma'(s) + \textstyle\sum_z c'_z \cdot \gamma'(z)]}$$

$$\textsc{Comp}\quad \frac{(\gamma, \gamma', p_1) \Downarrow_o \gamma_1, \gamma'_1 \qquad (\gamma_1, \gamma'_1, p_2) \Downarrow_o \gamma_2, \gamma'_2}{(\gamma, \gamma', p_1; p_2) \Downarrow_o \gamma_2, \gamma'_2} \qquad \textsc{If-Pos}\quad \frac{(\gamma, e) \Downarrow c \quad c > 0 \quad (\gamma, \gamma', p_1) \Downarrow_o \gamma_1, \gamma'_1}{(\gamma, \gamma', \textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2) \Downarrow_o \gamma_1, \gamma'_1}$$

$$\textsc{If-Neg}\quad \frac{(\gamma, e) \Downarrow c \quad c < 0 \quad (\gamma, \gamma', p_2) \Downarrow_o \gamma_2, \gamma'_2}{(\gamma, \gamma', \textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2) \Downarrow_o \gamma_2, \gamma'_2} \qquad \textsc{For-Base}\quad \frac{a \geq b}{(\gamma, \gamma', \textbf{for}\ x\ \textbf{in range}(a,b,c) : p) \Downarrow_o \gamma, \gamma'}$$

$$\textsc{For-Rec}\quad \frac{a < b \quad c > 0 \quad (\gamma[x \mapsto a], \gamma'[x \mapsto 0], p) \Downarrow_o \gamma_1, \gamma'_1 \quad (\gamma_1, \gamma'_1, \textbf{for}\ x\ \textbf{in range}(a+c, b, c) : p) \Downarrow_o \gamma_2, \gamma'_2}{(\gamma, \gamma', \textbf{for}\ x\ \textbf{in range}(a,b,c) : p) \Downarrow_o \gamma_2, \gamma'_2}$$

Fig. 8. The big-step operational semantics $(\gamma, \gamma', p) \Downarrow_o \gamma_1, \gamma'_1$ performs automatic differentiation of p, including sampling from singular integrals. Metavariables a, b, and c with subscripts/superscripts denote real values.

The Cauchy PV1 rule, based on the first formula in Proposition 3.6, applies when the singularity is closer to the lower bound of the domain of integration: $\gamma(s) \in (a, \frac{a+b}{2}]$. It then sets $b' = 2\gamma(s) - b$, which is the point that is the reflection of a across $\gamma(s)$.[7] The first step in symmetrical sampling is to partition the domain into the largest symmetrical region around the singularity, (a, b'), and the remaining region, (b', b). The second step is to generate a uniformly-sampled point t from the lower half of the symmetrical region $(a, \gamma(s))$, by transforming the provided sample u' from $\mathcal{U}(0,1)$, and then reflect t across the singularity to get $2\gamma(s) - t$. The third step evaluates the numerator at both points to obtain real values c^+ and c^-. The fourth step (i) scales the obtained c^+ and c^- by the width

[7] An alternative choice is to sample t such that the symmetric samples around the singularity are $s + t$ and $s - t$. Our choice to sample t so that the symmetric samples are t and $2s - t$ better matches the results in Propositions 3.3(b) and 3.6.

of the interval $2(\gamma(s) - a)$, (ii) divides the scaled values first by 2 for the 2 samples, and then by the denominator of the integrand (i.e., $t - \gamma(s)$ and $(2\gamma(s) - t) - \gamma(s)$), and (iii) adds the outcomes of these divisions. The result of the fourth step is $\frac{\gamma(s)-a}{t-\gamma(s)}(c^+ - c^-)$ after simplification. The last step uses the standard single-sample Monte Carlo estimate of the integral from (b', b) (which uses the provided sample u, instead of u', as described earlier) and adds this estimate to the result.

The Cauchy PV2 rule also uses symmetrical sampling and applies when the singularity is closer to the upper bound of the domain integration: $\gamma(s) \in [\frac{a+b}{2}, b)$. Based on the second formula in Proposition 3.6, this rule reflects b across $\gamma(s)$, i.e., $a' = 2\gamma(s) - b$. The largest symmetrical region around the singularity is (a', b) and then the remaining region is (a, a'). The Monte Carlo estimate of the symmetrical region and the remaining region matches the previous rule.

The Hadamard FP rule implements the integration by parts formula in Proposition 3.10. For each $0 \le i \le k-1$, it computes the ith derivative of e to get e_i via the derivative rules in Figure 6. It then evaluates the integral at $e_{k-1}/(x-s)$ to get c'. Next, it evaluates e_i at the endpoints a and b to get c_i^- and c_i^+, and computes $(c^+_{k-1-i}/(b-\gamma(s))^i) - (c^-_{k-1-i}/(a-\gamma(s))^i)$ for each i. Finally, it scales the sum of these terms by $(i-1)!/(k-1)!$ and subtracts it from $c'/(k-1)!$ to get the final result.

We do not provide rules for the case where the singularity and the bounds of integration coincide (e.g., **integral** (a, b) $e/(x-s)$ $\mathbf{d}x$ at $[s \mapsto a, \ldots]$). As a result, the compiler will get stuck on these programs, corresponding to the fact that in the denotational semantics, $\gamma(s) \in \{a, b\}$ results in err.

Operational Semantics for Programs. Figure 8 presents the operational semantics for programs. The big-step evaluation relation $(\gamma, \gamma', p) \Downarrow_o \gamma_1, \gamma_1'$ takes as input a primal context γ, a derivative context γ' (which is a context mapping each variable to a real number that represents the infinitesimal perturbation to the variable), and a program p, and returns updated contexts γ_1 and γ_1'. For instance, we can compute the derivative of $f(x, y) = xy$ with respect to y at $(x, y) = (2, 3)$ in the following way. If $\gamma = [x \mapsto 2, y \mapsto 3, z \mapsto 0]$ and $\gamma' = [x \mapsto 0, y \mapsto 1, z \mapsto 0]$, then we have that

$$(\gamma, \gamma', z = x \cdot y) \Downarrow_o [x \mapsto 2, y \mapsto 3, z \mapsto 6], [x \mapsto 0, y \mapsto 1, z \mapsto 2].$$

The bindings for x and y in γ and γ' remain the same, but z is updated in both contexts. The first output context binds z to 6 because $f(2, 3) = 6$, while the second output context for derivative binds z to 2 because the total derivative is $g(x, y, dx, dy) = dx \cdot y + x \cdot dy$ and $g(2, 3, 0, 1) = 2$.

The Expr-Assign rule evaluates an expression e and its source-to-source derivative with respect to each variable in e, scaled by appropriate infinitesimal values. The Integal-Assign rule draws two independent samples u, u' from the uniform distribution over the unit interval $(0, 1)$: $u \leftarrow \mathcal{U}(0, 1)$ and $u' \leftarrow \mathcal{U}(0, 1)$. Note that we assert that samples selected on different runs are independent and identically distributed. We also rely on the common assumption that samples are exact (i.e., we use the Real-RAM model), although the implementation uses floating-point numbers.

The derivative part of the Integal-Assign rule consists of two terms: $(k \cdot c') \cdot \gamma'(s)$ and $\sum_z c_z' \cdot \gamma'(z)$ for $z \in \mathrm{FV}(e) \setminus \{x, s\}$. In the first term, $k \cdot c'$ estimates the singular integral of $(k \cdot e)/(x-s)^{k+1}$, which is the derivative of the original integrand $e/(x-s)^k$ with respect to s. Similarly, in the second term, c_z' estimates the singular integral of $D[e](z, 1)/(x-s)^k$, which is the derivative of the original integrand with respect to $z \neq s, x$. Hence, these two terms essentially commute the derivative with respect to s and z, and the original singular integral of $e/(x-s)^k$; and this is based on our results in Section 3.3 (Propositions 3.12—3.15). On the other hand, a term involving the derivative with respect to x does not appear in the rule, because the derivative of the original singular integral with respect to x is 0: the integral binds the variable x, so it is constant in x. Note that this rule shares the same samples u and u' for the primal and the derivative computation.

The derivative is particularly interesting for singular integrals because the derivative and integral do not commute (Example 3.16). Moreover, the sampler for the Hadamard finite part integral

specified in the HADAMARD FP rule also does not commute with the integral. To see this, note that the operational semantics estimates the Hadamard finite part integral, by using the equation

$$\mathcal{H}\int_a^b \frac{f(x)}{(x-s)^k}dx = \frac{1}{k!}C\int_a^b \frac{f^{(k-1)}(u)}{u-s}du - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\frac{f^{(k-1-i)}(u)}{(u-s)^i}\Big|_{u=a}^{b},$$

and by applying symmetrical sampling to the Cauchy principal value integral on the right-hand side and evaluating the sum. Differentiating the sampler is incorrect because it involves differentiating the sampler for Cauchy principal value integral, which is incorrect because the derivative and Cauchy principal value integral do not commute (Example 3.16).

The COMP rule applied to $p_1; p_2$ evaluates p_1, producing new primal and derivative contexts, which are then used to evaluate p_2. The IF-POS rule applies when the condition e evaluates to a positive number and returns the evaluation of the if-body. The IF-NEG rule works analogously. In the base case of for-loop ($a \geq b$), the FOR-BASE rule does not perform any computation and simply returns γ, γ'. In the recursive case, the FOR-REC rule takes a program $\textbf{for } x \textbf{ in range}(a, b, c) : p$ and runs p in a context where x is set to a and its derivative is set to 0. This rule also propagates any errors from executing p or from using an increment $c \leq 0$, which would lead to divergence.

5.2 Soundness of Operational Semantics

In this subsection, we show that the operational semantics provides a consistent (and unbiased in simple cases) estimator of the denotation of the program and of its derivative. We begin with the correctness of the operational semantics, and the derivative transformation, for expressions.

LEMMA 5.1. *For every context γ and expression e, we have that $(\gamma, e) \Downarrow c$ if and only if $[\![e]\!]\gamma = c$.* «

LEMMA 5.2. *The derivative transformation is correct for every expression e, $n \in \mathbb{N}_0$, and $z \in Var$: $[\![D[e](z,n)]\!]\gamma = f^{(n)}(\gamma(z))$ for all $\gamma \in Ctx$, where $f : \mathbb{R} \to \mathbb{R}$ is defined by $f(u) = [\![e]\!](\gamma[z \mapsto u])$.* «

We recall a few standard definitions from statistics, and also a well-known result that leads to a standard recipe for building a family of so-called strongly-consistent estimators.

Definition 5.3. An *estimator* T is a real-valued random variable. The estimator T is *unbiased* for a real number $\theta \in \mathbb{R}$ if $\mathbb{E}[T] = \theta$. A family of estimators $(T_m)_{m\in\mathbb{N}}$ is *strongly consistent* for $\theta \in \mathbb{R}$ if T_m almost surely converges to θ as m tends to ∞, that is, $\Pr(\lim_{m\to\infty} T_m = \theta) = 1$. △

PROPOSITION 5.4 ([26, THEOREM 10.13]). *If estimators $(T_m)_{m\in\mathbb{N}}$ are independent and identically distributed (i.i.d.) as random variables, and T_1 is unbiased for θ and has finite variance, then the estimators $T'_m \triangleq \frac{1}{m}\sum_{i=1}^m T_i$ is unbiased for θ for all m, and their family is strongly consistent for θ.* «

We first present that the operational semantics is an unbiased and consistent estimator for the denotational semantics whenever a program is single-line with the integral construct.

THEOREM 5.5 (UNBIASEDNESS AND FINITE VARIANCE). *Let $p \equiv (y = \textbf{integral}\ (a,b)\ e/(x-s)^k\ \textbf{dx})$ and $\gamma, \gamma' \in Ctx$ with $[\![p]\!]\gamma \neq$ err. Let c, c' be estimators defined by $c \triangleq \hat{\gamma}(y)$ and $c' \triangleq \hat{\gamma}'(y)$, where $(\gamma, \gamma', p) \Downarrow_o \hat{\gamma}, \hat{\gamma}'$. Then, c and c' have finite variance, and are unbiased for $[\![p]\!](\gamma)(y)$ and $\sum_{z\in Var} \frac{d}{dt}[\![p]\!](\gamma[z \mapsto t])(y)\big|_{t=\gamma(z)} \cdot \gamma'(z)$, respectively.* «

COROLLARY 5.6. *Let $p \equiv (y = \textbf{integral}\ (a,b)\ e/(x-s)^k\ \textbf{dx})$ and $\gamma, \gamma' \in Ctx$ with $[\![p]\!]\gamma \neq$ err. Separate runs of the operational semantics produce (i.i.d.) random variables $\{c_i\}_{i\in\mathbb{N}}, \{c'_i\}_{i\in\mathbb{N}}$, that is, for all $i \in \mathbb{N}$, we obtain $(\gamma, \gamma', p) \Downarrow_o \hat{\gamma}_i, \hat{\gamma}'_i$ with $c_i \triangleq \hat{\gamma}_i(y)$ and $c'_i \triangleq \hat{\gamma}'_i(y)$. Define estimators $T_m \triangleq \frac{1}{m}\sum_{i=1}^m c_i$ and $T'_m \triangleq \frac{1}{m}\sum_{i=1}^m c'_i$ for $m \in \mathbb{N}$. Then, $(T_m)_{m\in\mathbb{N}}$ consists of unbiased estimators for $\theta \triangleq [\![p]\!](\gamma)(y)$, and it is strongly consistent for θ. Likewise, $(T'_m)_{m\in\mathbb{N}}$ consists of unbiased estimators for $\theta' \triangleq \sum_{z\in Var} \frac{d}{dt}[\![p]\!](\gamma[z \mapsto t])(y)\big|_{t=\gamma(z)} \cdot \gamma'(z)$, and it is strongly consistent for θ'.* «

We call a program *well-typed* if it satisfies a statically-determined, conservative type system that ensures that (i) no if statements occur after an integral and (ii) integrals are not nested (i.e., no variable that depends on a variable assigned to an integral can be present in the integrand). Appendix E formalizes the type system. The next theorem states that all well-typed programs induce strongly consistent estimators for the denotational semantics. Note that this result applies to all the programs in the evaluation (Appendices 2 and 6).

Theorem 5.7 (Consistency). *Let p be a well-typed program, $x \in Var$ be an arbitrary variable, and $\gamma, \gamma' \in Ctx$ be contexts with $[\![p]\!]\gamma \neq \text{err}$. Separate runs of the operational semantics produce (i.i.d.) random variables $\{c_i\}_{i\in\mathbb{N}}$ and $\{c_i'\}_{i\in\mathbb{N}}$, that is, for all $i \in \mathbb{N}$, we obtain $(\gamma, \gamma', p) \Downarrow_o \hat{\gamma}_i, \hat{\gamma}_i'$ with $c_i \triangleq \hat{\gamma}_i(x)$ and $c_i' \triangleq \hat{\gamma}_i'(x)$. Then, the family of estimators $T_m \triangleq \frac{1}{m}\sum_{i=1}^m c_i$ is strongly consistent for $[\![p]\!](\gamma)(x)$, and $T_m' \triangleq \frac{1}{m}\sum_{i=1}^m c_i'$ is strongly consistent for $\sum_{z\in Var} \frac{d}{dt}[\![p]\!](\gamma[z \mapsto t])(x)\big|_{t=\gamma(z)} \cdot \gamma'(z)$.* «

6 Evaluation

We implement the operational semantics of SingularFlow (Appendix 5) in JAX [11]. Users can write programs with singular integrals and use all the standard Python/JAX primitives (e.g., `jax.jit`, `jax.grad`, for loops, neural networks). Our implementation introduces a `singular_integrate` primitive that performs Monte Carlo integration and uses the `custom_vjp` jax primitive to define a custom derivative. JAX provides automatic differentiation for all other primitives in our implementation (e.g., arithmetic primitives).

We evaluate our implementation on two tasks: evaluating and differentiating singular integrals, and solving singular integral equations using feedforward neural networks. As a case study for the first task, we consider the finite Hilbert transform (Appendix 6.1), a fundamental integral transform used in science and engineering. For the second task, we solve singular integral equations—fundamental models in science and engineering [23, 44]. In particular, we provide three case studies: a model for the flow of air around a wing in aerodynamics (Appendix 6.2), and two models of the displacement of a crack in fracture mechanics (Appendix 6.3).

We selected these benchmarks as they represent important use cases of singular integrals in science and engineering (e.g., [23, 40, 44]). The historical uses of singular integrals include solving Cauchy-Riemann and Laplace equations [16, 34]. A more comprehensive list of problems modeled by singular integrals is given in Appendices 7 and 8, Appendix G, and standard textbooks (e.g., [23, 44]).

We compare our results using SingularFlow to several baselines (e.g., standard Monte Carlo estimates, manually-derived approximations, closed-form solutions) and existing systems (e.g., Mathematica [85]). For Appendix 6.2 and Appendix 6.3 we use the same methodology for solving integral equations as in Appendix 2, and provide training curves in Appendix D. We ran experiments on a 2019 MacBook with an Intel Core i9-9980HK CPU and 32GB RAM.

6.1 Finite Hilbert Transform

The *Hilbert transform* is a fundamental integral transformation akin to the Fourier transform, which is defined in terms of a singular integral over $\mathbb{R}$. The *finite Hilbert transform* is its finite version where the integral is taken over a finite domain. Formally, the finite Hilbert transform of a smooth function $f : \mathbb{R} \to \mathbb{R}$ is defined as follows, using the Cauchy principal value integral [40, Section 11]:

$$T_a f(s) \triangleq -\frac{1}{\pi} C \int_{-a}^{a} \frac{f(u)}{u-s} du, \tag{10}$$

where $a > 0$ and $-a < s < a$. The finite Hilbert transform (and the Hilbert transform in general) is used widely in diverse areas of science and engineering, including optics [27], scattering [10, 20],

Table 1. We calculate the finite Hilbert transform: $T_1 f(s) \triangleq -\frac{1}{\pi} C \int_{-1}^{1} \frac{f(u)}{u-s} du$ for $s = \frac{1}{2}$.

$f(u)$	Ground Truth	Ours	Standard	Mathematica
u	-0.462	$-0.46 \pm 4.1 \times 10^{-4}$	-3.6 ± 7.3	-0.46
u^2	-0.231	$-0.23 \pm 6.1 \times 10^{-4}$	-1.8 ± 3.7	-0.23
e^u	-0.291	$-0.29 \pm 9.9 \times 10^{-4}$	-11 ± 24	-0.29
ue^u	-0.894	$-0.89 \pm 5.6 \times 10^{-4}$	-6.1 ± 12	-0.89
$\sin u$	-0.409	$-0.41 \pm 3.7 \times 10^{-4}$	-3.4 ± 7.0	-0.41
$\cos u$	0.458	$0.46 \pm 1.0 \times 10^{-3}$	-5.1 ± 13	0.46

Table 2. The result of computing the derivative $\frac{d}{ds} T_1 f(s) = -\frac{1}{\pi} \mathcal{H} \int_{-1}^{1} \frac{f(u)}{(u-s)^2} du$ for $s = \frac{1}{2}$. Ground Truth corresponds to a by-hand derivation of the integrals into either a closed form or a form with an accurate numerical solution in [40]. PV is the Cauchy Principal value integral interpretation of the derivative rather than the correct Hadamard finite part interpretation. Standard is directly numerically estimating the integral. The Mathematica column shows the results of running the benchmarks in Mathematica [85]. In the Mathematica column, we mark a result with a dash if Mathematica does not return a numerical result.

$f(u)$	Ground Truth	Ours	PV	Standard	Mathematica
u	0.774	$0.77 \pm 8.1 \times 10^{-4}$	-5.9×10^{2}	-4.4×10^{3}	0.77
u^2	-7.47×10^{-2}	$-7.4 \times 10^{-2} \pm 8.1 \times 10^{-4}$	-2.9×10^{2}	-2.2×10^{3}	-7.4×10^{-2}
e^u	1.52	$1.5 \pm 9.9 \times 10^{-4}$	-1.9×10^{3}	-1.5×10^{4}	-
ue^u	0.468	$0.47 \pm 1.1 \times 10^{-3}$	-9.7×10^{2}	-7.3×10^{3}	-
$\sin u$	0.816	$0.82 \pm 1.0 \times 10^{-3}$	-5.6×10^{2}	-4.2×10^{3}	0.82
$\cos u$	0.868	$0.87 \pm 3.7 \times 10^{-4}$	-1.0×10^{3}	-7.7×10^{3}	0.87

and electrochemistry [81]; in geology to study seismic waves [28]; and in electrical engineering to study circuits [15], telecommunications [77], and control theory [74].

