Supplementary Material for "Stellar-AXIS"

SMath Hacks 2025

Algorithmic Speed Gains in Radiative Transfer and Orbital Mechanical Computations — Using Neural Network Approximations of Antiderivatives

1 Basic Network Theory

This section is written to establish a basic understanding of neural network theory. Feel free to skip (to section 2) if you have experience with them!

We begin by considering a feed-forward neural network, specifically a multilayer perceptron (MLP). Let:

- $\mathbf{x} \in \mathbb{R}^{n_0}$ be the input vector, where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_0} \end{bmatrix}. \tag{1}$$

- The network have L layers (not counting the input as a layer).
- Layer l produce an activation vector $\mathbf{h}^{(l)} \in \mathbb{R}^{n_l}$, where

$$\mathbf{h}^{(l)} = \begin{bmatrix} h_1^{(l)} \\ h_2^{(l)} \\ \vdots \\ h_{n_l}^{(l)} \end{bmatrix}. \tag{2}$$

- $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ be the weight matrix and bias vector for layer l, where

$$\mathbf{W}^{(l)} = \begin{bmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \cdots & w_{1n_{l-1}}^{(l)} \\ w_{21}^{(l)} & w_{22}^{(l)} & \cdots & w_{2n_{l-1}}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_l1}^{(l)} & w_{n_l2}^{(l)} & \cdots & w_{n_ln_{l-1}}^{(l)} \end{bmatrix}, \quad \mathbf{b}^{(l)} = \begin{bmatrix} b_1^{(l)} \\ b_2^{(l)} \\ \vdots \\ b_{n_l}^{(l)} \end{bmatrix}. \tag{3}$$

We assume that each layer l uses an activation function $\sigma^{(l)} : \mathbb{R} \to \mathbb{R}$ (which can vary by layer) applied element-wise. For example, $\sigma^{(l)}$ could be a ReLU, sigmoid, or other function, possibly subject to differentiability conditions discussed below.

1.1 Forward Propagation

The forward pass for each layer $l = 1, \ldots, L$ proceeds as:

$$\begin{aligned}\mathbf{z}^{(l)} &= \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, \quad \mathbf{z}^{(l)} \in \mathbb{R}^{n_l} \\ \mathbf{h}^{(l)} &= \sigma^{(l)}\big(\mathbf{z}^{(l)}\big), \quad \mathbf{h}^{(l)} \in \mathbb{R}^{n_l}\end{aligned} \tag{4}$$

where $\mathbf{h}^{(0)} \equiv \mathbf{x}$. For concreteness, if each layer uses the same activation σ, the overall computation can be compactly written as

$$\mathbf{h}^{(L)} = \sigma\big(\mathbf{W}^{(L)}\,\sigma\big(\mathbf{W}^{(L-1)}\cdots\sigma(\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)})+\mathbf{b}^{(2)}\big)\cdots+\mathbf{b}^{(L)}\big). \tag{5}$$

Here, $\mathbf{h}^{(L)}$ is the final output.

1.2 Backpropagation

Backpropagation computes the gradient of a loss function $\mathcal{L}$ with respect to all parameters $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$. We illustrate using a mean squared error (MSE) loss:

$$\mathcal{L} = \frac{1}{2}\|\mathbf{y} - \mathbf{h}^{(L)}\|^2, \tag{6}$$

where $\mathbf{y}$ is the target vector

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_L} \end{bmatrix}. \tag{7}$$

Output Layer Gradient. Define the error signal at layer l by $\boldsymbol{\delta}^{(l)}$. At the output layer L, we have:

$$\nabla_{\mathbf{h}^{(L)}}\mathcal{L} = \mathbf{h}^{(L)} - \mathbf{y}, \tag{8}$$

and thus

$$\boldsymbol{\delta}^{(L)} = \big(\mathbf{h}^{(L)} - \mathbf{y}\big) \odot \sigma'^{(L)}\big(\mathbf{z}^{(L)}\big), \tag{9}$$

where $\odot$ denotes element-wise multiplication and $\sigma'^{(L)}$ is the derivative of $\sigma^{(L)}$ evaluated element-wise.