The finite Hilbert transform (and its derivative) often does not admit a closed-form formula, requiring the numerical estimation of the transform (and its derivatives) in many cases [40, Section 14]. Inspired by this, we estimate the finite Hilbert transform and its derivative. We estimate the finite Hilbert transform for six smooth functions King [40, Table 14.6]. In the following tables, we present the mean and standard deviations across 10 runs using 10,000 samples.

Estimation of the Finite Hilbert Transform. Table 1 depicts the estimates of $T_1 f(s)$ at $s = \frac{1}{2}$ for different functions f. The Ground Truth column shows the exact values (in 3-digit precision) computed using closed-form formulas King [40, Table 14.6]. The Ours column uses SingularFlow to produce correct results, accounting for the singularity at $u = s$ in Equation (10). The Standard column uses standard Monte Carlo integration, producing incorrect results because it ignores the singularity. Mathematica [85] produces an accurate numerical estimate (`N`) after singularity-aware symbolic integration (`Integrate` with the option `PrincipalValue->True`) [86].

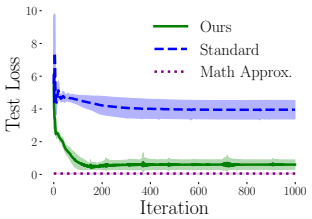
Estimation of the Derivative. Table 2 depicts estimates of the derivative $\frac{d}{ds} T_1 f(s)$ at $s = 1/2$ for different functions f. We hand-derived closed-form formulas for the Ground Truth derivatives of the finite Hilbert transform. The Ours column uses SingularFlow to compute the derivative, where SingularFlow automatically computes the Hadamard finite part integral (with a singularity at $u = s$). The result is correct up to the second bit of precision. PV uses the principal value interpretation of the result of commuting the derivative and the integral, which is incorrect. PV and Standard compute results that are orders of magnitude different from the correct result.

We run the experiments in Table 2 in Mathematica by setting the option `PrincipalValue->True` in the `Integrate` function, taking the derivative, evaluating the result at $s = 1/2$, and wrapping the result in the `N` function. Mathematica produces correct results for x, x^2, $\sin x$, and $\cos x$,

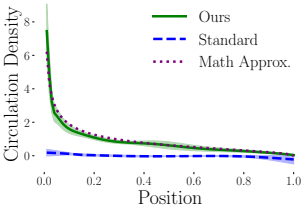
Table 3. Timing results, averaged across benchmarks, for the median and standard deviation of the runtime (in ms) for the finite Hilbert transform and its derivative as well as the optimized versions (with `jax.jit`).

Benchmark	Standard	Ours	Standard (w. jit)	Ours (w. jit)
Finite Hilbert Transform	6.3 ± 0.22	21 ± 1.5	0.97 ± 0.47	1.2 ± 0.19
Deriv. Finite Hilbert Transform	30 ± 17	140 ± 7.1	0.96 ± 0.44	0.9 ± 0.41

(a) The NACA 0012 airfoil.



(b) Loss over iterations.



(c) Learned solution.

Fig. 9. In aerodynamics, the flow of air around a wing can be modeled by a singular integral equation. Figure 9a shows a symmetrical, thin airfoil (model NACA 0012). We compare a Monte Carlo method that accounts for the singularity (Ours) to standard Monte Carlo integration (Standard) and a by-hand derivation of a mathematical approximation (Math Approx.). Figure 9b shows the loss over iterations and Figure 9c shows the function predicted by the three methods. Our method converges to a loss (squared error) that is less than the other methods and the predicted function is close to the math approximation.

but does not fully simplify results for e^x and xe^x. Mathematica does not support the Hadamard finite part integral and therefore does not produce the desired result. We confirm this by using `NIntegrate` with `PrincipalValue->True`, with `Exclusions->1/2`, and with the singularity listed with the domain of integration, as specified in the documentation [86].

Computation Time. Table 3 shows the timing results for the finite Hilbert transform and its derivative. We run all benchmarks 25 times and report the median and standard deviation of runtimes (with a single warm up iteration discarded). Each benchmark uses 10,000 samples to estimate the integral. We include the timing results to show that the overhead of using SingularFlow is not prohibitive. For the sake of completeness, we include more detailed timing results in Appendix D.1.

6.2 Airfoil Equation

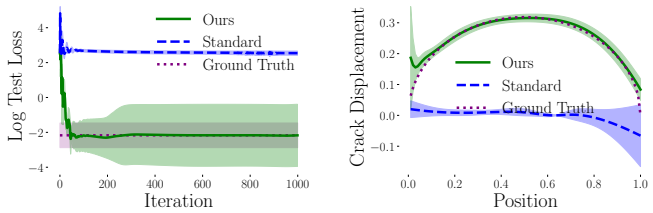
Figure 9a shows a symmetrical, thin airfoil (model NACA 0012). It is symmetrical about the x-axis and therefore easier to analyze mathematically than the thin airfoil in Appendix 2. We compare a Monte Carlo method that accounts for the singularity (Ours) to standard Monte Carlo integration (Standard) and a by-hand derivation of a mathematical approximation (Math Approx.).

Results. The results for Figure 9b are similar to those in Figure 2b in that the Ours loss is similar to the loss in Math Approx.

Figure 9c is also similar to the results in Figure 2c. Because the airfoil is symmetric, one of the assumptions in the derivation of the approximation of the airfoil equation applies, which means that the approximation should be closer to the ground truth than in the case of the airfoil in Figure 2c. Indeed, Ours and Math Approx. produce results that are closer to each other than results than in Figure 2c, while Standard produces results that are far from both of them.

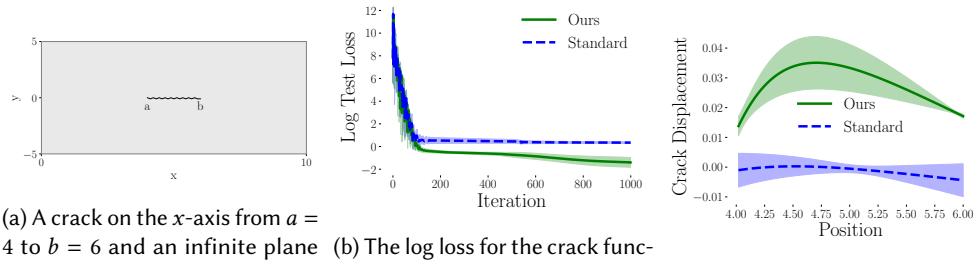
6.3 Crack Problem

A *crack problem* is a problem involving the discontinuous process by which an object splits resulting in an opening, the sliding two pieces of material, or the tearing of an object. Mechanical engineers



(a) Loss for crack function [35]. (b) Learned solution.

Fig. 10. In fracture mechanics, the displacement of a crack can be modeled by a singular integral equation. We compare a Monte Carlo method that accounts for the singularity (Ours) to standard Monte Carlo integration (Standard) and the ground truth closed-form solution (Ground Truth). Figure 10a shows the loss over iterations and Figure 10b shows the function predicted by the three methods. The neural approaches initially have high loss, but the loss of Ours decreases and matches the ground truth closely, while Standard stagnates and produces a poor model of the crack displacement.



(a) A crack on the x-axis from $a = 4$ to $b = 6$ and an infinite plane strip in the y-axis.

(b) The log loss for the crack function in Kaya and Erdogan [38].

(c) Learned solution.

Fig. 11. Figure 11a shows a crack on the x-axis from $a = 4$ to $b = 6$ and an infinite plane strip in the y-axis (truncated for display). We compare a Monte Carlo method that accounts for the singularity (Ours) to a standard method (Standard). Figure 11b shows the loss over iterations. Figure 11c shows the function predicted by the two methods. Ours converges to a loss that is orders of magnitude lower than Standard.

study crack problems to accurately model the failure of materials such as metals and concrete [38]. Characterization of cracks due to material manufacture, processing, or machining is important for ensuring the safety of structures as well as ensuring that manufactured goods with imperfections that maintain structural integrity. The discontinuity in stress leads to singularities in modeling.

1D Crack Problem. Harold Page Starr [35, Example 3.2] provides the following model:

$$-C\int_0^1 \frac{\sigma(t)}{t-x}dt + \int_0^1 \frac{\sigma(t)}{t+x}dx = 4x - 2\sqrt{x+x^2}, \tag{11}$$

where $\sigma(t)$ is the *crack displacement*, which is the distance separating the two materials at a given point t. In this singular integral equation, the goal is to solve for the unknown $\sigma(t)$. The equation $\sigma(t) = \frac{2}{\pi}\sqrt{t-t^2}$ is the closed-form solution to this problem [35].

Figure 10a shows the log loss over iterations for training a neural network to solve the crack problem. Standard Monte Carlo integration (Standard) slightly increases the loss throughout training, while our method (Ours) converges to a low loss. The neural network converges to a smaller squared-error loss than ground truth (Ground Truth). Figure 10b depicts the functions produced by Ours and by Math Approx. Standard predicts a function that is close to the constant zero, while Ours predicts a function that is an arc and looks similar to Ground Truth. While there is a small anomaly near zero, we show that the cause is a single outlier run and that upping the number of samples from 50 to 500 removes the anomaly (see Appendix D.3).

2D Crack Problem. Kaya and Erdogan [38] model this problem with the singular integral equation:

$$\mathcal{H}\int_a^b \frac{V(t)}{(t-x)^2}dt + \int_a^b V(t)K_0(t,x)dt = -\pi\frac{1+\kappa}{2\mu}p(x),$$

where $a = 4$ and $b = 6$ are the start and end of the crack, $\kappa = 3 - 4 * \nu = 2$ is the elastic constant with $\nu = 0.25$ is the Poisson ratio for granite, μ is the shear modulus, $p(x) = 1$ is the surface traction representing constant load, and the kernel $K_0(t,x) = -\frac{1}{(t+x)^2}$ is simplified from [38, Equations (62-66)]. The unknown $V(x)$ is the displacement of the crack opening, i.e., the size of the crack. In contrast to the previous integral equations, this one has Hadamard finite part integral rather than a Cauchy principal value integral. We could not find a closed-form solution in the literature and therefore do not have a ground truth solution.

Figure 11a depicts a crack on the x-axis from $a = 4$ to $b = 6$ and an infinite plane strip in the y-axis. Figure 11b shows the log loss over iterations for training a neural network to solve the crack problem. The SingularFlow implementation (Ours) approaches a low loss, while standard Monte Carlo integration (Standard) stays roughly constant at a high loss. Figure 11c depicts the functions predicted by Ours and Standard. Standard predicts a function that seems to be a constant, while Ours predicts a function that is more complex and varies with the input.

7 Related Work

We organize the related work into work from the programming languages community and work from the machine learning community. We provide further discussion of related math, numerical methods, and applications to computer graphics in Appendix G. To our knowledge, our paper is the first to provide a semantics of programs with singular integrals as well as their derivatives.

Standard Integrals, Discontinuities, and Derivatives. A line of work studies differentiable programming languages with support for integration [4, 61, 75]. Expressing programs using integrals can be easier to manipulate than expressing them using sampling [67, 75]. In problems specified using integration, discontinuities caused by control flow lead to challenges in modeling and differentiation [17, 36, 46, 49, 58, 66, 76]. Researchers have addressed these challenges in various ways, in specific contexts ranging from root finding to rendering to path planning [4, 7, 37, 61, 75, 87].

Probabilistic programs are programs that involve sampling and denote a measure [5, 41, 83]. Executing a probabilistic program (i.e., performing inference) often involves estimating the integral of the measure to compute a statistic such as the expected value of a function [8, 14, 18, 25, 72]. An advantage of this interface is that it simplifies expressing certain computations such as a random walk, where the measure is difficult to manually construct and integrate. Recent works have analyzed existing inference algorithms and/or developed new ones to ensure correctness in the presence of discontinuities [6, 39, 47, 48, 50, 51, 70, 84, 88].

None of the above works study semantics of programs with *singular* integrals or their *derivatives*.

Neural Solvers for Singular Integral Equations. Physically-informed neural networks (PINNs) are an approach to solving integral equations using neural networks [73]. Researchers have applied this framework to problems with singular integrals arising in fields such as potential theory and electrostatics [33, 54, 80]. However, these works *hand-derive* Riemann integrals from a *specific* class of singular integrals (i.e., principal value integrals), and differentiate them with respect to *nonsingular* variables (written as x, y in our work). As we show in Example 3.16, naively differentiating these integrals (or their estimators) with respect to singular variables is generally incorrect. We could not find open-source implementations of these works to compare against our approach.

We use PINNs to solve singular integral equations. SingularFlow *automatically* estimates a *larger* class of singular integrals, and differentiates them possibly with respect to *singular* variables.

8 Limitations and Future Work

In this section, we discuss the limitations of our work and directions for future research.

Multivariate and Nested Integrals. Multivariate integrals with singularities, $\int_S f(\mathbf{x})/g(\mathbf{x})d\mathbf{x}$, where S is a multivariate domain of integration, have the problem that singularities may now lay along a surface $g(\mathbf{x}) = 0$ rather than a point. If the integrand is separable in the variables of integration, then the problem can be reduced to our system. Future work could identify such integrands automatically and work to expand the theory to handle the case where the singularity is not separable. Perhaps the Schwartz kernel theorem is a good starting point [30].

Multiple Singularities. The extension to multiple singularities in the integrand is possible. An implementation could use a different sampling approach for different terms in the integrand. Perhaps it could tag samples in a way similar to Michel et al. [61].

More General Singularities. In this work, we studied singularities of the form $f(x, \theta)/(x - s)^k$, where k is a positive integer. Kutt [43] studies the case where k is a positive real number. Prior work claims that a general class of functions can be used as a change of variables for these singular integrals [64]. Future work could extend the language to support these more general singularities.

Another opportunity to generalize handling singularities is *Hadamard regularization*, which defines the integral of a function with a singularity coincident to a bound of integration [43]. The approach is to take the limit approaching the singularity, analytically integrate, and then add in a regularization factor that cancels out the singularity. For example, $⨍_0^1 \frac{1}{x}dx \triangleq \lim_{\epsilon\to 0} \int_\epsilon^1 \frac{1}{x}dx + \ln\epsilon = \ln 1 = 0$, which cancels the $-\ln\epsilon$ singularity by adding $\ln\epsilon$. A challenge is that Hadamard regularization requires the analytical integral and suffers from some pathologies. For instance, Hadamard regularization is not invariant to changes of variables. Concretely, if $y = 2x$ then $⨍_0^1 \frac{1}{x}dx$ should equal $⨍_0^2 \frac{1}{y}dy$, but $⨍_0^1 \frac{1}{x}dx = \ln 1$ is not equal to $⨍_0^2 \frac{1}{y}dy \triangleq \lim_\epsilon \int_\epsilon^2 \frac{1}{y}dy - \ln\epsilon = \ln 2$. More broadly, there is a rich literature on singular integral operators [2, 12, 23, 29, 44, 79].

Broader Applications. Support for automatic differentiation opens up a variety of additional applications such as solving inverse problems [17, 50, 52, 87]. Some inverse problems most relevant to the applications that we study are designing the wing of a plane to optimize for a desired amount of lift [21] or optimizing a material so that cracks propagate slowly when they occur [3]. To highlight one very general example, it is possible to solve integral equations on infinite domains by mapping the infinite domain to a finite one with a singularity using the *Kelvin transform* [65]. We provide additional discussion in Appendix G.

9 Conclusions

This paper introduced the first formal semantics for a programming language with support for singular integrals. In doing so, we provided a self-contained and rigorous presentation of the mathematical theory of singular integrals using only elementary calculus, including results that we had not seen proven rigorously using elementary techniques.

We provided a denotational and operational semantics for SingularFlow, and proved that the two semantics agree in that the operational semantics is an unbiased and finite-variance estimator of the denotational semantics and averaging these estimators leads to a strongly-consistent estimator family. We further defined a derivative operation that applies to SingularFlow programs and proved that it is correct. We evaluated SingularFlow by using it to implement the Hilbert transform as well as to solve singular integral equations arising in aerodynamics and mechanical engineering.

In the future, we hope our work will help researchers better formalize and reason about integral equations, and lead to flexible, ubiquitous tools for solving problems in science and engineering.

10 Data-Availability Statement

The artifact associated with this paper is available online [60].

Acknowledgements

We would like to thank Andrew Lumsdaine for his feedback and encouragement in early versions of this work. We would like to thank Michael Carbin for his guidance on the project including helping us navigate the transition from studying variational inference discussed in Appendix F to singular integral equations. We thank Logan Weber, Charles Yuan, Ellie Cheng, Tian Jin, and Teodoro Collins for their feedback on the paper. We thank the anonymous reviewers for their feedback.

J. Michel was supported in part by the National Science Foundation (CCF-1751011, CCF-1918839), the Sloan Foundation, and SRC JUMP 2.0 (CoCoSys). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. H. Yang was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) (No. RS-2023-00279680).

References

[1] Ira H Abbott, Albert E Von Doenhoff, and Louis Stivers Jr. 1945. *Summary of airfoil data.* Technical Report No. 824. National Advisory Committee for Aeronautics (NACA). https://ntrs.nasa.gov/citations/19930090976

[2] Helmut Abels. 2012. *Pseudodifferential and Singular Integral Operators: An Introduction with Applications.* De Gruyter. doi:10.1515/9783110250312

[3] A.M. Afsar, M. Anisuzzaman, and J.I. Song. 2009. Inverse problem of material distribution for desired fracture characteristics in a thick-walled functionally graded material cylinder with two diametrically-opposed edge cracks. *Engineering Fracture Mechanics* 76, 7 (2009). doi:10.1016/j.engfracmech.2008.12.003

[4] Sai Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically Differentiating Parametric Discontinuities. *ACM Transactions on Graphics* 40, 4, Article 107 (2021). doi:10.1145/3450626.3459775

[5] Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva (Eds.). 2020. *Foundations of Probabilistic Programming.* Cambridge University Press. doi:10.1017/9781108770750

[6] McCoy R. Becker, Alexander K. Lew, Xiaoyan Wang, Matin Ghavami, Mathieu Huot, Martin C. Rinard, and Vikash K. Mansinghka. 2024. Probabilistic Programming with Programmable Variational Inference. *Proceedings of the ACM on Programming Languages* 8, PLDI, Article 233 (2024). doi:10.1145/3656463

[7] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. 2020. Learning with Differentiable Pertubed Optimizers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS).* https://dl.acm.org/doi/10.5555/3495724.3496521

[8] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. 2019. Pyro: deep universal probabilistic programming. *Journal of Machine Learning Research* 20, 28 (2019). https://dl.acm.org/doi/10.5555/3322706.3322734

[9] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe and. 2017. Variational Inference: A Review for Statisticians. *J. Amer. Statist. Assoc.* 112, 518 (2017). doi:10.1080/01621459.2017.1285773

[10] M. M. Block and R. N. Cahn. 1985. High-energy $p\overline{p}$ and pp forward elastic scattering and total cross sections. *Reviews of Modern Physics* 57, 2 (1985). doi:10.1103/RevModPhys.57.563

[11] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.* https://github.com/jax-ml/jax

[12] A. P. Calderon and A. Zygmund. 1952. On the existence of certain singular integrals. *Acta Mathematica* 88 (1952). doi:10.1007/BF02392130

[13] Harry W Carlson and Robert J Mack. 1980. Studies of leading-edge thrust phenomena. *Journal of Aircraft* 17, 12 (1980). doi:10.2514/3.57981

[14] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan: A Probabilistic Programming Language. *Journal of Statistical Software* 76, 1 (2017). doi:10.18637/jss.v076.i01

[15] J. R. Carson. 1926. The Heaviside operational calculus. *Bull. Amer. Math. Soc.* 32, 1 (1926). https://www.ams.org/journals/bull/1926-32-01/S0002-9904-1926-04162-8/

[16] AL Cauchy. 1826. Sur un nouveau genre de calcul analogue au calcul infinitésimal. *Oeuvres Complet d'Augustin Cauchy, Gauthier-Villars, Paris* (1826).

[17] Swarat Chaudhuri and Armando Solar-Lezama. 2010. Smooth Interpretation. In *Programming Language Design and Implementation*. doi:10.1145/1806596.1806629

[18] Marco F. Cusumano-Towner, Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. 2019. Gen: a general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. doi:10.1145/3314221.3314642

[19] R. de L. Kronig. 1926. On the Theory of Dispersion of X-Rays. *Journal of the Optical Society of America* 12, 6 (1926). doi:10.1364/JOSA.12.000547

[20] Alexandre Deur, Stanley J. Brodsky, and Guy F. de Téramond. 2019. The Spin Structure of the Nucleon. *Reports on Progress in Physics* 82, 7 (2019). doi:10.1088/1361-6633/ab0b8f

[21] A.M. Elizarov and E.V. Fedorov. 1990. Airfoil optimization by the method of inverse boundary-value problems. *Journal of Applied Mathematics and Mechanics* 54, 4 (1990). doi:10.1016/0021-8928(90)90058-I

[22] Conal Elliott. 2009. Beautiful differentiation. In *International Conference on Functional Programming*. doi:10.1145/1596550.1596579

[23] Ricardo Estrada and Ram P Kanwal. 2000. *Singular integral equations*. Springer Science & Business Media. doi:10.1007/978-1-4612-1382-6

[24] Leandro Farina, Marcos R.S. Ferreira, and Victor Péron. 2016. The airfoil equation on near disjoint intervals: Approximate models and polynomial solutions. *J. Comput. Appl. Math.* 298, 15 (2016). doi:10.1016/j.cam.2015.11.024

[25] Tor Erlend Fjelde, Kai Xu, David Widmann, Mohamed Tarek, Cameron Pfiffer, Martin Trapp, Seth D. Axen, Xianda Sun, Markus Hauru, Penelope Yong, Will Tebbutt, Zoubin Ghahramani, and Hong Ge. 2025. Turing.jl: a general-purpose probabilistic programming language. *ACM Transactions on Probabilistic Machine Learning* (2025). doi:10.1145/3711897

[26] G.B. Folland. 1999. *Real Analysis: Modern Techniques and Their Applications* (second ed.). John Wiley & Sons.

[27] Mark Fox. 2010. *Optical Properties of Solids* (second ed.). Oxford University Press.

[28] Walter I. Futterman. 1962. Dispersive Body Waves. *Journal of Geophysical Research* 67, 13 (1962). doi:10.1029/JZ067i013p05279

[29] I. M. Gelfand and Georgi E. Shilov. 1964. *Generalized Functions, Volume 1: Properties And Operations*. Academic Press.

[30] I. M. Gelfand and N. Ya. Vilenkin. 1964. *Generalized Functions, Volume 4: Applications of Harmonic Analysis*. Academic Press.

[31] Mark Gillespie, Denise Yang, Mario Botsch, and Keenan Crane. 2024. Ray Tracing Harmonic Functions. *ACM Transactions on Graphics* 34, 4, Article 99 (2024). doi:10.1145/3658201

[32] Andreas Griewank and George F. Corliss (Eds.). 1991. *Automatic differentiaion of algorithms: Theory, implementation and application*. SIAM.

[33] Rui Guo, Zhichao Lin, Tao Shan, Maokun Li, Fan Yang, Shenheng Xu, and Aria Abubakar. 2021. Solving Combined Field Integral Equation With Deep Neural Network for 2-D Conducting Object. *IEEE Antennas and Wireless Propagation Letters* 20, 4 (2021). doi:10.1109/LAWP.2021.3056460

[34] J. Hadamard. 1932. Le problème de Cauchy et les équations aux dérivées partielles linéaires hyperboliques. Paris: Hermann & Cie.

[35] Jr. Harold Page Starr. 1991. *On the Numerical Solution of One-Dimensional Integral and Differential Equations*. Ph.D. thesis. Yale University. https://apps.dtic.mil/sti/citations/ADA250388

[36] Mathieu Huot, Alexander K. Lew, Vikash K. Mansinghka, and Sam Staton. 2023. ωPAP Spaces: Reasoning Denotationally About Higher-Order, Recursive Probabilistic and Differentiable Programs. In *Proceedings of the 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. doi:10.1109/LICS56636.2023.10175739

[37] Daniel Johnson and Ronald Fedkiw. 2024. Addressing discontinuous root-finding for subsequent differentiability in machine learning, inverse problems, and control. *J. Comput. Phys.* 497 (2024). doi:10.1016/j.jcp.2023.112624

[38] A. C. Kaya and F. Erdogan. 1987. On the solution of integral equations with strongly singular kernels. *Quart. Appl. Math.* 45, 1 (1987). https://www.ams.org/journals/qam/1987-45-01/S0033-569X-1987-0885170-6/

[39] Basim Khajwal, C.-H. Luke Ong, and Dominik Wagner. 2023. Fast and Correct Gradient-Based Optimisation for Probabilistic Programming via Smoothing. In *Proceedings of the32nd European Symposium on Programming (ESOP)*. doi:10.1007/978-3-031-30044-8_18

[40] Frederick W. King. 2009. *Hilbert Transforms: Volume 1*. Cambridge University Press. doi:10.1017/CBO9780511721458

[41] Dexter Kozen. 1981. Semantics of probabilistic programs. *J. Comput. System Sci.* 22, 3 (1981). doi:10.1016/0022-0000(81)90036-2

[42] H. A. Kramers. 1926. Wellenmechanik und halbzahlige Quantisierung. *Zeitschrift für Physik* 39 (1926). doi:10.1007/BF01451751

[43] Helmut Richard Kutt. 1975. *On the numerical evaluation of finite-part integrals involving an algebraic singularity.* Ph.D. thesis. Stellenbosch University. http://hdl.handle.net/10019.1/67934

[44] EG Ladopoulos. 2000. *Singular integral equations: linear and non-linear theory and its applications in science and engineering.* Springer-Verlag. doi:10.1007/978-3-662-04291-5

[45] Serge Lang. 1997. *Undergraduate analysis* (2nd ed.). Springer Science & Business Media. doi:10.1007/978-1-4757-2698-5

[46] Jacob Laurel, Rem Yang, Gagandeep Singh, and Sasa Misailovic. 2022. A dual number abstraction for static analysis of Clarke Jacobians. *Proceedings of the ACM on Programming Languages* 6, POPL, Article 56 (2022). doi:10.1145/3498718

[47] Wonyeol Lee, Xavier Rival, and Hongseok Yang. 2023. Smoothness Analysis for Probabilistic Programs with Application to Optimised Variational Inference. *Proceedings of the ACM on Programming Languages* 7, POPL, Article 12 (2023). doi:10.1145/3571205

[48] Wonyeol Lee, Hangyeol Yu, Xavier Rival, and Hongseok Yang. 2019. Towards Verified Stochastic Variational Inference for Probabilistic Programs. *Proceedings of the ACM on Programming Languages* 4, POPL, Article 16 (2019). doi:10.1145/3371084

[49] Wonyeol Lee, Hangyeol Yu, Xavier Rival, and Hongseok Yang. 2020. On correctness of automatic differentiation for non-differentiable functions. In *Neural Information Processing Systems*. https://dl.acm.org/doi/10.5555/3495724.3496288

[50] Wonyeol Lee, Hangyeol Yu, and Hongseok Yang. 2018. Reparameterization gradient for non-differentiable models. In *International Conference on Neural Information Processing Systems*. https://dl.acm.org/doi/10.5555/3327345.3327459

[51] Alexander K. Lew, Mathieu Huot, Sam Staton, and Vikash K. Mansinghka. 2023. ADEV: Sound Automatic Differentiation of Expected Values of Probabilistic Programs. *Proceedings of the ACM on Programming Languages* 7, POPL, Article 5 (2023). doi:10.1145/3571198

[52] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Transactions on Graphics* 37, 6 (2018). doi:10.1145/3272127.3275109

[53] I. M. Longman. 1958. On the Numerical Evaluation of Cauchy Principal Values of Integrals. *Math. Tables Aids Comput.* 12, 63 (1958). doi:10.2307/2002022

[54] Liyao Lyu, Zhen Zhang, Minxin Chen, and Jingrun Chen. 2022. MIM: A deep mixed residual method for solving high-order partial differential equations. *J. Comput. Phys.* 452 (2022). doi:10.1016/j.jcp.2021.110930

[55] K.W. Mangler. 1952. *Improper Integrals in Theoretical Aerodynamics.* Her Majesty's Stationery Office. https://reports.aerade.cranfield.ac.uk/handle/1826.2/107

[56] P. A. Martin and F. J. Rizzo. 1996. Hypersingular Integrals: How Smooth Must the Density be? *Internat. J. Numer. Methods Engrg.* 39, 4 (1996). doi:10.1002/(SICI)1097-0207(19960229)39:4<687::AID-NME876>3.0.CO;2-S

[57] P. A. Martin, F. J. Rizzo, and Fritz Joseph Ursell. 1989. On Boundary Integral Equations for Crack Problems. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 421, 1861 (1989). doi:10.1098/rspa.1989.0014

[58] Damiano Mazza and Michele Pagani. 2021. Automatic differentiation in PCF. *Proceedings of the ACM on Programming Languages* 5, POPL, Article 28 (2021). doi:10.1145/3434309

[59] R. Mertig, M. Böhm, and A. Denner. 1991. Feyn Calc - Computer-algebraic calculation of Feynman amplitudes. *Computer Physics Communications* 64, 3 (1991). doi:10.1016/0010-4655(91)90130-D

[60] Jesse Michel, Wonyeol Lee, and Hongseok Yang. 2024. Semantics of Integrating and Differentiating Singularities (v1.0.3). DOI: https://doi.org/10.5281/zenodo.15213473.

[61] Jesse Michel, Kevin Mu, Xuanda Yang, Sai Praveen Bangaru, Elias Rojas Collins, Gilbert Bernstein, Jonathan Ragan-Kelley, Michael Carbin, and Tzu-Mao Li. 2024. Distributions for Compositionally Differentiating Parametric Discontinuities. *Proceedings of the ACM on Programming Languages* 8, OOPSLA1, Article 126 (2024). doi:10.1145/3649843

[62] Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Transactions on Graphics* 42, 4, Article 82 (2023). doi:10.1145/3592400

[63] Giovanni Monegato. 1994. Numerical evaluation of hypersingular integrals. *J. Comput. Appl. Math.* 50, 1–3 (1994). doi:10.1016/0377-0427(94)90287-9

[64] Giovanni Monegato. 2009. Definitions, properties and applications of finite-part integrals. *J. Comput. Appl. Math.* 229, 2 (2009). doi:10.1016/j.cam.2008.04.006

[65] Mohammad Sina Nabizadeh, Ravi Ramamoorthi, and Albert Chern. 2021. Kelvin transformations for simulations on infinite domains. *ACM Transactions on Graphics* 40, 4, Article 97 (2021). doi:10.1145/3450626.3459809

[66] Henrik Nilsson. 2003. Functional automatic differentiation with dirac impulses. In *International Conference on Functional Programming*.

[67] Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Du Phan, and Jonathan P. Chen. 2019. Functional Tensors for Probabilistic Programming. (2019). arXiv:1910.10775

[68] Alan V. Oppenheim and Ronald W. Schafer. 2009. *Discrete-Time Signal Processing* (3rd ed.). Prentice Hall Press.

[69] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*. https:

//dl.acm.org/doi/10.5555/3454287.3455008
[70] Max B. Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J. Maddison. 2020. Gradient estimation with stochastic softmax tricks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*. https://dl.acm.org/doi/10.5555/3495724.3496202
[71] Robert Piessens, Elise de Doncker-Kapenga, Christoph W Überhuber, and David K Kahaner. 1983. *Quadpack: a subroutine package for automatic integration*. Springer-Verlag. doi:10.1007/978-3-642-61786-7
[72] Oriol Abril Pla, Virgile Andréani, Colin Carroll, Larry Dong, Christopher Fonnesbeck, Maxim Kochurov, Ravin Kumar, Junpeng Lao, Christian C. Luhmann, Osvaldo A. Martin, Michael Osthege, Ricardo Vieira, Thomas V. Wiecki, and Robert Zinkov. 2023. PyMC: a modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science* 9, e1516 (2023). doi:10.7717/PEERJ-CS.1516
[73] M. Raissi, P. Perdikaris, and G.E. Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378 (2019). doi:10.1016/j.jcp.2018.10.045
[74] María M. Seron, Julio H. Braslavsky, and Graham C. Goodwin. 1997. *Fundamental Limitations In Filtering And Control*. Springer-Verlag. doi:10.1007/978-1-4471-0965-5
[75] Benjamin Sherman, Jesse Michel, and Michael Carbin. 2021. λ_S: Computable Semantics for Differentiable Programming with Higher-Order Functions and Datatypes. *Proceedings of the ACM on Programming Languages* 5, POPL, Article 3 (2021). doi:10.1145/3434284
[76] Benjamin Sherman, Luke Sciarappa, Adam Chlipala, and Michael Carbin. 2018. Computable decision making on the reals and other spaces: via partiality and nondeterminism. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. doi:10.1145/3209108.3209193
[77] Hervé Sizun. 2005. *Radio Wave Propagation for Telecommunication Applications*. Springer-Verlag. doi:10.1007/b137896
[78] Richard Mikael Slevinsky and Sheehan Olver. 2017. A fast and well-conditioned spectral method for singular integral equations. *J. Comput. Phys.* 332 (2017). doi:10.1016/j.jcp.2016.12.009
[79] Elias M. Stein. 1970. *Singular Integrals and Differentiability Properties of Functions*. Princeton University Press.
[80] Jia Sun, Yinghua Liu, Yizheng Wang, Zhenhan Yao, and Xiaoping Zheng. 2023. BINN: A deep learning approach for computational mechanics problems based on boundary integral equations. *Computer Methods in Applied Mechanics and Engineering* 410 (2023). doi:10.1016/j.cma.2023.116012
[81] Mirna Urquidi-Macdonald, Silvia Real, and Digby D. Macdonald. 1990. Applications of Kramers—Kronig transforms in the analysis of electrochemical impedance data—III. Stability and linearity. *Electrochimica Acta* 35, 10 (1990). doi:10.1016/0013-4686(90)80010-L
[82] A. W. van der Vaart. 1998. *Asymptotic Statistics*. Cambridge University Press. doi:10.1017/CBO9780511802256
[83] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. 2018. An Introduction to Probabilistic Programming. (2018). arXiv:1809.10756
[84] Dominik Wagner, Basim Khajwal, and Luke Ong. 2024. Diagonalisation SGD: Fast & Convergent SGD for Non-Differentiable Models via Reparameterisation and Smoothing. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*. https://proceedings.mlr.press/v238/wagner24a.html
[85] Wolfram Research, Inc. 2024. Mathematica, Version 14.0. https://www.wolfram.com/mathematica
[86] Wolfram Research, Inc. 2024. Singularity Handling. Wolfram Language & System Documentation Center. https://reference.wolfram.com/language/tutorial/NIntegrateIntegrationStrategies.html#122144792 Last visited on 4/25/2024.
[87] Yuting Yang, Connelly Barnes, Andrew Adams, and Adam Finkelstein. 2022. Aδ: Autodiff for Discontinuous Programs - Applied to Shaders. In *ACM Transactions on Graphics*, Vol. 41. Article 135. doi:10.1145/3528223.3530125
[88] Yuan Zhou, Bradley J. Gram-Hansen, Tobias Kohn, Tom Rainforth, Hongseok Yang, and Frank Wood. 2019. LF-PPL: A Low-Level First Order Probabilistic Programming Language for Non-Differentiable Models. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*. http://proceedings.mlr.press/v89/zhou19b.html

Received 2024-11-15; accepted 2025-03-06

A Proofs for Section 3

A.1 Proofs for Appendix 3.1

Lemma A.1. *For all smooth $f : \mathbb{R} \to \mathbb{R}$, if $f(s) = 0$, then $\lim_{u\to s} \frac{f(u)}{u-s}$ exists and equals $f'(s)$.* «

Proof. Let $g_s(u) = u - s$. We prove the lemma as follows:

$$\lim_{u\to s} \frac{f(u)}{g_s(u)} = \lim_{u\to s} \frac{f'(u)}{1} = f'(s),$$

where the first equality follows from L'Hopital's rule and the second equality follows from f being smooth (at least C^1). Here, L'Hopital's rule is applicable because $\lim_{u\to s} f(u) = \lim_{u\to s} g_s(u) = 0$, $\lim_{u\to s} g_s'(u) = 1 \neq 0$, and $\lim_{u\to s} f'(u)$ exists. □

Proposition 3.1. *For all smooth f, the integral $\int_a^b \frac{f(u)}{u-s} du$ exists if and only if $f(s) = 0$.* «

Proof. ($\Longrightarrow$) We prove the contrapositive: if $f(s) \neq 0$, then $\int_a^b \frac{f(u)}{u-s} du$ does not exist. Without loss of generality, there exists some $\epsilon > 0$ such that $f(s) = \epsilon$. By continuity, there exists $\delta \in (0, \min(b-s, s-a))$ such that for every $u \in B_\delta(s)$, $|f(u) - f(s)| < \frac{\epsilon}{2}$. Note that $\frac{\epsilon}{2} < f(u) < \frac{3\epsilon}{2}$ for all $u \in B_\delta(s)$. We split the integral into two parts:

$$\int_a^b \frac{f(u)}{u-s} du = \int_{[a,b]\setminus B_\delta(s)} \frac{f(u)}{u-s} du + \int_{B_\delta(s)} \frac{f(u)}{u-s} du.$$

The first integral exists because it is the integral of a function that is continuous over a compact set and so bounded, and the integral of a bounded function over a compact set always exists. But the second integral does not have a finite value because $\int_{B_\delta(s)} \left|\frac{f(u)}{u-s}\right| du \geq \frac{\epsilon}{2} \int_{B_\delta(s)} \left|\frac{1}{u-s}\right| du = \infty$.

($\Longleftarrow$) It is sufficient to show that the integrand $g(u) = f(u)/(u-s)$ is continuous on $[a, b]$ because such a continuous function on $[a, b]$ is bounded, and this boundedness ensures that the integral of the function over $[a, b]$ has a well-defined finite value. Since f is smooth and $f(s) = 0$, Lemma A.1 implies that g is indeed continuous on $[a, b]$. □

Proposition 3.3. *For every smooth $f : \mathbb{R} \to \mathbb{R}$, the following hold:*

(a) $C\int_a^b \frac{f(u)}{u-s} du$ is well-defined and

$$C\int_a^b \frac{f(u)}{u-s} du = \int_a^s \frac{f(u) - f(2s-u)}{u-s} du - \int_{2s-b}^a \frac{f(2s-u)}{u-s} du, \tag{4}$$

where the two Riemann integrals in Equation (4) are well-defined.

(b) The Cauchy principal value integral is a linear operator on smooth functions:

$$C\int_a^b \frac{c_1 f_1(u) + c_2 f_2(u)}{u-s} du = c_1 \cdot \left(C\int_a^b \frac{f_1(u)}{u-s} du\right) + c_2 \cdot \left(C\int_a^b \frac{f_2(u)}{u-s} du\right)$$

for all $c_1, c_2 \in \mathbb{R}$ and smooth $f_1, f_2 : \mathbb{R} \to \mathbb{R}$.

(c) If $\int_a^b \frac{f(u)}{u-s} du$ is well-defined, then $C\int_a^b \frac{f(u)}{u-s} du = \int_a^b \frac{f(u)}{u-s} du$. «

Proof. We prove each claim as follows.

Proof of (a). Recall the definition of the Cauchy principal value integral:

$$C\int_a^b \frac{f(u)}{u-s} du = \lim_{\epsilon\to 0^+} \left(\int_a^{s-\epsilon} \frac{f(u)}{u-s} du + \int_{s+\epsilon}^b \frac{f(u)}{u-s} du\right). \tag{12}$$

Here, the latter integral can be rewritten with the change of variables $v = 2s - u$ as follows:

$$\int_{s+\epsilon}^{b} \frac{f(u)}{u-s}du = \int_{2s-b}^{s-\epsilon} \frac{f(2s-v)}{s-v}dv = -\left(\int_{2s-b}^{a} \frac{f(2s-v)}{v-s}dv + \int_{a}^{s-\epsilon} \frac{f(2s-v)}{v-s}dv\right).$$

By substituting this back into (12), we obtain (a):

$$\begin{aligned} C\int_a^b \frac{f(u)}{u-s}du &= \lim_{\epsilon\to 0^+}\left(\int_a^{s-\epsilon} \frac{f(u)-f(2s-u)}{u-s}du - \int_{2s-b}^{a} \frac{f(2s-u)}{u-s}du\right) \\ &= \int_a^s \frac{f(u)-f(2s-u)}{u-s}du - \int_{2s-b}^{a} \frac{f(2s-u)}{u-s}du. \end{aligned} \tag{13}$$

The two Riemann integrals in (13) are well-defined because their integrands are continuous over the compact domain of integration: the first integrand $(f(u) - f(2s-u))/(u-s)$ is continuous by Lemma A.1 (with f being smooth and $f(s) - f(2s-s) = 0$) and the second integrand is continuous over the domain of integration because $a < s$ and $2s - b < s$ (by $s < b$). Further, the equality in (13) holds by the following fact: if $g : \mathbb{R} \to \mathbb{R}$ is continuous, then $G : \mathbb{R} \to \mathbb{R}$ defined by $G(t) = \int_a^t g(u)du$ is continuous.