Hidden Layer Gradients. For a hidden layer l, the error signal is

$$\boldsymbol{\delta}^{(l)} = \big(\mathbf{W}^{(l+1)}\big)^\top \boldsymbol{\delta}^{(l+1)} \;\odot\; \sigma'^{(l)}\big(\mathbf{z}^{(l)}\big). \tag{10}$$

Weight and Bias Gradients. Given the layer's error signal, the partial derivatives of the loss are

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)}\,\big(\mathbf{h}^{(l-1)}\big)^\top, \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\delta}^{(l)}, \tag{12}$$

assuming standard vector/matrix dimensional consistency.

Weight and Bias Updates. Using the gradients computed above, the weights and biases are updated through gradient descent or its variants. For a learning rate $\eta > 0$, the update rules are:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}}, \tag{13}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}. \tag{14}$$

This iterative adjustment minimizes the loss function $\mathcal{L}$ over successive training epochs.

Summary of Backpropagation. The essence of backpropagation is captured in the recursive computation of error signals and gradients:

$$\boldsymbol{\delta}^{(l)} = \big(\mathbf{W}^{(l+1)}\big)^\top \boldsymbol{\delta}^{(l+1)} \odot \sigma'^{(l)}\big(\mathbf{z}^{(l)}\big), \quad \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} \big(\mathbf{h}^{(l-1)}\big)^\top. \tag{15}$$

2 Modified Loss Functions for Antiderivative Training

We now describe a procedure to train a neural network $F_\theta(x)$ as an approximate antiderivative of a known function f. This approach requires that F_θ be differentiable (in x) almost everywhere, so we typically assume smooth activation functions or carefully handle points of non-smoothness.

2.1 1D Integration

Let $f(x)$ be a given function on $[a, b] \subset \mathbb{R}$. We want

$$F'(x) \approx f(x),$$

where $F(x)$ is a neural network model $F_\theta(x)$. Since an antiderivative is unique up to a constant, we fix F_θ at least at one "anchor" point a:

$$F_\theta(a) \approx C_a,$$

where C_a is a known (or desired) reference value. To enforce these conditions, define the loss:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(\frac{d}{dx} F_\theta(x_i) - f(x_i) \Big)^2 + \lambda \big(F_\theta(a) - C_a \big)^2, \tag{16}$$

where $\{x_i\}_{i=1}^N \subset [a, b]$ are training (collocation) points, λ is a hyperparameter to balance the derivative matching and anchor constraints, and

$$\frac{d}{dx} F_\theta(x_i)$$

is computed by backpropagation through the network architecture with respect to the input x. In practice, if σ is smooth, this is straightforward. If σ is piecewise differentiable (e.g. ReLU), one uses subdifferential definitions or avoids sampling exactly at non-differentiable points.

- **Anchor Points:** One may choose multiple anchors, say $F_\theta(a_k) \approx C_{a_k}$, each added to the loss with a suitable coefficient.
- **Existence and Uniqueness:** In one dimension, for a continuous f, there is a unique (up to a constant) function F with $F'(x) = f(x)$. Here we anchor the constant by imposing $F_\theta(a) = C_a$.

3 Extension to Higher-Dimensional Cases

For $f(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^d$, constructing an exact "antiderivative" requires caution. Suppose we wish to find $F_\theta(\mathbf{x})$ such that

$$\nabla F_\theta(\mathbf{x}) \approx f(\mathbf{x}),$$

where now f must be a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^d$, or we want some mixed partial derivative to match a scalar $f(\mathbf{x})$. The correct formulation depends on whether f is a gradient field:

$$f(\mathbf{x}) = \nabla G(\mathbf{x}) \quad \text{for some scalar field } G.$$

If f is not a conservative field, no single scalar F can satisfy $\nabla F = f$ globally. Below are two scenarios:

3.1 Direct Approach via Mixed Derivatives

If we want F_θ to satisfy

$$\frac{\partial^k F_\theta}{\partial x_1 \partial x_2 \cdots \partial x_k}(\mathbf{x}) \approx f(\mathbf{x}),$$

then we must check integrability conditions (e.g. Schwarz's theorem/Clairaut's theorem on equality of mixed partials) and have suitable boundary or anchor conditions. A general form of the loss:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\partial^k F_\theta}{\partial x_1 \cdots \partial x_k}(\mathbf{x}_i) - f(\mathbf{x}_i) \right)^2 + \sum_j \lambda_j \Big(F_\theta(\mathbf{x}_j) - C_j \Big)^2.$$

Here, each anchor $\mathbf{x}_j$ ensures a boundary condition or constant offset. This approach is only valid if the PDE/integrability constraints hold for f.

3.2 Hierarchical Integration (Separable or Ordered Integration)

Consider approximating

$$\int \cdots \int f(\mathbf{x})\, dx_1\, dx_2 \cdots dx_d$$

by splitting dimension-by-dimension. For instance:

1. Train $F_\theta^{(1)}(\mathbf{x})$ such that $\frac{\partial}{\partial x_1} F_\theta^{(1)}(\mathbf{x}) \approx f(\mathbf{x})$.
2. Next, train $F_\theta^{(2)}(\mathbf{x})$ so that $\frac{\partial}{\partial x_2} F_\theta^{(2)}(\mathbf{x}) \approx F_\theta^{(1)}(\mathbf{x})$.
3. Continue for higher dimensions in an analogous way.

Though conceptually simpler, one should note:

- **Order of Integration:** In genuinely multivariate integrals, the path/ordering of partial integrations can affect the intermediate functions.
- **Error Accumulation:** Each stage is approximate, so errors may propagate.
- **Non-Conservative Fields:** If f is not conservative, the final integrated function could be inconsistent across different paths.

4 Adaptive Sampling

We now address how to choose training (collocation) points in a more efficient manner than uniform sampling. The method subdivides the domain into subregions, estimates variance, and allocates samples accordingly.

4.1 Mathematical Framework

Step 1: Partition the Domain For a domain $\mathcal{D} \subseteq \mathbb{R}^d$, partition it into k subregions $\{\mathcal{D}_i\}_{i=1}^k$. In the simplest case, divide each dimension into m equal parts, giving $k = m^d$ hyperrectangles.

Step 2: Estimate Variance in Each Subregion For each subregion $\mathcal{D}_i$, sample a small number n_i of points $\{x_j\} \subset \mathcal{D}_i$. Define

$$\overline{f_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} f(x_j), \quad \mathrm{Var}(f;\, \mathcal{D}_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} \big(f(x_j) - \overline{f_i}\big)^2.$$

(Optionally, one may use $1/(n_i - 1)$ for an unbiased variance estimate.)

Step 3: Allocate Sampling Points Given a total sampling budget N, assign

$$N_i = \max\Big\{1,\ \left\lfloor \frac{\mathrm{Var}(f;\, \mathcal{D}_i)}{\sum_{r=1}^k \mathrm{Var}(f;\, \mathcal{D}_r)}\, N \right\rfloor \Big\}.$$

A small positive lower bound (e.g. at least 1 sample) is often set to prevent ignoring subregions entirely.

Step 4: Generate Points Generate N_i random points (or quasi-random, e.g. low-discrepancy) within each $\mathcal{D}_i$. These constitute the final set of training points for the neural network.

Step 5: Integration Region Identification (During Inference) To identify which subregion a test point $\mathbf{x} \in \mathcal{D}$ belongs to, check the coordinate bounds of each subregion. For example, in 2D, if

$$x_1 \in [a_1, b_1], \quad x_2 \in [a_2, b_2],$$

and these intervals are each split into m subintervals, we find the index of subinterval containing x_1 and that containing x_2. This pair of indices yields the hyperrectangle $\mathcal{D}_i$.