Proof of (b). We apply the definition of the Cauchy principal value integral, use the linearity of the limit and the Riemann integral, and then apply the definition again:

$$\begin{aligned} &C\int_a^b \frac{c_1 f_1(u) + c_2 f_2(u)}{u-s}du \\ &= \lim_{\epsilon\to 0^+}\left(\int_a^{s-\epsilon} \frac{c_1 f_1(u) + c_2 f_2(u)}{u-s}du + \int_{s+\epsilon}^{b} \frac{c_1 f_1(u) + c_2 f_2(u)}{u-s}du\right) \\ &= c_1 \cdot \lim_{\epsilon\to 0^+}\left(\int_a^{s-\epsilon} \frac{f_1(u)}{u-s}du + \int_{s+\epsilon}^{b} \frac{f_1(u)}{u-s}du\right) + c_2 \cdot \lim_{\epsilon\to 0^+}\left(\int_a^{s-\epsilon} \frac{f_2(u)}{u-s}du + \int_{s+\epsilon}^{b} \frac{f_2(u)}{u-s}du\right) \\ &= c_1 \cdot \left(C\int_a^b \frac{f_1(u)}{u-s}du\right) + c_2 \cdot \left(C\int_a^b \frac{f_2(u)}{u-s}du\right). \end{aligned}$$

Proof of (c). We apply Proposition 3.1 to f, which gives $f(s) = 0$. From this, we obtain (c):

$$C\int_a^b \frac{f(u)}{u-s}du = \lim_{\epsilon\to 0^+}\left(\int_a^{s-\epsilon} \frac{f(u)}{u-s}du + \int_{s+\epsilon}^{b} \frac{f(u)}{u-s}du\right) = \int_a^s \frac{f(u)}{u-s}du + \int_s^b \frac{f(u)}{u-s}du,$$

where the last two Riemann integrals are well-defined because their integrands are continuous on $\mathbb{R}$ by Lemma A.1 (with f being smooth and $f(s) = 0$). The first equality above applies the definition of the Cauchy principal value integral, and the second equality holds by the aforementioned fact: the antiderivative of a continuous function is continuous. □

Proposition 3.6. *For every smooth function $f : \mathbb{R} \to \mathbb{R}$,*

$$C\int_a^b \frac{f(u)}{u-s}du = \begin{cases} \displaystyle\int_a^s \frac{f(u)-f(2s-u)}{u-s}du + \int_{2s-a}^{b} \frac{f(u)}{u-s}du & \text{if } s \in \left(a, \frac{a+b}{2}\right] \\ \displaystyle\int_s^b \frac{f(u)-f(2s-u)}{u-s}du + \int_a^{2s-b} \frac{f(u)}{u-s}du & \text{if } s \in \left[\frac{a+b}{2}, b\right). \end{cases} \tag{5}$$

«

Proof. Suppose $s \in \left[\frac{a+b}{2}, b\right)$. Let $a' = 2s - b$. Since $a \le a' < s$, we can split the integral into two:

$$C\int_a^b \frac{f(u)}{u-s}du = \int_a^{a'} \frac{f(u)}{u-s}du + C\int_{a'}^b \frac{f(u)}{u-s}du.$$

We then expand the second integral in the RHS by Definition 3.2:

$$C\int_{a'}^{b}\frac{f(u)}{u-s}du=\lim_{\epsilon\to 0}\left(\int_{a'}^{s-\epsilon}\frac{f(u)}{u-s}du+\int_{s+\epsilon}^{b}\frac{f(u)}{u-s}du\right).$$

By using the substitution $v=2s-u$, we have:

$$\begin{aligned}C\int_{a'}^{b}\frac{f(u)}{u-s}du&=\lim_{\epsilon\to 0}\left(-\int_{b}^{s+\epsilon}\frac{f(2s-v)}{(2s-v)-s}dv+\int_{s+\epsilon}^{b}\frac{f(u)}{u-s}du\right)\\&=\lim_{\epsilon\to 0}\left(-\int_{s+\epsilon}^{b}\frac{f(2s-v)}{v-s}dv+\int_{s+\epsilon}^{b}\frac{f(u)}{u-s}du\right)\\&=\lim_{\epsilon\to 0}\int_{s+\epsilon}^{b}\frac{f(u)-f(2s-u)}{u-s}du\\&=\int_{s}^{b}\frac{f(u)-f(2s-u)}{u-s}du,\end{aligned}$$

where the last equation follows from Lemma A.6, which states that the integrand can be extended to a smooth function, justifying the limit. This proves the desired equation. The proof for the other case (i.e., when $s\in\left(a,\frac{a+b}{2}\right]$) is analogous, so we omit it. □

A.2 Proofs for Appendix 3.2 (Technical Lemmas)

Lemma A.2 (Leibniz integral rule [45, Theorem 7.1]). *Let $f : U \times \mathbb{R} \to \mathbb{R}$ be a function of u and v for some open set $U \subseteq \mathbb{R}$ with $[a,b] \subseteq U$. If f and $\frac{\partial}{\partial v} f$ are continuous on $[a,b] \times \mathbb{R}$, then $g(v) \triangleq \int_a^b f(u,v)du$ is differentiable on $\mathbb{R}$ and*

$$\frac{d}{dv}\int_a^b f(u,v)du = \int_a^b \frac{\partial}{\partial v} f(u,v)du \qquad (v \in \mathbb{R}). \qquad «$$

Lemma A.3 (Taylor's theorem [45, Theorem 3.1]). *For all smooth $f : \mathbb{R} \to \mathbb{R}$ and integers $k \geq 0$,*

$$f(u) - \sum_{i=0}^{k} \frac{f^{(i)}(v)}{i!}(u-v)^i = \int_v^u \frac{f^{(k+1)}(t)}{k!}(u-t)^k dt \qquad (u, v \in \mathbb{R}). \qquad «$$

Lemma A.4. *Let $f : U \times \mathbb{R}^m \to \mathbb{R}$ for some open set $U \subseteq \mathbb{R}$ and integer $m \geq 1$ with $[a,b] \subseteq U$. If f is continuous on $[a,b] \times \mathbb{R}^m$, then the function $g(v_1,\ldots,v_m) \triangleq \int_a^b f(u,v_1,\ldots,v_m)du$ is well-defined and continuous on $\mathbb{R}^m$.* «

Proof. The proof is trivial when $a = b$, so assume $a < b$. By the continuity of f on the compact set $[a,b]$, g is well-defined on $\mathbb{R}^m$. To show the continuity of g, consider any $v \in \mathbb{R}^m$ and $\epsilon > 0$. We need to show that for some $\delta > 0$,

$$\|v - v'\|_2 < \delta \implies |g(v) - g(v')| < \epsilon \qquad (v, v' \in \mathbb{R}^m).$$

Let $S \triangleq [a,b] \times \{v' \in \mathbb{R}^m \mid \|v - v'\|_2 \leq 1\}$. Since f is continuous on the compact set S, f is uniformly continuous on S. Hence, there exists $\delta > 0$ such that $\|w - w'\|_2 < \delta$ implies $|f(w) - f(w')| < \epsilon/(b-a)$ for all $w, w' \in \mathbb{R}^{n+1}$. This δ satisfies the above claim as follows:

$$\left|g(v) - g(v')\right| = \left|\int_a^b f(u,v) - f(u,v')du\right| \leq \int_a^b \left|f(u,v) - f(u,v')\right|du < (b-a)\cdot\frac{\epsilon}{b-a} = \epsilon,$$

where the last inequality is by $\|(u,v) - (u,v')\|_2 \leq \|v - v'\|_2 < \delta$. □

Lemma A.5. *Let $f : U \times \mathbb{R}^m \to \mathbb{R}$ for some open set $U \subseteq \mathbb{R}$ and integer $m \geq 1$ with $[a,b] \subseteq U$. If f is smooth on $[a,b] \times \mathbb{R}^m$, then $g(v_1,\ldots,v_m) \triangleq \int_a^b f(u,v_1,\ldots,v_m)du$ is smooth on $\mathbb{R}^m$.* «

Proof. It suffices to prove that for all $n \in \mathbb{N}_0$ and $i_1,\ldots,i_n \in \{1,\ldots,m\}$,

(i) the function $\frac{\partial^n}{\partial v_{i_1}\cdots\partial v_{i_n}} g$ is well-defined and continuous on $\mathbb{R}^m$; and
(ii) for all integers $n \geq 0$, $i_1,\ldots,i_n \in \{1,\ldots,m\}$, and $(v_1,\ldots,v_m) \in \mathbb{R}^m$,

$$\frac{\partial^n}{\partial v_{i_1}\cdots\partial v_{i_n}} g(v_1,\ldots,v_m) = \int_a^b \frac{\partial^n}{\partial v_{i_1}\cdots\partial v_{i_n}} f(u,v_1,\ldots,v_m)du. \tag{14}$$

We prove this by induction on n.

- *Case $n = 0$.* In this case, $\frac{\partial^n}{\partial v_{i_1}\cdots\partial v_{i_n}} g = g$ and $\frac{\partial^n}{\partial v_{i_1}\cdots\partial v_{i_n}} f = f$. Hence, Equation (14) simply becomes the definition of g. Also, by applying Lemma A.4 to the definition of g (i.e., $g(v_1,\ldots,v_m) = \int_a^b f(u,v_1,\ldots,v_m)du$), we obtain that g is well-defined and continuous on $\mathbb{R}^m$. Here, the lemma is applicable because f is smooth on $U \times \mathbb{R}^m$.
- *Case $n \geq 1$.* Let $i_1,\ldots,i_n \in \{1,\ldots,m\}$. By induction hypothesis, the function $\frac{\partial^n}{\partial v_{i_2}\cdots\partial v_{i_n}} g$ is continuous on $\mathbb{R}^m$ and satisfies the following: for all $(v_1,\ldots,v_m) \in \mathbb{R}^m$,

$$\frac{\partial^{n-1}}{\partial v_{i_2}\cdots\partial v_{i_n}} g(v_1,\ldots,v_m) = \int_a^b \frac{\partial^{n-1}}{\partial v_{i_2}\cdots\partial v_{i_n}} f(u,v_1,\ldots,v_m)du.$$

By applying the Leibniz integral rule (Lemma A.2) to the above equation (more precisely, to the function $h(u, v_{i_1}) \triangleq \frac{\partial^{n-1}}{\partial v_{i_2} \cdots \partial v_{i_n}} f(u, v_1, \ldots, v_m)$), we obtain that for all $(v_1, \ldots, v_m) \in \mathbb{R}^m$,

$$\frac{\partial}{\partial v_{i_1}} \left(\frac{\partial^{n-1}}{\partial v_{i_2} \cdots \partial v_{i_n}} g \right) (v_1, \ldots, v_m) = \int_a^b \frac{\partial}{\partial v_{i_1}} \left(\frac{\partial^{n-1}}{\partial v_{i_2} \cdots \partial v_{i_n}} f \right) (u, v_1, \ldots, v_m) du, \tag{15}$$

where both sides are well-defined. Here, the Leibniz integral rule is applicable because h is smooth on $U \times \mathbb{R}$.

We now show (i) and (ii). First, Equation (15) shows that $\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} g$ is well-defined on $\mathbb{R}^m$. Second, Lemma A.4 applied to Equation (15) shows that $\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} g$ is continuous on $\mathbb{R}^m$. Here, the lemma is applicable because $\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_1}} f$ is smooth on $U \times \mathbb{R}^m$. Third, Equation (15) is exactly Equation (14). □

Lemma A.6. *For all smooth $f : \mathbb{R} \to \mathbb{R}$ and integers $k \geq 0$, the function*

$$g(u, v) \triangleq \frac{1}{(u - v)^{k+1}} \left(f(u) - \sum_{i=0}^{k} \frac{f^{(i)}(v)}{i!} (u - v)^i \right) \tag{16}$$

defined over $\mathbb{R}^2 \setminus \{(c, c) : c \in \mathbb{R}\}$ can be extended to a smooth function $\hat{g} : \mathbb{R}^2 \to \mathbb{R}$, which satisfies

$$\hat{g}(u, u) = \frac{f^{(k+1)}(u)}{k!} \qquad (u \in \mathbb{R}). \tag{«}$$

Proof. Let $f : \mathbb{R} \to \mathbb{R}$ be a smooth function and $k \geq 0$ be an integer. Then, for all $u, v \in \mathbb{R}$,

$$\begin{aligned} f(u) - \sum_{i=0}^{k} \frac{f^{(i)}(v)}{i!} (u - v)^i &= \int_v^u \frac{f^{(k+1)}(t)}{k!} (u - t)^k dt \\ &= \int_0^1 \frac{f^{(k+1)}\left(t'u + (1 - t')v\right)}{k!} \cdot \left((1 - t')(u - v)\right)^k \cdot (u - v) dt' \\ &= (u - v)^{k+1} \int_0^1 \frac{f^{(k+1)}\left(t'u + (1 - t')v\right)}{k!} (1 - t')^k dt'. \end{aligned} \tag{17}$$

Here, the first line is by Taylor's theorem (Lemma A.3), and the second line holds for the following reason: if $u = v$, then this line is trivial, and if $u \neq v$, then the change of variable $t' = (t - v)/(u - v)$ gives this line because $t = t'u + (1 - t')v$ and $dt = (u - v)dt'$.

We now define $\hat{g} : \mathbb{R}^2 \to \mathbb{R}$ as

$$\hat{g}(u, v) \triangleq \int_0^1 \frac{f^{(k+1)}\left(tu + (1 - t)v\right)}{k!} (1 - t)^k dt.$$

Then, $\hat{g}$ satisfies the claims of the lemma. First, $\hat{g}$ is an extension of g: by Equation (17), $\hat{g}$ satisfies Equation (16) for all $u, v \in \mathbb{R}$ with $u \neq v$. Second, $\hat{g}$ is smooth on $\mathbb{R}^2$ by Lemma A.5, which is applicable because the integrand of $\hat{g}$ is a smooth function of t, u, v on $\mathbb{R}^3$. Lastly, for all $u \in \mathbb{R}$, $\hat{g}(u, u) = \frac{f^{(k+1)}(u)}{k!} \int_0^1 (1 - t)^k dt = \frac{f^{(k+1)}(u)}{(k+1)!}$, as desired. □

A.3 Proofs for Appendix 3.2 (Main Results)

Proposition 3.7. *For all smooth $f : \mathbb{R} \to \mathbb{R}$ and integers $k \geq 0$, the integral $\int_a^b \frac{f(u)}{(u-s)^{k+1}} du$ exists if and only if $f^{(i)}(s) = 0$ for all $i \in \{0, \ldots, k\}$.* «

Proof. ($\Rightarrow$) We prove the contrapositive: if $f^{(i)}(s) \neq 0$ for some $0 \leq i \leq k$, then $\int_a^b \frac{f(u)}{(u-s)^{k+1}} du$ does not exist. Let i be the smallest index such that $f^{(i)}(s) \neq 0$. Without loss of generality, we assume that $f^{(i)}(s) > 0$. Since $f^{(0)}(s) = \ldots = f^{(i-1)}(s) = 0$, Lemma A.6 implies that the function $g : \mathbb{R} \to \mathbb{R}$ defined by $g(u) \triangleq f(u)/(u-s)^i$ is continuous on $\mathbb{R}$ with $g(s) = f^{(i)}(s)/i! > 0$. By the continuity of g, there exists $\delta \in (0, \min(b-s, s-a))$ such that $|g(u) - g(s)| < \frac{1}{2} g(s)$ for every $u \in B_\delta(s)$. Note that $\frac{1}{2} g(s) < g(u) < \frac{3}{2} g(s)$ for all $u \in B_\delta(s)$. We split the integral into two parts:

$$\int_a^b \frac{f(u)}{(u-s)^{k+1}} du = \int_{[a,b] \setminus B_\delta(s)} \frac{g(u)}{(u-s)^{k+1-i}} du + \int_{B_\delta(s)} \frac{g(u)}{(u-s)^{k+1-i}} du.$$

The first integral is bounded because it is the integral of a function that is continuous over a compact domain and is thus bounded over the domain. The second integral, however, does not exist because

$$\int_{B_\delta(s)} \left| \frac{g(u)}{(u-s)^{k+1-i}} \right| du \geq \frac{g(s)}{2} \int_{B_\delta(s)} \left| \frac{1}{(u-s)^{k+1-i}} \right| du = +\infty,$$

where the equality is from $g(s) > 0$ and $k + 1 - i \geq 1$.

($\Leftarrow$) It is sufficient to show that the integrand $h(u) \triangleq f(u)/(u-s)^{k+1}$ is continuous on $[a, b]$ because a continuous function over a compact set is bounded and the integral of a bounded function over a compact set has a finite well-defined value. Since f is smooth and $f^{(i)}(s) = 0$ for all $0 \leq i \leq k$, Lemma A.6 implies that h is indeed continuous on $[a, b]$. □

Proposition 3.9. *For every smooth $f : \mathbb{R} \to \mathbb{R}$ and integer $k \geq 1$, the following hold:*

(a) $\mathcal{H} \int_a^b \frac{f(u)}{(u-s)^{k+1}} du$ is well-defined and satisfies integration by parts:

$$\mathcal{H} \int_a^b \frac{f(u)}{(u-s)^{k+1}} du = \begin{cases} -\frac{f(u)}{u-s}\Big|_{u=a}^b + C \int_a^b \frac{f'(u)}{u-s} du & \text{if } k = 1 \\ -\frac{1}{k} \frac{f(u)}{(u-s)^k}\Big|_{u=a}^b + \frac{1}{k} \mathcal{H} \int_a^b \frac{f'(u)}{(u-s)^k} du & \text{if } k \geq 2. \end{cases} \tag{6}$$

(b) The Hadamard finite part value integral is a linear operator on smooth functions:

$$\mathcal{H} \int_a^b \frac{c_1 f_1(u) + c_2 f_2(u)}{(u-s)^{k+1}} du = c_1 \cdot \left(\mathcal{H} \int_a^b \frac{f_1(u)}{(u-s)^{k+1}} du \right) + c_2 \cdot \left(\mathcal{H} \int_a^b \frac{f_2(u)}{(u-s)^{k+1}} du \right)$$

for all $c_1, c_2 \in \mathbb{R}$ and smooth $f_1, f_2 : \mathbb{R} \to \mathbb{R}$.

(c) If $\int_a^b \frac{f(u)}{(u-s)^{k+1}} du$ is well-defined, then $\mathcal{H} \int_a^b \frac{f(u)}{(u-s)^{k+1}} du = \int_a^b \frac{f(u)}{(u-s)^{k+1}} du$. «

Proof. We prove each statement as follows.

Proof of (a): Well-definedness. We show that the Hadamard finite part integral is well-defined. Since the Hadamard finite part integral is defined as a constant times the k-th derivative of the Cauchy principal value integral (Definition 3.8), it is sufficient to show that $g(s) \triangleq C \int_a^b \frac{f(u)}{u-s} du$ is smooth on (a, b). By the linearity of the Cauchy principal value integral (Proposition 3.3(b)),

$$g(s) = C \int_a^b \frac{f(u) - f(s)}{u-s} du + C \int_a^b \frac{f(s)}{u-s} du$$

$$= \int_a^b \frac{f(u) - f(s)}{u - s} du + C \int_a^b \frac{f(s)}{u - s} du, \tag{18}$$

where the second line is by Proposition 3.3(c). Here, this proposition is applicable because the first integral in Equation (18) is well-defined: the integrand of this integral is smooth in u by Lemma A.6, so this integral is well-defined.

Given this, it is sufficient to show that each of the two integrals in Equation (18) is smooth in s. The first integral is smooth in s by Lemma A.5, which is applicable because the integrand of this integral is smooth in (u, s) on $\mathbb{R}^2$ by Lemma A.6. For the second integral, applying Proposition 3.3(a) yields

$$C \int_a^b \frac{f(s)}{u - s} du = \int_a^s \frac{f(s) - f(s)}{u - s} du - \int_{2s-b}^a \frac{f(s)}{u - s} du = f(s) \cdot \ln(s - u) \Big|_{u=a}^{2s-b} \tag{19}$$

where the last equality is from $a, 2s - b < s$ (since $a < s < b$). This implies that the second integral is also smooth in s. Hence, g is smooth on (a, b) and the Hadamard finite part integral is well-defined.

Proof of (a): Integration by parts. We show that the Hadamard finite part integral satisfies Equation (6), i.e., integration by parts. Equation (6) is equivalent to:

$$\mathcal{H} \int_a^b \frac{f(u)}{(u - s)^{k+1}} du = -\frac{1}{k} \frac{f(u)}{(u - s)^k} \Big|_{u=a}^b + \frac{1}{k!} \frac{\partial^{k-1}}{\partial s^{k-1}} \left(C \int_a^b \frac{f'(u)}{u - s} du \right), \tag{20}$$

where $\frac{\partial^0}{\partial s^0}$ denotes the identity function. The proof proceeds by induction on k.

Base case ($k = 1$). We first compute $\mathcal{H} \int_a^b \frac{f(u)}{(u-s)^2} du$ as follows:

$$\begin{aligned}
\mathcal{H} \int_a^b \frac{f(u)}{(u - s)^2} du &= \frac{d}{ds} \left(C \int_a^b \frac{f(u)}{u - s} du \right) \\
&= \frac{d}{ds} \left(\int_a^b \frac{f(u) - f(s)}{u - s} du + f(s) \cdot C \int_a^b \frac{1}{u - s} du \right) \\
&= \int_a^b \frac{d}{ds} \left(\frac{f(u) - f(s)}{u - s} \right) du + f'(s) \cdot C \int_a^b \frac{1}{u - s} du + f(s) \cdot \frac{d}{ds} \left(C \int_a^b \frac{1}{u - s} du \right) \\
&= \int_a^b \frac{-f'(s)(u - s) + f(u) - f(s)}{(u - s)^2} du + C \int_a^b \frac{f'(s)}{u - s} du \\
&\quad + f(s) \cdot \frac{d}{ds} \left(C \int_a^b \frac{1}{u - s} du \right).
\end{aligned}$$

Here, the first line is by the definition of the Hadamard finite part integral, and the second line is by Equation (18) and the linearity of the Cauchy principal value integral (Proposition 3.3(b)). The third line is by the Leibniz integral rule (Lemma A.2), and because $(f(u) - f(s))/(u - s)$ is smooth in u and s and $C \int_a^b \frac{1}{u-s} du$ is smooth in s (both of which were shown above). The last line is by the linearity of the Cauchy principal value integral.

We next compute $C \int_a^b \frac{f'(u)}{u-s} du$ as follows:

$$\begin{aligned}
C \int_a^b \frac{f'(u)}{u - s} du &= \int_a^b \frac{f'(u) - f'(s)}{u - s} du + C \int_a^b \frac{f'(s)}{u - s} du \\
&= \int_a^b \frac{-f'(s)(u - s) + f'(u)(u - s)}{(u - s)^2} du + C \int_a^b \frac{f'(s)}{u - s} du,
\end{aligned}$$

where the first line is by Equation (18) and the smoothness of f'.

By combining these results, we obtain the desired equality (i.e., Equation (6) for the $k = 1$ case):

$$\begin{aligned}
&\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^2}du - C\int_a^b \frac{f'(u)}{u-s}du \\
&= \int_a^b \frac{f(u) - f(s) - f'(u)(u-s)}{(u-s)^2}du + f(s)\cdot\frac{d}{ds}\left(C\int_a^b \frac{1}{u-s}du\right), \\
&= -\int_a^b \frac{f(s) - f(u) - f'(u)(s-u)}{(s-u)^2}du + f(s)\cdot\frac{d}{ds}\left(\ln\frac{b-s}{s-a}\right) \\
&= -\int_a^b \frac{d}{du}\left(\frac{f(s)-f(u)}{s-u}\right)du + f(s)\cdot\frac{s-a}{b-s}\cdot\frac{-(s-a)-(b-s)}{(s-a)^2} \\
&= -\frac{f(s)-f(u)}{s-u}\bigg|_{u=a}^b + f(s)\cdot\frac{b-a}{(s-b)(s-a)} \\
&= -\frac{f(u)-f(s)}{u-s}\bigg|_{u=a}^b + f(s)\cdot\frac{1}{s-u}\bigg|_{u=a}^b = -\frac{f(u)}{u-s}\bigg|_{u=a}^b.
\end{aligned}$$

Here, the third line is by Equation (19), and the second last line is by the fundamental theorem of calculus and the smoothness of $(f(s) - f(u))/(s-u)$ in u (which was shown above).

Inductive case ($k \geq 2$). The proof of the claim (i.e., Equation (6) for $k > 2$) is similar to the previous case:

$$\begin{aligned}
\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du &= \frac{1}{k}\frac{d}{ds}\left(\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^k}du\right) \\
&= \frac{1}{k}\frac{d}{ds}\left(-\frac{1}{k-1}\frac{f(u)}{(u-s)^{k-1}}\bigg|_{u=a}^b + \frac{1}{k-1}\mathcal{H}\int_a^b \frac{f'(u)}{(u-s)^{k-1}}du\right) \\
&= -\frac{1}{k}\frac{f(u)}{(u-s)^k}\bigg|_{u=a}^b + \frac{1}{k}\mathcal{H}\int_a^b \frac{f'(u)}{(u-s)^k}du,
\end{aligned}$$

where the first and third line follow from Definition 3.8 and the second line uses the induction hypothesis (Equation (20)).

Proof of (b). The linearity of the Hadamard finite part integral holds as follows:

$$\begin{aligned}
\mathcal{H}\int_a^b \frac{c_1 f_1(u) + c_2 f_2(u)}{(u-s)^{k+1}}du &= \frac{1}{k!}\frac{d^k}{ds^k}\left(C\int_a^b \frac{c_1 f_1(u) + c_2 f_2(u)}{u-s}du\right) \\
&= \frac{1}{k!}\frac{d^k}{ds^k}\left(c_1\cdot C\int_a^b \frac{f_1(u)}{u-s}du + c_2\cdot C\int_a^b \frac{f_2(u)}{u-s}du\right) \\
&= c_1\cdot\left(\mathcal{H}\int_a^b \frac{f_1(u)}{(u-s)^{k+1}}du\right) + c_2\cdot\left(\mathcal{H}\int_a^b \frac{f_2(u)}{(u-s)^2}du\right),
\end{aligned}$$

where the first line is by the definition of the Hadamard finite part integral, the second line is by the linearity of the Cauchy principal value integral (Proposition 3.3(b)), and the last line is by the linearity of differentiation and by the definition of the Hadamard finite part integral.

Proof of (c). Suppose that $\int_a^b \frac{f(u)}{(u-s)^{k+1}}du$ is well-defined. Then, Proposition 3.7 implies that $f^{(i)}(s) = 0$ for all $i \in \{0, \ldots, k\}$. From this, we can prove the desired equality as follows:

$$\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du = \frac{1}{k!}\frac{d^k}{ds^k}\left(C\int_a^b \frac{f(u)}{u-s}du\right) = \frac{1}{k!}\frac{d^k}{ds^k}\left(\int_a^b \frac{f(u)}{u-s}du\right)$$

$$= \frac{1}{k!}\int_a^b \frac{d^k}{ds^k}\left(\frac{f(u)}{u-s}\right)du = \frac{1}{k!}\int_a^b k!\frac{f(u)}{(u-s)^{k+1}}du = \int_a^b \frac{f(u)}{(u-s)^{k+1}}du.$$

Here, the first equality is by the definition of the Hadamard finite part integral, and the second equality is by Propositions 3.1 and 3.3 with $f(s) = 0$. The third equality is by the Leibniz integral rule (Lemma A.2) and the smoothness of $f(u)/(u-s)$ in u and s (by Lemma A.6 with $f(s) = 0$). □

Proposition 3.10. *For all integers $k \geq 1$ and smooth functions $f : \mathbb{R} \to \mathbb{R}$,*

$$\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du = \frac{1}{k!}C\int_a^b \frac{f^{(k)}(u)}{u-s}du - \sum_{i=1}^{k}\frac{(i-1)!}{k!}\frac{f^{(k-i)}(u)}{(u-s)^i}\Bigg|_{u=a}^{b}. \tag{7}$$

«

Proof. The proof proceeds by induction on k. For $k = 1$, the claimed equality follows immediately from Proposition 3.9. For $k \geq 2$, the desired equality can be shown as follows:

$$\begin{aligned}
&\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}}du \\
&= \frac{1}{k}\mathcal{H}\int_a^b \frac{f'(u)}{(u-s)^k}du - \frac{1}{k}\frac{f(u)}{(u-s)^k}\Bigg|_{u=a}^{b} \\
&= \frac{1}{k}\left(\frac{1}{(k-1)!}C\int_a^b \frac{(f')^{(k-1)}(u)}{u-s}du - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\frac{(f')^{(k-1-i)}(u)}{(u-s)^i}\Bigg|_{u=a}^{b}\right) - \frac{1}{k}\frac{f^{(0)}(u)}{(u-s)^k}\Bigg|_{u=a}^{b} \\
&= \frac{1}{k!}C\int_a^b \frac{f^{(k)}(u)}{u-s}du - \sum_{i=1}^{k}\frac{(i-1)!}{k!}\frac{f^{(k-i)}(u)}{(u-s)^i}\Bigg|_{u=a}^{b},
\end{aligned}$$

where the first equality is by Proposition 3.9 and the second equality is by induction hypothesis on $(k-1, f')$. □

A.4 Proofs for Appendix 3.3

Lemma A.7. *Let $m \geq 0$ be an integer and $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$ be a smooth function. Then, for all integers $k \geq 0$, the function*

$$g(u, v, w_1, \ldots, w_m) \triangleq \frac{1}{(u-v)^{k+1}} \left(f(u, w_1, \ldots, w_m) - \sum_{i=0}^{k} \frac{\frac{\partial^i}{\partial u^i} f(v, w_1, \ldots, w_m)}{i!} (u-v)^i \right) \tag{21}$$

defined over $(\mathbb{R}^2 \setminus \{(c, c) : c \in \mathbb{R}\}) \times \mathbb{R}^m$ can be extended to a smooth function $\hat{g} : \mathbb{R}^2 \times \mathbb{R}^m \to \mathbb{R}$. Moreover, $\hat{g}$ satisfies

$$\hat{g}(u, u, w_1, \ldots, w_m) = \frac{\frac{\partial^{k+1}}{\partial u^{k+1}} f(u, w_1, \ldots, w_m)}{k!} \qquad (u, w_1, \ldots, w_m \in \mathbb{R}).$$

«

Proof. This lemma is a generalization of Lemma A.6. The proof of this lemma is exactly the same as the proof of Lemma A.6, except that we now add extra variables $(w_1, \ldots, w_m)$ to the arguments of f, g, and $\hat{g}$. □

Proposition 3.12. *For every integer $m \geq 0$ and smooth function $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$, the function $g(s, v_1, \ldots, v_m) = C \int_a^b \frac{f(u, v_1, \ldots, v_m)}{u-s} du$ is smooth on $(a, b) \times \mathbb{R}^m$. Moreover, for every integer $n \geq 0$ and $i_1, \ldots, i_n \in \{1, \ldots, m\}$,*

$$\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} \left(C \int_a^b \frac{f(u, v_1, \ldots, v_m)}{u-s} du \right) = C \int_a^b \frac{\frac{\partial^n}{\partial v_{i_1} \cdots \partial v_{i_n}} f(u, v_1, \ldots, v_m)}{u-s} du. \tag{8}$$

«

Proof. We first prove smoothness and then (8).

Proof of smoothness. The proof is a generalization of (a part of) the proof for Proposition 3.9(a). By the linearity of the Cauchy principal value integral (Proposition 3.3(b)),

$$\begin{aligned} g(s, v) &= C \int_a^b \frac{f(u, v) - f(s, v)}{u-s} du + C \int_a^b \frac{f(s, v)}{u-s} du \\ &= \int_a^b \frac{f(u, v) - f(s, v)}{u-s} du + f(s, v) \cdot \ln(s-u) \Big|_{u=a}^{2s-b} \end{aligned} \tag{22}$$

for all $(s, v) \in (a, b) \times \mathbb{R}^m$, where the equality for the first integral is by Proposition 3.3(c), and the equality for the second integral is by the calculation we did in the proof of Proposition 3.9(a). Here, Proposition 3.3(c) is applicable because the first integral in (22) is well-defined: the integrand of this integral is smooth in u by Lemma A.6, so this integral is well-defined.

Given this, it suffices to show that the two terms in (22) are smooth in (s, v) on $(a, b) \times \mathbb{R}^m$. For the first term, which is an integral, Lemma A.7 implies that its integrand is smooth in (s, u, v) on $\mathbb{R} \times \mathbb{R} \times \mathbb{R}^m$. From this and Lemma A.5, the first term is smooth in (s, v) on $\mathbb{R} \times \mathbb{R}^m$. The second term is smooth in (s, v) on $(a, b) \times \mathbb{R}^m$ because $s \in (a, b)$ implies $a, 2s - b < s$.

Proof of (8). Since all partial derivatives of f (including higher-order ones) are smooth on $\mathbb{R} \times \mathbb{R}^m$, it suffices to show the following: for every $i \in \{1, \ldots, m\}$,

$$\frac{\partial}{\partial v_i} \left(C \int_a^b \frac{f(u, v_1, \ldots, v_m)}{u-s} du \right) = C \int_a^b \frac{\frac{\partial}{\partial v_i} f(u, v_1, \ldots, v_m)}{u-s} du$$

for all $(s, v_1, \ldots, v_m) \in (a, b) \times \mathbb{R}^m$. This claim holds as follows:

$$\frac{\partial}{\partial v_i} \left(C \int_a^b \frac{f(u, v_1, \ldots, v_m)}{u-s} du \right)$$

$$
\begin{aligned}
&= \frac{\partial}{\partial v_i}\left(\int_a^s \frac{f(u,v_1,\ldots,v_m)-f(2s-u,v_1,\ldots,v_m)}{u-s}du - \int_{2s-b}^a \frac{f(2s-u,v_1,\ldots,v_m)}{u-s}du\right) \qquad (23)\\
&= \int_a^s \frac{\partial}{\partial v_i}\left(\frac{f(u,v_1,\ldots,v_m)-f(2s-u,v_1,\ldots,v_m)}{u-s}\right)du - \int_{2s-b}^a \frac{\partial}{\partial v_i}\left(\frac{f(2s-u,v_1,\ldots,v_m)}{u-s}\right)du\\
&= \int_a^s \frac{\left(\frac{\partial}{\partial v_i}f\right)(u,v_1,\ldots,v_m)-\left(\frac{\partial}{\partial v_i}f\right)(2s-u,v_1,\ldots,v_m)}{u-s}du - \int_{2s-b}^a \frac{\left(\frac{\partial}{\partial v_i}f\right)(2s-u,v_1,\ldots,v_m)}{u-s}du\\
&= C\int_a^b \frac{\frac{\partial}{\partial v_i}f(u,v_1,\ldots,v_m)}{u-s}du,
\end{aligned}
$$

where the first and the last equalities are by Proposition 3.3(a), and the second equality is by the Leibniz integral rule (Lemma A.2). Here, the Leibniz rule is applicable because the two integrands in (23) are smooth in (u, v_i) on $\mathbb{R}^2$: the smoothness of the first integrand is by Lemma A.7, and that of the second integrand is by $a, 2s-b < s$ (due to $s \in (a,b)$). □

Proposition 3.13. *For every integer $m \geq 0$, smooth function $f : \mathbb{R}\times\mathbb{R}^m \to \mathbb{R}$, and integer $k \geq 1$, the function $g(s,v_1,\ldots,v_m) = \mathcal{H}\int_a^b \frac{f(u,v_1,\ldots,v_m)}{(u-s)^{k+1}}du$ is smooth on $(a,b)\times\mathbb{R}^m$. Moreover, for every integer $n \geq 0$ and $i_1,\ldots,i_n \in \{1,\ldots,m\}$,*

$$
\frac{\partial^n}{\partial v_{i_1}\cdots\partial v_{i_n}}\left(\mathcal{H}\int_a^b \frac{f(u,v_1,\ldots,v_m)}{(u-s)^{k+1}}du\right) = \mathcal{H}\int_a^b \frac{\frac{\partial^n}{\partial v_{i_1}\cdots\partial v_{i_n}}f(u,v_1,\ldots,v_m)}{(u-s)^{k+1}}du. \qquad (9)
$$

«

Proof. As in the proof of Proposition 3.12, we first prove smoothness and then (9).

Proof of smoothness. By Proposition 3.10,

$$
g(s,v) = \frac{1}{k!}C\int_a^b \frac{\frac{\partial^k}{\partial u^k}f(u,v)}{u-s}du - \sum_{i=1}^k \frac{(i-1)!}{k!}\frac{\frac{\partial^{k-i}}{\partial u^{k-i}}f(u,v)}{(u-s)^i}\Bigg|_{u=a}^b
$$

for all $(s,v) \in (a,b)\times\mathbb{R}^m$. By Proposition 3.12, the integral in the RHS is smooth in (s,v) on $(a,b)\times\mathbb{R}^m$. Further, by $s \in (a,b)$, the summation in the RHS is smooth in (s,v) on $(a,b)\times\mathbb{R}^m$. Hence, g is smooth on $(a,b)\times\mathbb{R}^m$.

Proof of (9). Since all partial derivatives of f (including higher-order ones) are smooth on $\mathbb{R}\times\mathbb{R}^m$, it suffices to show the following as in the proof of Proposition 3.12: for every $i \in \{1,\ldots,m\}$,

$$
\frac{\partial}{\partial v_i}\left(\mathcal{H}\int_a^b \frac{f(u,v_1,\ldots,v_m)}{(u-s)^{k+1}}du\right) = \mathcal{H}\int_a^b \frac{\frac{\partial}{\partial v_i}f(u,v_1,\ldots,v_m)}{(u-s)^{k+1}}du
$$

for all $(s,v_1,\ldots,v_m) \in (a,b)\times\mathbb{R}^m$. This claim holds as follows:

$$
\begin{aligned}
&\frac{\partial}{\partial v_i}\left(\mathcal{H}\int_a^b \frac{f(u,v_1,\ldots,v_m)}{(u-s)^{k+1}}du\right)\\
&= \frac{\partial}{\partial v_i}\left(\frac{1}{k!}C\int_a^b \frac{\frac{\partial^k}{\partial u^k}f(u,v_1,\ldots,v_m)}{u-s}du - \sum_{i=1}^k \frac{(i-1)!}{k!}\frac{\frac{\partial^{k-i}}{\partial u^{k-i}}f(u,v_1,\ldots,v_m)}{(u-s)^i}\Bigg|_{u=a}^b\right)\\
&= \frac{1}{k!}C\int_a^b \frac{\frac{\partial}{\partial v_i}\frac{\partial^k}{\partial u^k}f(u,v_1,\ldots,v_m)}{u-s}du - \sum_{i=1}^k \frac{(i-1)!}{k!}\frac{\frac{\partial}{\partial v_i}\frac{\partial^{k-i}}{\partial u^{k-i}}f(u,v_1,\ldots,v_m)}{(u-s)^i}\Bigg|_{u=a}^b\\
&= \frac{1}{k!}C\int_a^b \frac{\frac{\partial^k}{\partial u^k}\left(\frac{\partial}{\partial v_i}f\right)(u,v_1,\ldots,v_m)}{u-s}du - \sum_{i=1}^k \frac{(i-1)!}{k!}\frac{\frac{\partial^{k-i}}{\partial u^{k-i}}\left(\frac{\partial}{\partial v_i}f\right)(u,v_1,\ldots,v_m)}{(u-s)^i}\Bigg|_{u=a}^b
\end{aligned}
$$

$$= \mathcal{H}\int_a^b \frac{\frac{\partial}{\partial v_i} f(u, v_1, \ldots, v_m)}{(u-s)^{k+1}} du,$$

where the first and the last equalities are by Proposition 3.10, the second equality is by Proposition 3.12, and the third equality is by the smoothness of f. □

Proposition 3.14. *For every smooth function $f : \mathbb{R} \to \mathbb{R}$ and integer $n \geq 1$,*

$$\frac{d^n}{ds^n}\left(C\int_a^b \frac{f(u)}{u-s} du\right) = n! \cdot \mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{n+1}} du.$$

«

Proof. By the definition of the Hadamard finite part integral,

$$\frac{d^n}{ds^n}\left(C\int_a^b \frac{f(u)}{u-s} du\right) = n! \cdot \frac{1}{n!}\frac{d^n}{ds^n}\left(C\int_a^b \frac{f(u)}{u-s} du\right) = n! \cdot \mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{n+1}} du.$$

□

Proposition 3.15. *For every smooth function $f : \mathbb{R} \to \mathbb{R}$ and integers $n, k \geq 1$,*

$$\frac{d^n}{ds^n}\left(\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}} du\right) = \frac{(n+k)!}{k!} \cdot \mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{n+k+1}} du.$$

«

Proof. By the definition of the Hadamard finite part integral,

$$\begin{aligned}
\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{n+k+1}} du &= \frac{1}{(n+k)!}\frac{d^{n+k}}{ds^{n+k}}\left(C\int_a^b \frac{f(u)}{u-s} du\right) \\
&= \frac{1}{(n+k)!}\frac{d^n}{ds^n}\left(\frac{d^k}{ds^k}\left(C\int_a^b \frac{f(u)}{u-s} du\right)\right) \\
&= \frac{k!}{(n+k)!}\frac{d^n}{ds^n}\left(\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^{k+1}} du\right).
\end{aligned}$$

□

B Proofs for Section 4

Lemma 4.1. *For every expression e, its semantics $[\![e]\!]$ is smooth on Ctx.* «

Proof. The proof is by induction on the structure of e. A constant or a projection function (which is used to interpret variables) is smooth. The constant and variable cases are covered by this fact. Also, by assumption, the semantics of every function symbol h is smooth, and if all of functions $f : \mathbb{R}^m \to \mathbb{R}$ and $g_1, \ldots, g_m : Ctx \to \mathbb{R}$ are smooth, then their composition $\gamma \mapsto f(g_1(\gamma), \ldots, g_m(\gamma))$ is smooth by the chain rule. Hence, the claim of the lemma on the function application case from the induction hypothesis. □

Theorem 4.2. *For every program p and $\gamma \in Ctx$, if $[\![p]\!]\gamma \neq \mathrm{err}$, then $[\![p]\!]$ is smooth at γ.* «

Proof. The proof is by induction on the structure of p.