5 Theorem 1: The Universal Approximation Theorem

The Universal Approximation Theorem establishes that neural networks with at least one hidden layer and a non-linear activation function can approximate any continuous function on a compact subset of $\mathbb{R}^n$ to any desired degree of accuracy, given sufficient neurons.

Statement

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function defined on a compact subset $K \subseteq \mathbb{R}^n$, and let $\epsilon > 0$. Then, there exists a neural network F_θ with a single hidden layer, a finite number of neurons, and a non-linear activation function σ, such that:

$$\sup_{x \in K} |f(x) - F_\theta(x)| < \epsilon.$$

Proof

The proof is based on Cybenko's 1989 paper and involves the following steps:

Step 1: Partition of Unity. Since K is compact, $f(x)$ is uniformly continuous on K. This allows us to cover K with a finite number of overlapping compact regions $\{U_i\}_{i=1}^m$, each associated with a continuous bump function $\phi_i(x)$ such that:

$$\sum_{i=1}^{m} \phi_i(x) = 1 \quad \forall x \in K, \quad \phi_i(x) \geq 0 \quad \forall x.$$

Each $\phi_i(x)$ is a localized function that effectively "partitions" K into smaller regions.

Step 2: Local Approximation by Neurons. For any localized region U_i corresponding to $\phi_i(x)$, define a localized function $f_i(x) = f(x)\phi_i(x)$. Each $f_i(x)$ can be closely approximated by a single neuron with sufficient weights and biases:

$$f_i(x) \approx \sigma(\mathbf{w}_i^T x + b_i),$$

where σ is the activation function. The sigmoidal activation function σ is particularly useful because it is continuous and non-linear, and it can approximate step-like transitions required for the bump functions.

Step 3: Superposition of Approximations. The universal approximation capability of the neural network comes from the linearity of the output layer, allowing the network to combine the approximations of $f_i(x)$ across all i. Specifically, the neural network approximates $f(x)$ as:

$$F_\theta(x) = \sum_{i=1}^{m} c_i \sigma(\mathbf{w}_i^T x + b_i),$$

where c_i are trainable weights that scale the contribution of each neuron. The error between $F_\theta(x)$ and $f(x)$ can be made arbitrarily small by increasing m and tuning $\mathbf{w}_i$, b_i, and c_i.

Step 4: Error Control. The error between $f(x)$ and $F_\theta(x)$ is given by:

$$|f(x) - F_\theta(x)| \leq \sum_{i=1}^{m} |f_i(x) - \sigma(\mathbf{w}_i^T x + b_i)|.$$

Uniform continuity of $f(x)$ and compactness of K ensure that the error in approximating each $f_i(x)$ can be controlled. By increasing the number of neurons m and refining the network parameters, the overall error can be made less than ϵ.

Conclusion. By the above construction, we have shown that a neural network with a single hidden layer and non-linear activation function σ can approximate any continuous function $f(x)$ on a compact domain K to arbitrary precision. Thus, the theorem is proven.

References

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4), 303–314.

6 Theorem 2: An Error Bound for Neural Network Antiderivatives

Assumptions

We assume the following conditions:

1. **Continuity and Absolute Continuity:** Let $f : [a, b] \to \mathbb{R}$ be a continuous function. Let $F : [a, b] \to \mathbb{R}$ be an *absolutely continuous* function such that
$$F'(x) = f(x)$$
almost everywhere (a.e.) on $[a, b]$, and with initial condition $F(a) = C_a$.

2. **Neural Network Representation:** Let $F_\theta : [a, b] \to \mathbb{R}$ be a neural network approximation to F that is also absolutely continuous on $[a, b]$. In particular, suppose that F_θ has a derivative $F'_\theta(x)$ defined almost everywhere (which is typical for networks with piecewise continuous activations, such as ReLU), and that it satisfies
$$F_\theta(x) = F_\theta(a) + \int_a^x F'_\theta(t)\, dt \quad \text{for all } x \in [a, b].$$

3. **Integrability:** We assume that the error function $|f(t) - F'_\theta(t)|$ is Lebesgue integrable on $[a, b]$.