Case of $x = e$. To show that $[\![x = e]\!](\gamma) = \gamma[x \mapsto [\![e]\!]\gamma]$ is well-defined and smooth, it suffices to show that $[\![e]\!]$ is smooth in γ. This follows from Lemma 4.1.

Case of $y = \textbf{integral}\ (a, b)\ e/(x-s)^k\ \mathbf{d}x$. Since $y = \textbf{integral}\ (a, b)\ e/(x-s)^k\ \mathbf{d}x$ is not err, we are guaranteed that $s \notin \mathrm{FV}(e)$. It suffices to show that $c = ([\![y = \textbf{integral}\ (a, b)\ e/(x-s)^k\ \mathbf{d}x]\!]\gamma)(y)$ is well-defined and smooth at γ.

The function $g_\gamma(u) = [\![e]\!](\gamma[x \mapsto u])$ on $\mathbb{R}$ is smooth by Lemma 4.1. When $\gamma(s) \notin [a, b]$, the integral $\int_a^b \frac{g(u)}{(u-\gamma(s))^k} du$ is well-defined since its integrand is continuous on the closed interval $[a, b]$ and thus bounded on the interval, and the Riemann integral of a bounded continuous function over a closed interval is well-defined. The remaining case is $\gamma(s) \in (a, b)$. In this case, we use Propositions 3.3 and 3.9 and derive the well-definedness of $[\![p]\!]\gamma$, the first case is $\gamma(s) \in (a, b) \wedge k = 1$ and the second case is $\gamma(s) \in (a, b) \wedge k > 1$.

For the smoothness of $[\![p]\!]$, it suffices to show that for some open set U containing γ, $[\![p]\!]$ is smooth. When $\gamma(s) \in (a, b)$, we set $U \triangleq \{\gamma \in Ctx \mid \gamma(s) \in (a, b)\}$. Then, if $k = 1$, $[\![p]\!]$ is smooth on U by Proposition 3.12, and if $k > 1$, $[\![p]\!]$ is smooth on U by Proposition 3.13. It remains to deal with the case $\gamma(s) \notin [a, b]$. Let $U \triangleq \{\gamma \in Ctx \mid \gamma(s) \notin [a, b]\}$. The desired smoothness of $[\![p]\!]$ on U then follows from the repeated use of Lemma A.2.

Case of $p_1; p_2$. Since $[\![p_1; p_2]\!]\gamma$ is not err, we are guaranteed that $\gamma' = [\![p_1]\!]\gamma$ is not err. Furthermore, by the inductive hypothesis, it denotes a smooth function in the parameters at γ. We can repeat this argument for p_2 at γ'.

Case of $\textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2$. Since $[\![\textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2]\!]\gamma \neq \mathrm{err}$, we know that there is a ball B in parameter space around γ that lies exclusively within p_1 or exclusively within p_2 (this is because the region $S = \mathbb{R} \setminus \{0\}$ is open, so for point p, there exists an $\epsilon > 0$ such that $B_p(\epsilon) \subset S$). In either case, we can directly apply the inductive hypothesis to get the desired result.

Case of $\textbf{for}\ x\ \textbf{in range}(a, b, c) : p$. We prove this by induction on a. In the base case where $a \geq b$, this function is the identity, which is smooth. In the inductive case, where $a < b$, we have that $c > 0$, guaranteeing eventual termination, and that $[\![p]\!](\gamma[x \mapsto a]) \neq \mathrm{err}$ for any iteration because we know that the output is not err. □

C Proofs for Section 5

Lemma 5.2. *The derivative transformation is correct for every expression e, $n \in \mathbb{N}_0$, and $z \in Var$: $[\![D[e](z,n)]\!]\gamma = f^{(n)}(\gamma(z))$ for all $\gamma \in Ctx$, where $f : \mathbb{R} \to \mathbb{R}$ is defined by $f(u) = [\![e]\!](\gamma[z \mapsto u])$.* «

Proof. It is equivalent to prove the following: for all expressions e and $n \in \mathbb{N}_0$,

$$[\![D[e](z,n)]\!](\gamma[z \mapsto u]) = \frac{d^n}{du^n}\Big([\![e]\!](\gamma[z \mapsto u])\Big) \qquad (\gamma \in Ctx,\ u \in \mathbb{R}).$$

We prove this claim by induction on n and the structure of e. For $n = 0$, the claim holds because the 0th derivative of a function is the function itself. For $n \geq 1$, we have three cases for e. If e is c or x, then the claim holds because (i) the nth derivative of a constant function is 0, and (ii) the nth derivative of the identity function is $\mathbf{1}[n = 1 \wedge z = x]$. If e is $h(e_1, \ldots, e_m)$, then the claim holds as follows: for every $\gamma \in Ctx$ and $u \in \mathbb{R}$,

$$\begin{aligned}
&[\![D[h(e_1,\ldots,e_m)](z,n)]\!](\gamma[z \mapsto u]) \\
&= [\![D[dh_1 \cdot de_1 + \cdots + dh_m \cdot de_m](z,n-1)]\!](\gamma[z \mapsto u]) \\
&= \frac{d^{n-1}}{du^{n-1}}\left([\![dh_1 \cdot de_1 + \cdots + dh_m \cdot de_m]\!](\gamma[z \mapsto u])\right) \\
&= \frac{d^{n-1}}{du^{n-1}}\left(\sum_{i=1}^{m} [\![dh_i]\!](\gamma[z \mapsto u]) \cdot [\![de_i]\!](\gamma[z \mapsto u])\right) \\
&= \frac{d^{n-1}}{du^{n-1}}\left(\sum_{i=1}^{m} [\![\mathcal{D}_i h(e_1,\ldots,e_m)]\!](\gamma[z \mapsto u]) \cdot [\![D[e](z,1)]\!](\gamma[z \mapsto u])\right) \\
&= \frac{d^{n-1}}{du^{n-1}}\left(\sum_{i=1}^{m} [\![\mathcal{D}_i h]\!]([\![e_1]\!](\gamma[z \mapsto u]),\ldots,[\![e_m]\!](\gamma[z \mapsto u])) \cdot \frac{d}{du}\left([\![e_i]\!](\gamma[z \mapsto u])\right)\right) \\
&= \frac{d^{n-1}}{du^{n-1}}\left(\frac{d}{du}\left([\![h]\!]([\![e_1]\!](\gamma[z \mapsto u]),\ldots,[\![e_m]\!](\gamma[z \mapsto u]))\right)\right) \\
&= \frac{d^{n-1}}{du^{n-1}}\left(\frac{d}{du}\left([\![h(e_1,\ldots,e_m)]\!](\gamma[z \mapsto u])\right)\right) \\
&= \frac{d^{n}}{du^{n}}\left([\![h(e_1,\ldots,e_m)]\!](\gamma[z \mapsto u])\right).
\end{aligned}$$

Here, the second, fourth, and fifth lines are by the definition of $D[\cdot](z,n)$, $[\![\cdot]\!]$, dh_i, and de_i. The third line is by induction hypothesis on $n-1$. The fourth-to-last line is by the definition of $[\![\cdot]\!]$ and induction hypothesis on (n, e_i). The third-to-last line is by the chain rule and the assumption that $[\![\mathcal{D}_i h]\!]$ is the ith partial derivative of $[\![h]\!]$. The last two lines are by the definition of $[\![\cdot]\!]$ and differentiation. □

Lemma 5.1. *For every context γ and expression e, we have that $(\gamma, e) \Downarrow c$ if and only if $[\![e]\!]\gamma = c$.* «

Proof. The proof follows by induction on the structure of e. □

Theorem 5.5 (Unbiasedness and Finite Variance). *Let $p \equiv (y = \textbf{\textit{integral}}\ (a,b)\ e/(x-s)^k\ \boldsymbol{dx})$ and $\gamma, \gamma' \in Ctx$ with $[\![p]\!]\gamma \neq$ err. Let c, c' be estimators defined by $c \triangleq \hat{\gamma}(y)$ and $c' \triangleq \hat{\gamma}'(y)$, where $(\gamma, \gamma', p) \Downarrow_o \hat{\gamma}, \hat{\gamma}'$. Then, c and c' have finite variance, and are unbiased for $[\![p]\!](\gamma)(y)$ and $\sum_{z \in Var} \frac{d}{dt}[\![p]\!](\gamma[z \mapsto t])(y)\big|_{t=\gamma(z)} \cdot \gamma'(z)$, respectively.* «

Proof. We first show that c is an unbiased estimator for θ. Let $c_0 \triangleq \theta$. If $\gamma(s) \notin [a,b]$ and $a = b$, we have $c = 0$ by the operational semantics and also $c_0 = 0$ by the denotational semantics. Hence,

we have the desired $\mathbb{E}[c] = c_0$ in this case. If $\gamma(s) \in [a, b]$ and $a \neq b$, we show the unbiasedness as follows:

$$\begin{aligned}\mathbb{E}[c] = \mathbb{E}_{t \leftarrow \mathcal{U}(a,b)}\left[\frac{(b-a)\cdot [\![e]\!](\gamma[x \mapsto t])}{(t-\gamma(s))}\right] &= \int_a^b \frac{1}{b-a}\cdot\frac{(b-a)\cdot [\![e]\!](\gamma[x \mapsto t])}{(t-\gamma(s))}dt \\ &= \int_a^b \frac{[\![e]\!](\gamma[x \mapsto t])}{(t-\gamma(s))}dt \\ &= c_0,\end{aligned}$$

where the first equality follows from operational semantics and Lemma 5.1, the second from the definition of expectation, and the last equality from the denotational semantics. It remains to deal with the case that $\gamma(s) \in (a, b)$. Assume $\gamma(s) \in (a, b)$. If $k = 1$ and $\gamma(s) \in [(a+b)/2, b)$, we have

$$\mathbb{E}[c] = \mathbb{E}_{t \leftarrow \mathcal{U}(a,2\gamma(s)-b)}\left[\mathbb{E}_{t' \leftarrow \mathcal{U}(\gamma(s),b)}\left[\frac{((2\gamma(s)-b)-a)\cdot [\![e]\!](\gamma[x \mapsto t])}{(t-\gamma(s))}\right.\right. \tag{24}$$

$$\left.\left. + \frac{(b-\gamma(s))\cdot\left([\![e]\!](\gamma[x \mapsto t']) - [\![e]\!](\gamma[x \mapsto 2\gamma(s)-t'])\right)}{(t'-\gamma(s))}\right]\right] \tag{25}$$

$$= \mathbb{E}_{t' \leftarrow \mathcal{U}(a,2\gamma(s)-b)}\left[\frac{((2\gamma(s)-b)-a)\cdot [\![e]\!](\gamma[x \mapsto t])}{(t-\gamma(s))}\right] \tag{26}$$

$$+ \mathbb{E}_{t \leftarrow \mathcal{U}(\gamma(s),b)}\left[\frac{(b-\gamma(s))\cdot\left([\![e]\!](\gamma[x \mapsto t']) - [\![e]\!](\gamma[x \mapsto 2\gamma(s)-t'])\right)}{(t'-\gamma(s))}\right] \tag{27}$$

$$= \int_a^{2\gamma(s)-b} \frac{1}{(2\gamma(s)-b)-a}\cdot\frac{((2\gamma(s)-b)-a)\cdot [\![e]\!](\gamma[x \mapsto t])}{(t-\gamma(s))}dt \tag{28}$$

$$+ \int_{\gamma(s)}^b \frac{1}{b-\gamma(s)}\cdot\frac{(b-\gamma(s))\cdot\left([\![e]\!](\gamma[x \mapsto t']) - [\![e]\!](\gamma[x \mapsto 2\gamma(s)-t'])\right)}{(t'-\gamma(s))}dt' \tag{29}$$

$$= \int_a^{2\gamma(s)-b} \frac{[\![e]\!](\gamma[x \mapsto t])}{(t-\gamma(s))}dt + \int_{\gamma(s)}^b \frac{[\![e]\!](\gamma[x \mapsto t']) - [\![e]\!](\gamma[x \mapsto 2\gamma(s)-t'])}{(t'-\gamma(s))}dt' \tag{30}$$

$$= C\int_a^b \frac{[\![e]\!](\gamma[x \mapsto u])}{u-\gamma(s)}du \tag{31}$$

$$= c_0. \tag{32}$$

The first equality follows from the operational semantics and Lemma 5.1, the second and third from the definition of expectation, the fifth from Proposition 3.6, and the last from the denotational semantics. The case that $k = 1$ and $\gamma(s) \in (a, (a+b)/2]$ can be proved similarly. The remaining case is that $k > 1$ and $\gamma(s) \in (a, b)$. Assume $k > 1$ and $\gamma(s) \in (a, b)$. Let $e_i = D[e](x, i)$ for all $i \in \{0, \ldots, k-1\}$, and let $f : \mathbb{R} \to \mathbb{R}$ be the smooth function defined by $f(u) = [\![e]\!](\gamma[x \mapsto u])$. Then, the random variable c has the following form by Lemma 5.1, the operational semantics, and the correctness of the D operator:

$$\begin{aligned}c &= \frac{c'}{(k-1)!} - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\left(\frac{[\![e_{k-1-i}]\!][x \mapsto b]}{(b-\gamma(s))^i} - \frac{[\![e_{k-1-i}]\!][x \mapsto a]}{(a-\gamma(s))^i}\right) \\ &= \frac{c'}{(k-1)!} - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\left(\frac{f^{(k-1-i)}(b)}{(b-\gamma(s))^i} - \frac{f^{(k-1-i)}(a)}{(a-\gamma(s))^i}\right),\end{aligned}$$

where c' is a random variable such that

$$(\gamma, t, \textbf{integral}\ (a,b)\ e_{k-1}/(x-s)\ \mathbf{d}x) \Downarrow_o c'.$$

Using this observation, we derive the desired unbiasedness of c as follows:

$$\mathbb{E}_c[c] = \mathbb{E}_{c'}\left[\frac{c'}{(k-1)!} - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\left(\frac{f^{(k-1-i)}(b)}{(b-\gamma(s))^i} - \frac{f^{(k-1-i)}(a)}{(a-\gamma(s))^i}\right)\right] \tag{33}$$

$$= \frac{\mathbb{E}_{c'}[c']}{(k-1)!} - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\left(\frac{f^{(k-1-i)}(b)}{(b-\gamma(s))^i} - \frac{f^{(k-1-i)}(a)}{(a-\gamma(s))^i}\right) \tag{34}$$

$$= \frac{1}{(k-1)!}C\int_a^b \frac{f^{(k-1)}(u)}{u-\gamma(s)}du - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\left(\frac{f^{(k-1-i)}(b)}{(b-\gamma(s))^i} - \frac{f^{(k-1-i)}(a)}{(a-\gamma(s))^i}\right) \tag{35}$$

$$= \mathcal{H}\int_a^b \frac{f(u)}{(u-\gamma(s))^k}du \tag{36}$$

$$= \mathcal{H}\int_a^b \frac{[\![e]\!](\gamma[x\mapsto u])}{(u-\gamma(s))^k}du \tag{37}$$

$$= c_0. \tag{38}$$

Here the third equality follows from the unbiasedness of the operational semantics in the case of the Cauchy principal value integral, which we showed above. The fourth equality uses Proposition 3.10, and the last two equalities follow from the definition of f and the denotational semantics.

Next, we show that c has finite variance. We will prove that $|c| < B$ for some non-random constant $B > 0$. Note that the boundedness of c implies

$$Var(c) = \mathbb{E}[(c-\mathbb{E}[c])^2] \le 2\mathbb{E}[c^2] + 2(\mathbb{E}[c])^2 < 4B^2 < \infty,$$

that is, the finiteness of the variance of c.

Let

$$f : \mathbb{R} \to \mathbb{R}, \qquad f(u) \triangleq [\![e]\!](\gamma[x\mapsto u]).$$

Note that f is smooth. When $\gamma(s) \notin [a,b]$,

$$c = (b-a)\cdot\frac{f(t)}{t-\gamma(s)} \quad \text{for some random } t \in [a,b],$$

but the right-hand side of this equation is a continuous function on t evaluated at some point in the closed interval $[a,b]$, and so it is bounded. When $\gamma(s) \in [(a+b)/2, b)$ and $k = 1$,

$$c = \frac{((2\gamma(s)-b)-a)\cdot f(t)}{(t-\gamma(s))} + \frac{(b-\gamma(s))\cdot(f(t') - f(2\gamma(s)-t'))}{(t'-\gamma(s))}$$

for some random $t \in [a,(2\gamma(s)-b)]$ and $t' \in [\gamma(s), b]$. But the right-hand side of this equation is a continuous function on (t,t') evaluated at some pair in the closed rectangle $[a,(2\gamma(s)-b)]\times[\gamma(s),b]$, where the continuity with respect to t' follows from Lemma A.1. Thus, the right-hand side is bounded. The case that $\gamma(s) \in (a,(a+b)/2]$ and $k = 1$ can be handled similarly. The remaining case is that $k > 1$ and $\gamma(s) \in (a,b)$. In this case,

$$c = \frac{c'}{(k-1)!} - \sum_{i=1}^{k-1}\frac{(i-1)!}{(k-1)!}\left(\frac{f^{(k-1-i)}(b)}{(b-\gamma(s))^i} - \frac{f^{(k-1-i)}(a)}{(a-\gamma(s))^i}\right)$$

where c' is a random variable such that

$$(\gamma, t, \textbf{integral}\ (a,b)\ D[e](x,k-1)/(x-s)\ \mathbf{d}x) \Downarrow_o c'.$$

By what we have already shown for the $k = 1$ case, the random variable c' is bounded. Thus, c is bounded as well.

We now prove the case of the derivative. First, we expand the derivative:

$$\begin{aligned}\theta' &\triangleq \sum_{z \in Var} \frac{\partial [\![p]\!](\gamma[z \mapsto \hat{z}])(y)}{\partial \hat{z}} \gamma'(z) \\ &= \sum_{z \in Var} \frac{\partial [\![\textbf{integral } (a,b)\ e/(x-s)^k\ \mathbf{d}x]\!](\gamma[z \mapsto \hat{z}])(y)}{\partial \hat{z}} \gamma'(z).\end{aligned}$$

Next, the Integral-Assign rule in the operational semantics (Figure 5) maps y to $k \cdot c' \cdot \gamma'(s) + \sum_{z \in Var \wedge z \neq s} c'_z \cdot \gamma'(z)$ and does not change the other variables. Taking expectation of this expression and using the linearity of expectation, we have:

$$\mathbb{E}_{c', c'_z}\left[k \cdot c' \cdot \gamma'(s) + \sum_{z \in Var \wedge z \neq s} c'_z \cdot \gamma'(z)\right] = k \cdot \mathbb{E}_{c'}[c'] \cdot \gamma'(s) + \sum_{z \in Var \wedge z \neq s} \mathbb{E}_{c'_z}[c'_z] \cdot \gamma'(z). \tag{39}$$

Now we break the proof into cases.

Case of $s \in (a, b)$ and $k = 1$: Expanding the definition of γ'_{out}, we have:

$$\begin{aligned}\theta' &= \sum_{z \in Var} \frac{\partial \left(C \int_a^b \frac{g(u,z,\hat{z})}{x - \gamma(s)} du\right)}{\partial \hat{z}} \gamma'(z) \qquad \text{where } g(u, z, \hat{z}) \triangleq [\![e]\!](\gamma[x \mapsto u][z \mapsto \hat{z}]) \\ &= \frac{\partial \left(C \int_a^b \frac{g(u,z,\hat{z})}{x - \gamma(s)} du\right)}{\partial \gamma(s)} \gamma'(s) + \sum_{z \in Var \wedge z \neq s} \frac{\partial \left(C \int_a^b \frac{g(u,z,\hat{z})}{x - \gamma(s)} du\right)}{\partial \hat{z}} \gamma'(z) \\ &= \left(\mathcal{H} \int_a^b \frac{g(u, z, \hat{z})}{(x - \gamma(s))^2} du\right) \gamma'(s) + \sum_{z \in Var \wedge z \neq s} \left(C \int_a^b \frac{\frac{\partial g(u,z,\hat{z})}{\partial \hat{z}}}{x - \gamma(s)} du\right) \gamma'(z),\end{aligned}$$

where the last equation follows from Proposition 3.12 and Proposition 3.14. Since $k = 1$ and by Equation (24) and Lemma 5.2, we conclude that Equation (39) is equivalent to the above expression.

Case of $s \in (a, b)$ and $k > 1$: Expanding the definition of γ'_{out}, we have:

$$\begin{aligned}\theta' &= \sum_{z \in Var} \frac{\partial \left(\mathcal{H} \int_a^b \frac{g(u,z,\hat{z})}{(x - \gamma(s))^k} du\right)}{\partial \hat{z}} \gamma'(z) \qquad \text{where } g(u, z, \hat{z}) \triangleq [\![e]\!](\gamma[x \mapsto u][z \mapsto \hat{z}]) \\ &= \frac{\partial \left(\mathcal{H} \int_a^b \frac{g(u,z,\hat{z})}{(x - \gamma(s))^k} du\right)}{\partial \gamma(s)} \gamma'(s) + \sum_{z \in Var \wedge z \neq s} \frac{\partial \left(\mathcal{H} \int_a^b \frac{g(u,z,\hat{z})}{(x - \gamma(s))^k} du\right)}{\partial \hat{z}} \gamma'(z) \\ &= k \cdot \left(\mathcal{H} \int_a^b \frac{g(u, z, \hat{z})}{(x - \gamma(s))^{k+1}} du\right) \gamma'(s) + \sum_{z \in Var \wedge z \neq s} \left(\mathcal{H} \int_a^b \frac{\frac{\partial g(u,z,\hat{z})}{\partial \hat{z}}}{(x - \gamma(s))^k} du\right) \gamma'(z),\end{aligned}$$

where the last equation follows from Proposition 3.13 and Proposition 3.15. By Equation (33) and Lemma 5.2, we conclude that Equation (39) is equivalent to the above expression.

Case of $s \notin (a, b)$: Expanding the definition of γ'_{out}, we have:

$$\theta' = \sum_{z \in Var} \frac{\partial \left(\int_a^b \frac{g(u,z,\hat{z})}{(x - \gamma(s))^k} du\right)}{\partial \hat{z}} \gamma'(z) \qquad \text{where } g(u, z, \hat{z}) \triangleq [\![e]\!](\gamma[x \mapsto u][z \mapsto \hat{z}])$$

$$= \frac{\partial\left(\int_a^b \frac{g(u,z,\hat{z})}{(x-\gamma(s))^k}du\right)}{\partial\gamma(s)}\gamma'(s) + \sum_{z\in Var\wedge z\neq s}\frac{\partial\left(\int_a^b \frac{g(u,z,\hat{z})}{(x-\gamma(s))^k}du\right)}{\partial\hat{z}}\gamma'(z)$$

$$= k\cdot\left(\int_a^b \frac{g(u,z,\hat{z})}{(x-\gamma(s))^{k+1}}du\right)\gamma'(s) + \sum_{z\in Var\wedge z\neq s}\left(\int_a^b \frac{\frac{\partial g(u,z,\hat{z})}{\partial\hat{z}}}{(x-\gamma(s))^k}du\right)\gamma'(z),$$

where the last equation follows from Lemma A.2. The equivalence of Equation (39) and the above equation follows from the definition of the correctness of standard Monte Carlo integration and Lemma 5.2.

The proof bounded variance matches the primal case. □

Corollary 5.6. *Let $p \equiv (y = \textbf{\textit{integral}}\,(a,b)\ e/(x-s)^k\ \textbf{\textit{dx}})$ and $\gamma,\gamma' \in Ctx$ with $[\![p]\!]\gamma \neq \text{err}$. Separate runs of the operational semantics produce (i.i.d.) random variables $\{c_i\}_{i\in\mathbb{N}}, \{c_i'\}_{i\in\mathbb{N}}$, that is, for all $i \in \mathbb{N}$, we obtain $(\gamma,\gamma',p) \Downarrow_o \hat{\gamma}_i,\hat{\gamma}_i'$ with $c_i \triangleq \hat{\gamma}_i(y)$ and $c_i' \triangleq \hat{\gamma}_i'(y)$. Define estimators $T_m \triangleq \frac{1}{m}\sum_{i=1}^m c_i$ and $T_m' \triangleq \frac{1}{m}\sum_{i=1}^m c_i'$ for $m \in \mathbb{N}$. Then, $(T_m)_{m\in\mathbb{N}}$ consists of unbiased estimators for $\theta \triangleq [\![p]\!](\gamma)(y)$, and it is strongly consistent for θ. Likewise, $(T_m')_{m\in\mathbb{N}}$ consists of unbiased estimators for $\theta' \triangleq \sum_{z\in Var}\frac{d}{dt}[\![p]\!](\gamma[z\mapsto t])(y)\big|_{t=\gamma(z)}\cdot\gamma'(z)$, and it is strongly consistent for θ'.* «

Proof. Let an arbitrarily variable $x \in Var$ be given. Since all c_i's are independent and identically distributed, and c_1 has finite variance and its expectation is θ by Theorem 5.5, we can apply Proposition 5.4 and derive the desired conclusion, that is, the estimator family $(T_m)_{m\in\mathbb{N}}$ constructed by taking the average over the c_i's is strongly consistent for θ and consists of unbiased estimators for θ. After applying the linearity of expectation, the proof for the derivative for each term in the sum is similar to the above proof. □

Proposition C.1 ([82, Theorem 2.3]). *Let X be random element defined on $\mathbb{R}^k$. Let $g : \mathbb{R}^k \to \mathbb{R}^m$ be a continuous function. If the estimator family $(X_n)_{n\in\mathbb{N}}$ is consistent for X, then the estimator family $(g(X_n))_{n\in\mathbb{N}}$ is consistent for $g(X)$.* «

Lemma C.2. *For arbitrary γ, let $\theta \triangleq [\![p]\!]\gamma(y)$ and $\theta' \triangleq \sum_{z\in Var}[\![p]\!](\gamma[z\mapsto t])(y)\big|_{t=\gamma(z)}\gamma'(z)$. For every deterministic statement in p (everything rule but integral assignment) such that p does not denote* err *at γ, if $(\gamma,\gamma',p) \Downarrow_o \gamma_1,\gamma_1'$ then $\gamma_1(y) = \theta$ and $\gamma_1'(y) = \theta'$.* «

Proof. For assignment, $[\![x = e]\!](\gamma)$ is the identity for all $z \neq x$ and likewise for $(\gamma,\gamma',x = e) \Downarrow_o \gamma_1,\gamma_1'$. For $x = z$, we apply Lemma 5.1 to show equivalence in the primal case and Lemma 5.2 to show equivalence in the derivative case.

For sequential composition, by the inductive hypothesis we have that if $(\gamma,\gamma',p_1) \Downarrow_o \gamma_1,\gamma_1'$ then $\gamma_1(y) = \theta_1 \triangleq [\![p_1]\!](\gamma)(y)$ and $\gamma_1'(y) = \theta_1' \triangleq \sum_{z\in Var}[\![p_1]\!](\gamma[z\mapsto t])(y)\big|_{t=\gamma(z)}\gamma'(z)$ and if $(\gamma_1,\gamma_1',p_2) \Downarrow_o \gamma_2,\gamma_2'$ then $\gamma_2(y) = \theta_2 \triangleq [\![p_2]\!](\gamma_1)(y)$ and $\gamma_2'(y) = \theta_2' \triangleq \sum_{z\in Var}[\![p_1]\!](\gamma_1[z\mapsto t])(y)\big|_{t=\gamma_1(z)}\gamma_1'(z)$. Composing these two results gives the desired conclusion.

For conditionals, the proof breaks into two cases. When $c > 0$ in $(\gamma,e) \Downarrow c$, the operational semantics evaluates $(\gamma,\gamma',p_1) \Downarrow_o \gamma_1,\gamma_1'$. We can apply the inductive hypothesis to show the desired conclusion. When $c < 0$, the proof is analogous to the above.

The for-loop base case is the identity for both the operational and denotational semantics, and the recursive case holds by the inductive hypothesis. □

Lemma C.3. *Let $p \triangleq (p_1; z = e)$ be a well-typed program such that $z = e$ is a deterministic assignment and either $p_1 \triangleq (y = \textbf{\textit{integral}}\,(a,b)\ e_1/(x-s)^k\ \textbf{\textit{dx}})$ is an integral assignment or $p_1 \triangleq (y = e_1)$ an assignment that is not a deterministic assignment. For every $\gamma \in Ctx$ such that the*

denotation of p is not err *and every derivative context $\gamma' \in Ctx$, we have that*

$$[\![p]\!](\gamma) = [\![z = e; p_1]\!](\gamma).$$

Let $\theta \triangleq [\![p]\!](\gamma)(y)$ and $\theta' \triangleq \sum_{w \in Var} [\![p_1]\!](\gamma[w \mapsto t])(y)\big|_{t=\gamma(w)} \gamma'(w)$. Let the family of random variables $\{c_i\}_{i\in\mathbb{N}}$,$\{c_i'\}_{i\in\mathbb{N}}$, $\{d_i\}_{i\in\mathbb{N}}$, and $\{d_i'\}_{i\in\mathbb{N}}$ obtained by the operational semantics for each of the two above programs (specifically, $(\gamma, \gamma', p) \Downarrow_o \hat{\gamma}_i, \hat{\gamma}_i'$ and $(\gamma, \gamma', z = e; p_1) \Downarrow_o \tilde{\gamma}_i, \tilde{\gamma}_i'$ with $c_i = \hat{\gamma}_i(y)$, $c_i' = \hat{\gamma}_i(y)$, $d_i = \tilde{\gamma}_i(y)$, and $d_i' = \tilde{\gamma}_i(y)$) be given. Then, $T_m = \frac{1}{m}\sum_{i=1}^m c_i$ and $U_m = \frac{1}{m}\sum_{i=1}^m d_i$ are strongly consistent for θ and $T_m' = \frac{1}{m}\sum_{i=1}^m c_i'$ and $U_m' = \frac{1}{m}\sum_{i=1}^m d_i'$ are strongly consistent for θ'. Also, $\gamma_1(x) = \gamma_2(x)$ and $\gamma_1'(x) = \gamma_2'(x)$ for every $x \neq y$ such that $x \in Var$. «

Proof. We assert that $z \notin \mathrm{FV}(p_1)$ because we can always rename z to a variable not in $\mathrm{FV}(p_1)$. By the definition of sequential composition and assignment in the denotational semantics:

$$[\![p_1; z = e]\!](\gamma) = [\![z = e]\!](\gamma[y \mapsto c]) = (\gamma[y \mapsto c])[z \mapsto [\![e]\!]\gamma],$$

where c in the second equation is as defined in the denotational semantics for the integral case and $c = [\![e_1]\!]\gamma$ in the assignment case. Since $z = e$ satisfies the Deterministic (assn) rule, we know that $y \notin \mathrm{FV}(e)$ we have:

$$(\gamma[y \mapsto c])[z \mapsto [\![e]\!]\gamma] = (\gamma[z \mapsto [\![e]\!]\gamma])[y \mapsto c], \tag{40}$$

Now we can expand this expression to:

$$(\gamma[z \mapsto [\![e]\!]\gamma])[y \mapsto c] = [\![p_1]\!](\gamma[z \mapsto [\![e]\!]\gamma]) = [\![z = e; p_1]\!](\gamma).$$

This proves the first identity.

To prove the operational semantics correct, we first observe that in the derivation tree for p we apply the Comp rule in Figure 8, the appropriate derivation for p_1, and then Expr-Assign rule for $z = e$. Since the Expr-Assign only binds z, the binding for y is unchanged, and e_1 only depends on deterministic variables, by the type system, we can apply Theorem 5.5 to prove consistency. Since e does not depend on y and moreover, is deterministic by the type system, we can apply 5.1 to prove correctness for z. The proof for $z = e; p_1$ is similar.

□

Lemma C.4. *Let $p \triangleq (z = e; y = \textbf{\textit{integral}}\ (a, b)\ e_1/(x - s)^k\ \boldsymbol{dx})$ be a well-typed program such that $z \neq s$. For every $\gamma \in Ctx$ such that the denotation of p is not* err*, there exists an e_2 such that:*

$$[\![p]\!](\gamma) = [\![y = \textbf{\textit{integral}}\ (a, b)\ e_2/(x - s)^k\ \boldsymbol{dx}; z = e]\!](\gamma).$$

Further, let $\theta \triangleq [\![p]\!]\gamma(y)$ and $\theta' \triangleq \sum_{w \in Var} [\![p_1]\!](\gamma[w \mapsto t])(y)\big|_{t=\gamma(w)} \gamma'(w)$. Let the family of random variables $\{c_i\}_{i\in\mathbb{N}}$, $\{c_i'\}_{i\in\mathbb{N}}$, $\{d_i\}_{i\in\mathbb{N}}$, and $\{d_i'\}_{i\in\mathbb{N}}$ obtained by the operational semantics for each of the two above programs (specifically, $(\gamma, \gamma', p) \Downarrow_o \hat{\gamma}_i, \hat{\gamma}_i'$ and $(\gamma, \gamma', z = e; y = \textbf{\textit{integral}}\ (a, b)\ e_2/(x - s)^k\ \boldsymbol{dx}) \Downarrow_o \tilde{\gamma}_i, \tilde{\gamma}_i'$ with $c_i = \hat{\gamma}_i(y), c_i' = \hat{\gamma}_i'(y), d_i = \tilde{\gamma}_i(y)$, and $d_i' = \tilde{\gamma}_i'(y)$) be given. Then, $T_m = \frac{1}{m}\sum_{i=1}^m c_i$ and $U_m = \frac{1}{m}\sum_{i=1}^m d_i$ are strongly consistent for θ and $T_m' = \frac{1}{m}\sum_{i=1}^m c_i'$ and $U_m' = \frac{1}{m}\sum_{i=1}^m d_i'$ are strongly consistent for θ'. Also, $\gamma_1(x) = \gamma_2(x)$ and $\gamma_1'(x) = \gamma_2'(x)$ for every $x \neq y$ such that $x \in Var$. «

Proof. First, we assume that $y \notin \mathrm{FV}(e)$ because we can always rename y to a variable not in $\mathrm{FV}(p)$ both in e and in Var. Let $p_1 \triangleq y = \textbf{integral}\ (a, b)\ e_1/(x - s)^k\ \mathbf{d}x$.

If $z = x$, then by the definition of sequential composition and assignment in the denotational semantics:

$$[\![x = e; p_1]\!](\gamma) = [\![p_1]\!](\gamma[x \mapsto [\![e]\!]\gamma]) = (\gamma[x \mapsto [\![e]\!]\gamma])[y \mapsto c],$$

where c is as defined in the denotational semantics. Since x is bound in the integral, we know that $x \notin \mathrm{FV}(p_1)$ and therefore:

$$\begin{aligned}(\gamma[x \mapsto [\![e]\!]\gamma])[y \mapsto c] &= (\gamma[y \mapsto c])[x \mapsto [\![e]\!]\gamma] \\ &= [\![x = e]\!](\gamma[y \mapsto c]) \\ &= [\![p_1; x = e]\!](\gamma).\end{aligned}$$

This proves the desired identity.

If $z \neq x$ and $z \neq s$, then by the definition of sequential composition and assignment in the denotational semantics:

$$[\![z = e; p_1]\!](\gamma) = [\![p_1]\!](\gamma[x \mapsto [\![e]\!]\gamma]).$$

By the type system (Figure 22), $z \notin \mathrm{FV}(e_1)$ and thus, we can set e_2 to e_1 where each z is replaced by e, noting that the denotation of e_1 and e_2 are equal at γ. Let $p_2 \triangleq y = \textbf{integral}\ (a, b)\ e_2/(x - s)^k\ \mathbf{d}x$. We apply a similar argument as in the $z = x$ case after replacing e_1 with e_2.

$$\begin{aligned}[\![p_1]\!](\gamma[x \mapsto [\![e]\!]\gamma]) &= [\![p_2]\!](\gamma[x \mapsto [\![e]\!]\gamma]) \\ &= (\gamma[x \mapsto [\![e]\!]\gamma])[y \mapsto c] \\ &= (\gamma[y \mapsto c])[x \mapsto [\![e]\!]\gamma] \\ &= [\![x = e]\!](\gamma[y \mapsto c]) \\ &= [\![x = e; p_2]\!](\gamma),\end{aligned}$$

where c in the second and third equation is as defined in the denotational semantics.

To prove the operational semantics correct, we first observe that in the derivation tree for p we apply the Comp rule in Figure 8, the Expr-Assign rule for $z = e$, and the Integral-Assign rule for p_1. Since the Integral-Assign rule only binds y, and e_1 only depend on deterministic variables by the type system, so we can apply 5.5 to prove consistency. Since the Integral-Assign rule only binds y, the binding for z is unchanged, and since e only depends on deterministic variables, we can apply 5.1 to prove correctness for z. The proof for $z = e; p_2$ is similar. □

Theorem 5.7 (Consistency). *Let p be a well-typed program, $x \in Var$ be an arbitrary variable, and $\gamma, \gamma' \in Ctx$ be contexts with $[\![p]\!]\gamma \neq \mathrm{err}$. Separate runs of the operational semantics produce (i.i.d.) random variables $\{c_i\}_{i\in\mathbb{N}}$ and $\{c'_i\}_{i\in\mathbb{N}}$, that is, for all $i \in \mathbb{N}$, we obtain $(\gamma, \gamma', p) \Downarrow_o \hat{\gamma}_i, \hat{\gamma}'_i$ with $c_i \triangleq \hat{\gamma}_i(x)$ and $c'_i \triangleq \hat{\gamma}'_i(x)$. Then, the family of estimators $T_m \triangleq \frac{1}{m}\sum_{i=1}^m c_i$ is strongly consistent for $[\![p]\!](\gamma)(x)$, and $T'_m \triangleq \frac{1}{m}\sum_{i=1}^m c'_i$ is strongly consistent for $\sum_{z\in Var} \frac{d}{dt}[\![p]\!](\gamma[z \mapsto t])(x)\big|_{t=\gamma(z)} \cdot \gamma'(z)$.* «

Proof. We start by breaking the program into a normal form with three parts: a deterministic part, a block of integral assignments, and a part that depends on the result of integration, but does not have any integral assignments.

We statically unroll all for-loops. For now, we consider the case where the first integral does not occur within a conditional block. The first step of normalization moves the deterministic assignments that occur after the first integral assignment above it. We define a *deterministic assign statement* as an assign statement that satisfies the Deterministic (assn) rule in Figure 22. The region of the program after the first integral is free of control flow and therefore the only cases we need to consider is swapping an assign statement that is not deterministic with a deterministic assignment statement and swapping an integral assignment statement with a deterministic assign statement. We prove the correctness of this transformation in Lemma C.3.

The second step of the normalization process is to move all the integral assignment statements up into a block following the first integral assignment. The type system ensures that there is no

nesting of integrals, so the free variables of the integrand (excluding the variable of integration) are statically determined (i.e., there is no need to quantify over all possible traces). In other words, the random variables produced by the sampling in the operational semantics are independent.