Theorem Statement

Under the above assumptions, the error between the true antiderivative F and its neural network approximation F_θ satisfies the bound
$$\sup_{x\in[a,b]} \big|F(x) - F_\theta(x)\big| \le |F(a) - F_\theta(a)| + \int_a^b \big|f(t) - F'_\theta(t)\big|\, dt.$$

Proof

Proof. Since F is absolutely continuous with $F'(x) = f(x)$ almost everywhere, by the Fundamental Theorem of Calculus for absolutely continuous functions we have
$$F(x) = F(a) + \int_a^x f(t)\, dt, \quad \text{for all } x \in [a, b].$$

Similarly, since F_θ is absolutely continuous and satisfies the corresponding initial condition, it holds that
$$F_\theta(x) = F_\theta(a) + \int_a^x F'_\theta(t)\, dt, \quad \text{for all } x \in [a, b].$$

Subtracting the second expression from the first gives
$$F(x) - F_\theta(x) = \Big(F(a) - F_\theta(a)\Big) + \int_a^x [f(t) - F'_\theta(t)]\, dt.$$

Taking absolute values and applying the triangle inequality yields
$$\big|F(x) - F_\theta(x)\big| \le \big|F(a) - F_\theta(a)\big| + \left|\int_a^x [f(t) - F'_\theta(t)]\, dt\right|.$$

Using the triangle inequality for integrals, we have
$$\left|\int_a^x [f(t) - F'_\theta(t)]\, dt\right| \le \int_a^x \big|f(t) - F'_\theta(t)\big| dt.$$

Thus, we obtain

$$\big|F(x) - F_\theta(x)\big| \le \big|F(a) - F_\theta(a)\big| + \int_a^x \big|f(t) - F'_\theta(t)\big| dt.$$

Since the above inequality holds for every $x \in [a, b]$, taking the supremum over $x \in [a, b]$ yields:

$$\sup_{x\in[a,b]} \big|F(x) - F_\theta(x)\big| \le \big|F(a) - F_\theta(a)\big| + \int_a^b \big|f(t) - F'_\theta(t)\big| dt.$$

This completes the proof. □

Remarks

- The absolute continuity assumption ensures that both F and F_θ can be recovered as the integral of their derivatives. For neural networks with non-smooth activations (e.g., ReLU), note that differentiability holds almost everywhere, and the integral representations are understood in the Lebesgue sense.
- In practical applications, the boundary term $|F(a) - F_\theta(a)|$ can be eliminated (or minimized) by enforcing the condition $F_\theta(a) = F(a)$ during training.
- While our theorem establishes that the overall antiderivative error is upper bounded by the sum of the initial condition error and the L^1 difference between F'_θ and f, further analysis is generally required to relate this derivative error to sampling density or network complexity.

7 Theorem 3: Effect of Increased Sampling Density on Neural Network Approximation Error

Theorem Statement

Let $f : \mathcal{X} \to \mathbb{R}$ be a continuous function on a compact domain $\mathcal{X}$. Suppose a neural network $\mathcal{N}_\theta$ (parameterized by θ) is trained on N sampled points $\{(x_i, f(x_i))\}_{i=1}^N$, drawn i.i.d. from some distribution P over $\mathcal{X}$. Denote by $\widehat{F}_N$ the learned network function and by $\mathcal{E}(\widehat{F}_N)$ an appropriate error measure (e.g., mean squared error or L^1 norm of the difference from f on $\mathcal{X}$).

Then, under standard assumptions of finite capacity (e.g., a neural network class with finite VC dimension or bounded Rademacher complexity) and well-posedness of the training process, the expected generalization error of $\widehat{F}_N$ decreases as the number of samples N (i.e., sampling density) increases. Formally, there exists a function $\varepsilon : \mathbb{N} \to \mathbb{R}_{\ge 0}$ such that

$$\mathbb{E}_{\{x_i\}\sim P}\big[\mathcal{E}(\widehat{F}_N)\big] \;\le\; \inf_{F\in\mathcal{H}} \mathcal{E}(F) \;+\; \varepsilon(N),$$

where $\mathcal{H}$ is the hypothesis class (the set of all neural networks of a given architecture), and $\varepsilon(N)$ is a monotonically decreasing function of N. Thus, as $N \to \infty$, $\varepsilon(N) \to 0$, leading to lower approximation error in expectation.

Proof

Step 1: Setup and Notation. Consider a hypothesis class $\mathcal{H}$ of neural networks (e.g., feedforward networks with fixed architecture) capable of mapping from $\mathcal{X} \subseteq \mathbb{R}^d$ to $\mathbb{R}$. We assume:

- $\{x_i\}_{i=1}^N$ are sampled i.i.d. from a probability distribution P on $\mathcal{X}$.
- The network $\widehat{F}_N$ is obtained by empirical risk minimization (ERM) on these samples, or a standard gradient-based procedure that seeks to minimize an empirical loss $\ell(\widehat{F}_N, f)$.

- The complexity of $\mathcal{H}$ can be measured by some capacity term (e.g., VC dimension, pseudo-dimension, or Rademacher complexity). Denote this capacity by $\mathrm{Comp}(\mathcal{H})$.

Step 2: Decomposition of Error. We focus on a general measure of error $\mathcal{E}$. A common example is the mean squared error on $\mathcal{X}$:

$$\mathcal{E}(F) \;=\; \int_{\mathcal{X}} \big(F(x) - f(x)\big)^2 \, dP(x),$$

but the argument extends to other losses or norms.

Denote the empirical risk of a function F on the sample $\{x_i\}$ by

$$\widehat{\mathcal{E}}_N(F) \;=\; \frac{1}{N} \sum_{i=1}^{N} \big(F(x_i) - f(x_i)\big)^2.$$

The learned network $\widehat{F}_N$ is chosen to minimize (or closely approximate the minimization of) $\widehat{\mathcal{E}}_N$ over $\mathcal{H}$.

Step 3: Uniform Convergence or Rademacher Bounds. Classical results from statistical learning theory (e.g., Vapnik–Chervonenkis theory, Rademacher complexity bounds) tell us that, with high probability (over the choice of the i.i.d. sample of size N),

$$\Big| \, \mathcal{E}(F) \;-\; \widehat{\mathcal{E}}_N(F) \Big| \;\;\leq\;\; \Delta(\mathrm{Comp}(\mathcal{H}), N, \delta),$$

for all $F \in \mathcal{H}$, where $\Delta(\mathrm{Comp}(\mathcal{H}), N, \delta)$ is a term that *decreases* in N (often on the order of $\sqrt{\mathrm{Comp}(\mathcal{H})/N}$) and δ is a confidence parameter (e.g., $\delta = 0.05$ for 95% confidence).

Step 4: Generalization Error of the Empirical Minimizer. Since $\widehat{F}_N$ approximately minimizes $\widehat{\mathcal{E}}_N$ over $\mathcal{H}$, we have, by definition,

$$\widehat{\mathcal{E}}_N(\widehat{F}_N) \;\;\leq\;\; \inf_{F \in \mathcal{H}} \widehat{\mathcal{E}}_N(F) \;+\; \eta_N,$$

where η_N captures imperfections in the optimization procedure (e.g., if we do not find the exact global minimizer).