Since the integrals do not depend on each other and there is now control flow in this region of the program as enforced by the type system, we can move up all the integral assignment expressions to the first integral assignment. More formally, the only cases are the sequential composition of programs of the form: $x = e$ and $y = \textbf{integral}\ (a, b)\ e/(x - s)^k\ \mathbf{d}x$. Thus, it is sufficient to prove that there is a way to swap expression assignment and integral assignment. We prove this result in Lemma C.4.

Now that we have transformed the program into a normal form, we prove that each part is correct.

The deterministic part of the program that can have discontinuities is correct by Lemma C.2, which shows that for every input context γ, the denotational semantics and operational semantics produce an equivalent output context γ_* for every deterministic statement.

Next, Corollary 5.6 applies to each integral assignment, giving a proof that the family of estimators $(T_m)_{m\in\mathbb{N}}$ is a consistent estimator for the function denoted by this line. We can collect all these random variables into a new random variable $X \triangleq (\hat{X}_1, \ldots, \hat{X}_k)$, where k is the number of integrals in the program. We can collect the estimator for each of these random variables into a product $X_n \triangleq (T_{(1,n)}, \ldots, T_{(k,n)})$ for each $y_1, \ldots, y_k$ that is assigned to an integral. Since the limit of a product is equal to the product of limits, we have that X_n is a strongly consistent for X.

The remaining part of the program corresponds to a deterministic, total, smooth function g from the random variables in $\mathbb{R}^k$ to $\mathbb{R}^{|Var|}$ that takes the random variable X. Totality comes from the fact that the denotation is not err at γ, satisfying the precondition of Theorem 4.2 and the fact that there are no conditional statements in this part of the program. Now we can apply Proposition C.1 to prove that $(g(X_n))_{n\in\mathbb{N}}$ is consistent for $g(X)$, and Lemma C.2 shows the equivalence between the denotational and operational semantics for the deterministic statements that define g.

Since the lemmas also apply to the lemmas, we can apply the same reasoning to the derivative of the program.

□

D Additional Results for Section 6

We provide a few additional results for the experiments in Appendix 6 for the sake of completeness.

D.1 Timing Benchmark Results for the Finite Hilbert Transform

Table 4 shows the median time as well as the standard deviation (in milliseconds) over 25 runs of the time for the Standard, Ours, Deriv. Standard, and Deriv. Ours methods on the finite Hilbert transform. As in the the rest of the evaluation, we use 10,000 samples to estimate the singular integrals. We drop the first iteration to avoid timing the JIT compilation of JAX, where the first iteration took about 1 order of magnitude longer than subsequent ones for Standard and Deriv. Standard. Without JIT compilation (i.e., using `jax.jit`), our method is slower than the baseline (Standard). However, with JIT compilation, our method is roughly as fast as the baseline (Standard (JIT)).

Table 4. Comparison of median times (in ms ± standard deviation) for Standard, Ours, Deriv. Standard, and Deriv Ours on the finite Hilbert transform, without and with running `jax.jit`.

$f(u)$	**Standard**	**Ours**	**Deriv. Standard**	**Deriv. Ours**
u	0.91 ± 0.09	3.01 ± 0.42	3.49 ± 0.14	19.67 ± 8.72
u^2	1.15 ± 0.04	4.04 ± 0.15	3.74 ± 0.22	25.11 ± 2.39
e^u	1.10 ± 0.10	3.40 ± 0.21	3.39 ± 0.18	22.55 ± 1.78
ue^u	1.21 ± 0.07	3.82 ± 0.12	3.76 ± 0.22	37.44 ± 10.69
$\sin(u)$	1.10 ± 0.06	3.79 ± 0.19	3.37 ± 0.19	32.68 ± 6.60
$\cos(u)$	1.09 ± 0.06	3.42 ± 0.22	4.10 ± 0.78	33.93 ± 1.66

$f(u)$	**Standard (JIT)**	**Ours (JIT)**	**Deriv. Standard (JIT)**	**Deriv. Ours (JIT)**
u	0.21 ± 0.06	0.22 ± 0.04	0.20 ± 0.13	0.21 ± 0.04
u^2	0.21 ± 0.12	0.21 ± 0.08	0.23 ± 0.05	0.18 ± 0.09
e^u	0.22 ± 0.13	0.28 ± 0.49	0.21 ± 0.04	0.24 ± 0.09
ue^u	0.22 ± 0.04	0.25 ± 0.09	0.20 ± 0.12	0.23 ± 0.02
$\sin(u)$	0.22 ± 0.13	0.25 ± 0.11	0.24 ± 0.10	0.27 ± 0.03
$\cos(u)$	0.21 ± 0.13	0.29 ± 0.07	0.23 ± 0.04	0.20 ± 0.08

D.2 Training Curves

We provide the training curves for each of the experiments in Appendix 6. Each plot shows the training loss over iterations where the integral estimation uses 50 samples. Fig. 12 shows the training loss for the thin airfoil problem with the NACA 6412 airfoil. Fig. 13 shows the training loss for the symmetrical airfoil problem with the NACA 0012 airfoil. Fig. 14 shows the training loss for the crack problem in Harold Page Starr [35]. Finally, Fig. 15 shows the training loss for the crack problem in Kaya and Erdogan [38].

D.3 Analysis of the Small Anomaly in the Crack Problem Simulation

There is a small anomaly near 0 in Figure 10b for the crack problem in Harold Page Starr [35]. To confirm that this is an edge effect that results from running the simulation with relatively few samples we (1) provide the five plots that make up the aggregated results in Figures 16-20 and

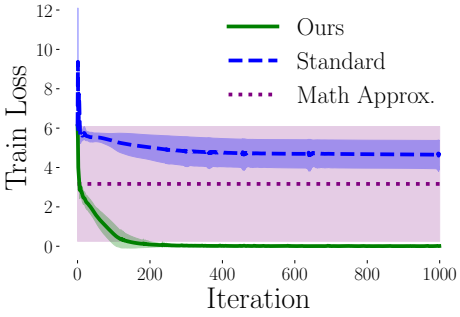


Fig. 12. Train loss over iterations for the asymmetrical airfoils.

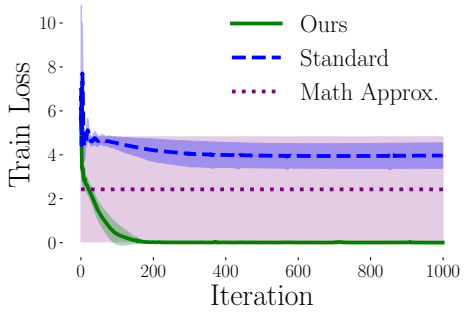


Fig. 13. Train loss over iterations for the symmetrical airfoil.

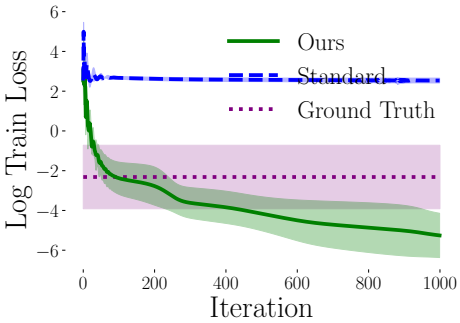


Fig. 14. The loss function for the crack function in Harold Page Starr [35].

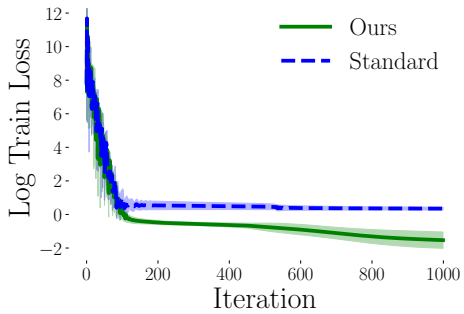


Fig. 15. The log loss for the crack function in Kaya and Erdogan [38].

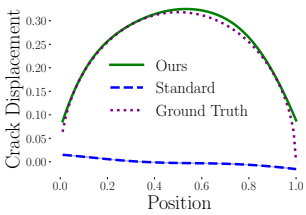


Fig. 16. A plot from experiment Figure 10b, where the random seed is 0.

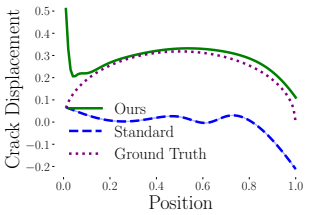


Fig. 17. A plot from experiment Figure 10b, where the random seed is 1.

(2) run the simulation again with 500 samples for the singular integral instead of 50 samples in Figure 21.

Figures 16-20 show that the anomaly comes primarily from a single run (Figure 17) of the simulation and not prominent in the other runs. In Figure 21, we see that the anomaly near 0 is no longer present when we use 500 samples for the singular integral.

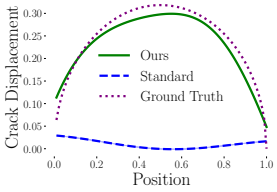


Fig. 18. A plot from experiment Figure 10b, where the random seed is 2.

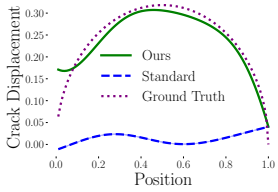


Fig. 19. A plot from experiment Figure 10b, where the random seed is 3.

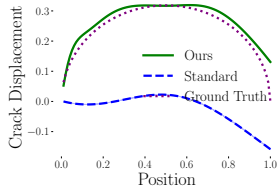
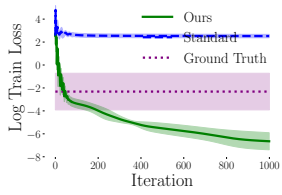
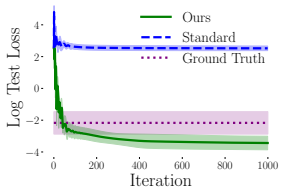


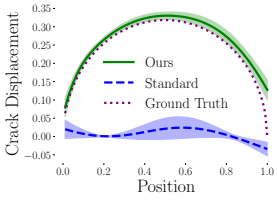
Fig. 20. A plot from experiment Figure 10b, where the random seed is 4.



(a) Train loss over iterations.



(b) Test loss over iterations.



(c) Learned solution.

Fig. 21. A numerical solution to the crack problem in Harold Page Starr [35] using a 500 sample estimate for the singular integral.

$$\textsc{Res-of-Integral (Assn)}\ \frac{\mathrm{FV}(e) \cap S \neq \emptyset}{(S, x = e) \to S \cup \{x\}} \qquad \textsc{Deterministic (Assn)}\ \frac{\mathrm{FV}(e) \cap S = \emptyset}{(S, x = e) \to S}$$

$$\textsc{No-Nesting}\ \frac{\big((\{s\} \cup \mathrm{FV}(e)) \setminus \{x\}\big) \cap S \neq \emptyset}{(S, y = \textbf{integral}\ (a, b)\ e/(x - s)^k\ \mathbf{d}x) \to \mathrm{err}} \qquad \textsc{Res-of-Integral (Base)}\ \frac{\big((\{s\} \cup \mathrm{FV}(e)) \setminus \{x\}\big) \cap S = \emptyset}{(S, y = \textbf{integral}\ (a, b)\ e/(x - s)^k\ \mathbf{d}x) \to S \cup \{y\}}$$

$$\frac{(S, p_1) \to \mathrm{err}}{(S, p_1; p_2) \to \mathrm{err}} \qquad \frac{(S, p_1) \to S' \qquad (S', p_2) \to \mathrm{err}}{(S, p_1; p_2) \to \mathrm{err}} \qquad \frac{(S, p_1) \to S' \qquad (S', p_2) \to S''}{(S, p_1; p_2) \to S''}$$

$$\textsc{No-If-After-Sample}\ \frac{S \neq \emptyset}{(S, \textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2) \to \mathrm{err}} \qquad \frac{S = \emptyset \qquad (S, p_1) \to \mathrm{err}}{(S, \textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2) \to \mathrm{err}}$$

$$\frac{S = \emptyset \qquad (S, p_2) \to \mathrm{err}}{(S, \textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2) \to \mathrm{err}} \qquad \frac{S = \emptyset \qquad (S, p_1) \to S' \qquad (S, p_2) \to S''}{(S, \textbf{ifpos}\ e\ \textbf{then}\ p_1\ \textbf{else}\ p_2) \to S' \cup S''}$$

$$\frac{Q_0 \triangleq S \qquad k = \lfloor (b - a)/c \rfloor \qquad \exists i \in \{0, \ldots, k - 1\}.\ (Q_i, p) \to Q_{i+1} \qquad Q_{i+1} = \mathrm{err}}{(S, \textbf{for}\ x\ \textbf{in range}(a, b, c) : p) \to \mathrm{err}}$$

$$\frac{S_0 \triangleq S \qquad k = \lfloor (b - a)/c \rfloor \qquad \forall i \in \{0, \ldots, k - 1\}.\ (S_i, p) \to S_{i+1}}{(S, \textbf{for}\ x\ \textbf{in range}(a, b, c) : p) \to S_k}$$

Fig. 22. The helper typing rule is defined as $(S, p) \to Q$, where S is a set of variables and Q is either a set of variables or err. In the above, whenever we write S with subscripts or super scripts, it represents a set. It checks that no if-statements occur after an integral and that integrals are not nested.

E Type System

Figure 22 depicts a type system that returns a (conservative) set of variables of integration that represent random variables in the operational semantics. It is a conservative analysis due to conditionals because the analysis may add a variable to the set of random variables when it is only a random variable in a branch that is never executed. It checks that no if-statements occur before an integral and that integrals are not nested. The helper typing rule is defined as $(S, p) \to Q$, where S is a set of variables and Q is either a set of variables or err.

We present the following definition of a well-typed program.

Definition E.1. A program p is *well-typed* if $Q \neq \mathrm{err}$, where $(\{\}, p) \to Q$. △

F Boundaries of the Available Theory

In this section, we discuss the limitations of the theory of singular integrals, which also provide a practical set of limitations of SingularFlow. In the following, we adopt the convention that $f(x)$ is a smooth function and $a, b \in \mathbb{R}$ with $a < b$. Much of the inspiration for examples comes from Kutt [43, Pages 47-48].

Hadamard Finite Part Integral of a Nonnegative Integrand can be Negative. The Riemann integral of a nonnegative function is nonnegative, which accords with the view of integration as the area under a curve. The Hadamard finite part integral does not satisfy this property as shown below.

Example F.1. Recall from Proposition 3.10 that $\mathcal{H}\int_a^b \frac{f(u)}{(u-s)^2}du = C\int_a^b \frac{f^{(1)}(u)}{u-s}du - \frac{f(u)}{(u-s)}\Big|_{u=a}^{b}$. So, $\mathcal{H}\int_{-1}^{1} \frac{1}{x^2}dx$, with $(a,b) = (-1,1)$ and $f(x) = 1$, is equal to $C\int_{-1}^{1} \frac{0}{x}dx - f(1) - f(-1) = 0 - 1 - 1 = -2$. In conclusion, the integrand $1/x^2$ is nonnegative, but the Hadamard finite part integral is negative. △

Singular Integrals are not Monotone Operators. The Riemann integral is *monotone* because if $f(x) \le g(x)$ for almost all $x \in \mathbb{R}$, then $\int_a^b f(x)dx \le \int_a^b g(x)dx$. Informally, the Riemann integral is monotone because the area under the curve of f is smaller than the area under the curve of g. In contrast, the Hadamard finite part integral is not monotone as we show in the following example.

Example F.2. The inequality $\frac{1}{x^3} < \frac{1+x}{x^3}$ for all real $x \neq 0$ holds because if $x \neq 0$ then $0 < \frac{1}{x^2}$. However, the Hadamard finite part integral of these functions is in the reverse order $\mathcal{H}\int_{-1}^{1} \frac{x+1}{x^3}dx < \mathcal{H}\int_{-1}^{1} \frac{1}{x^3}dx$. This is a consequence of $\mathcal{H}\int_{-1}^{1} \frac{x+1}{x^3}dx = \mathcal{H}\int_{-1}^{1} \frac{1}{x^2}dx + \mathcal{H}\int_{-1}^{1} \frac{1}{x^3}dx = -2 + \mathcal{H}\int_{-1}^{1} \frac{1}{x^3}dx$. △

Inequalities Depending on Convexity do not Necessarily Hold. Likewise, inequalities that are true of standard integrals do not necessarily hold for singular integrals.

Example F.3. The function $x \mapsto |x|$ on $\mathbb{R}$ is convex. Jensen's inequality states that

$$\left|\int_a^b \frac{f(x)}{b-a}dx\right| \le \int_a^b \frac{|f(x)|}{b-a}dx, \qquad \text{but} \qquad \left|\mathcal{H}\int_{-1}^{1} \frac{1}{x^2}\frac{1}{2}dx\right| = 1 > -1 = \mathcal{H}\int_{-1}^{1} \left|\frac{1}{x^2}\right|\frac{1}{2}dx.$$

Similarly, for the concave function ln, Jensen's inequality says that

$$\ln\left(\int_a^b \frac{f(x)}{b-a}dx\right) \ge \int_a^b \frac{\ln f(x)}{b-a}dx.$$

However, the opposite is true for the standard normal distribution (using Example F.2):

$$\ln\left(\int_{-1}^{1} \frac{e^{-\frac{1}{2x^2}}}{2}dx\right) \approx -1.56 < 0.5 = -\frac{1}{4}\cdot\mathcal{H}\int_{-1}^{1} \frac{1}{x^2}dx = \mathcal{H}\int_{-1}^{1} -\frac{1}{2x^2}\frac{1}{2}dx = \mathcal{H}\int_{-1}^{1} \frac{\ln\left(e^{-\frac{1}{2x^2}}\right)}{2}dx.$$ △

The failure of Jensen's inequality to hold has practical consequences. Without Jensen's inequality, we cannot prove that the ELBO is a lower bound on the log likelihood and as such, we cannot justify variational inference when the integral is specified in terms of singular integration [9].

G Related Work in Math, Numerics, and Computer Graphics

Mathematical Foundations of Singular Integrals. Cauchy [16] and Hadamard [34] introduce the Cauchy principal value and Hadamard finite part integrals, respectively. Gelfand and Shilov [29] studied singular integrals in the context of distribution theory and provided a formal justification of the Hadamard finite part integral. Their setup was different from ours in that they studied differentiating the integral operator (i.e., distribution), while we study differentiating with respect to a parameter in the integral. Calderon and Zygmund [12] study singular integrals depending on a parameter such as the Hilbert transform but do not study their derivatives as Hadamard finite part integrals. Stein [79] wrote a classic book on singular integrals, and Abels [2] provides a more modern treatment with many applications. King [40] is a comprehensive book covering the Cauchy principal value integral and its applications in the context of the Hilbert transform. Estrada and Kanwal [23], Ladopoulos [44] are texts on singular integral equations and their applications, such as in aerodynamics and mechanical engineering (studied in this paper), as well as many other domains.

Numerical Techniques for Singular Integrals. Researchers in the numerical analysis community developed techniques for evaluating singular integrals and for solving singular integral equations. We provide a brief overview but note that the literature is vast. Longman [53] develop a symmetric sampling technique for Cauchy principal value integrals. Kutt [43] develops numerical methods accounting for singularities, including those at an endpoint of the interval of integration, which is interesting future work. Monegato [63] provides an excellent overview.

Piessens et al. [71] is a FORTRAN library that provides a variety of quadrature rules, including for singular integrals. FeynCalc [59] is a Mathematica package that supports regularization integrals but does not apply to the benchmarks in this paper.[8] Slevinsky and Olver [78] develop a spectral method (i.e., using the Fourier transforms) that uses Chebyshev polynomials to efficiently evaluate singular integral equations. They build a Julia package that they apply to problems in fracture mechanics, electricity and magnetism, and solving the Helmholtz equation.

Singular Integrals in Computer Graphics. In computer graphics, researchers often want to create physically accurate simulations of physical phenomena. One approach to creating such simulations is to solve partial differential equations (PDEs) that model the physical phenomena. Some PDEs can be converted into an integral equation using *Green's functions* and solved using numerical methods. Nabizadeh et al. [65] solve problems on infinite domains using the *Kelvin transform*, which applies a change of coordinates $x \mapsto y = x/|x|^2$ to map an infinite domain around an object to a finite domain in the interior of the object. The problem is solved on a finite domain with a singularity that is accounted for in a hand-coded way.

Recent work develops a Monte Carlo solver for integral equations that caches function evaluations for efficiency [62]. Caching is not compatible with importance sampling (preventing handling singularities by importance sampling with a function proportional to the integrand around the singularity) and instead they use heuristics (e.g., clamping) to mitigate singularities [62, Section 3.3]. Other recent work presents a technique for rendering functions with singularities efficiently [31].

[8] As confirmed in correspondence with the developer.