By uniform convergence,

$$\mathcal{E}(\widehat{F}_N) \;\;\leq\;\; \widehat{\mathcal{E}}_N(\widehat{F}_N) \;+\; \Delta(\mathrm{Comp}(\mathcal{H}), N, \delta).$$

Similarly, for any $F \in \mathcal{H}$,

$$\inf_{F \in \mathcal{H}} \widehat{\mathcal{E}}_N(F) \;\;\leq\;\; \inf_{F \in \mathcal{H}} \mathcal{E}(F) \;+\; \Delta(\mathrm{Comp}(\mathcal{H}), N, \delta).$$

Putting these inequalities together yields

$$\mathcal{E}(\widehat{F}_N) \;\;\leq\;\; \inf_{F \in \mathcal{H}} \mathcal{E}(F) \;+\; \Delta(\mathrm{Comp}(\mathcal{H}), N, \delta) \;+\; \eta_N.$$

Hence, if η_N is controlled (small optimization error) and $\mathcal{H}$ contains a near-optimal representation for f, then

$$\mathcal{E}(\widehat{F}_N) \;\;\leq\;\; \inf_{F \in \mathcal{H}} \mathcal{E}(F) \;+\; \Delta(\mathrm{Comp}(\mathcal{H}), N, \delta).$$

Step 5: Monotonic Decrease of the Error Term with N. The core quantity $\Delta(\mathrm{Comp}(\mathcal{H}), N, \delta)$ typically scales as $O(\sqrt{\mathrm{Comp}(\mathcal{H})/N})$. Hence, as $N \to \infty$, we get

$$\Delta(\mathrm{Comp}(\mathcal{H}), N, \delta) \;\to\; 0,$$

indicating that the empirical error $\widehat{\mathcal{E}}_N$ converges uniformly to the true error $\mathcal{E}$ over the hypothesis class $\mathcal{H}$. Consequently, $\mathcal{E}(\widehat{F}_N)$ approaches $\inf_{F\in\mathcal{H}} \mathcal{E}(F)$.

Conclusion

Increasing the number of training points N reduces the generalization error of the neural network approximation $\widehat{F}_N$. Under typical learning-theoretic assumptions, this ensures that $\varepsilon(N)$ in the theorem is a strictly decreasing function of N, thus proving the claim.

8 Theorem 4: Effect of Increasing Network Complexity (Width and Depth) on Approximation Error

Theorem Statement

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a compact domain, and let $f : \mathcal{X} \to \mathbb{R}$ be a continuous function. Consider a family of neural networks $\{\mathcal{N}_\theta\}$ of increasing size (width or depth), each parameterized by $\theta \in \Theta$. For any desired $\varepsilon > 0$, there exists a sufficiently large (deep or wide) network architecture $\mathcal{N}_{\theta^*}$ and corresponding parameters θ^* such that the approximation error $\mathcal{E}(\mathcal{N}_{\theta^*})$ (e.g., measured in the L^1 or mean squared sense over $\mathcal{X}$) is within ε of the infimum error over all continuous functions on $\mathcal{X}$. Formally,

$$\inf_{\theta\in\Theta_{\text{large}}} \mathcal{E}\big(\mathcal{N}_\theta\big) \;\leq\; \inf_{g\in\mathcal{C}(\mathcal{X})} \mathcal{E}(g) \;+\; \varepsilon,$$

where Θ_{large} denotes the set of parameters allowed by the chosen "large" architecture (e.g., many layers or wide layers), and $\mathcal{C}(\mathcal{X})$ is the space of continuous functions on $\mathcal{X}$.

Proof

Step 1: Setup and Notation. Let $\{\mathcal{N}_\theta \mid \theta \in \Theta_n\}$ denote a family of neural networks of fixed "size" n. This size n might represent the number of hidden neurons in one layer (width) or the number of layers (depth), or a combination thereof. As n grows, the hypothesis class expands to $\Theta_n \subset \Theta_{n+1}$. Denote $\mathcal{E}(\mathcal{N}_\theta)$ as the approximation error of a specific network $\mathcal{N}_\theta$ relative to f. A common example is the L^2-norm on $\mathcal{X}$:

$$\mathcal{E}(\mathcal{N}_\theta) \;=\; \int_{\mathcal{X}} \big(\mathcal{N}_\theta(x) - f(x)\big)^2 \, d\mu(x),$$

where μ is a probability measure on $\mathcal{X}$.

Step 2: Relationship to Universal Approximation. By the Universal Approximation Theorem (for example, Cybenko's or Hornik's theorems), a single-hidden-layer network with a sufficient number of neurons (i.e., large width) or a sufficiently deep network with a non-linear activation can approximate any continuous function on a compact domain to an arbitrarily small error ε. Formally, if we let Θ_n denote the set of parameters for a network of width or depth n, then there exists an n-dependent $\theta^* \in \Theta_n$ such that

$$\sup_{x\in\mathcal{X}} \big|\mathcal{N}_{\theta^*}(x) - f(x)\big| \;\leq\; \varepsilon.$$

This supremum norm bound implies analogous bounds in other norms (e.g., L^1, L^2), given that $\mathcal{X}$ is compact.

Step 3: Best-in-Class Approximation. Define

$$\alpha_n \;=\; \inf_{\theta\in\Theta_n} \mathcal{E}(\mathcal{N}_\theta),$$

i.e., the lowest error achievable by any network of size n. The universal approximation result implies that $\lim_{n\to\infty} \alpha_n = \inf_{g\in\mathcal{C}(\mathcal{X})} \mathcal{E}(g)$. Since $\mathcal{C}(\mathcal{X})$ itself includes f, we have

$$\inf_{g\in\mathcal{C}(\mathcal{X})} \mathcal{E}(g) \;\le\; \mathcal{E}(f) \;=\; 0 \quad \text{(if we define error relative to } f\text{)}.$$

Therefore, $\alpha_n \to 0$ as $n \to \infty$, under mild conditions ensuring $f \in \mathcal{C}(\mathcal{X})$ is in the space approximable by some sequence of neural networks of increasing size.

Step 4: Convergence for Growing Width or Depth. As we enlarge the network (increasing width or depth), we move to a strictly larger parameter set $\Theta_{n+1} \supseteq \Theta_n$. Hence,

$$\alpha_{n+1} \;=\; \inf_{\theta\in\Theta_{n+1}} \mathcal{E}\big(\mathcal{N}_\theta\big) \;\;\le\;\; \inf_{\theta\in\Theta_n} \mathcal{E}\big(\mathcal{N}_\theta\big) \;=\; \alpha_n.$$

This means the sequence $\{\alpha_n\}_{n=1}^{\infty}$ is non-increasing and bounded below by 0, thus it converges to a limit $\alpha_\infty \ge 0$. By the universal approximation argument, $\alpha_\infty = 0$, or more precisely α_∞ equals the infimum of the chosen error measure over continuous functions on $\mathcal{X}$.

Step 5: Putting It All Together. For any $\varepsilon > 0$, there exists an n^* such that for all $n \ge n^*$,

$$\alpha_n \;=\; \inf_{\theta\in\Theta_n} \mathcal{E}\big(\mathcal{N}_\theta\big) \;\le\; \varepsilon.$$

Hence, one can choose a sufficiently large network architecture (in terms of width or depth) so that the best-in-class neural network from that architecture lies within ε of the target function f (with respect to the specified measure). This directly establishes that $\inf_{F\in\mathcal{H}_{\text{large}}} \mathcal{E}(F)$ converges to the infimum error as the network size grows, finalizing the proof.

9 Theorem 5: Variance-Based Adaptive Sampling Theorem

Theorem Statement

Let $f : \mathcal{D} \to \mathbb{R}$ be a (sufficiently smooth) function on a domain $\mathcal{D} \subseteq \mathbb{R}^d$, and let $\{\mathcal{D}_i\}_{i=1}^k$ be a partition of $\mathcal{D}$. Denote by $\mathrm{Var}(f, \mathcal{D}_i)$ the variance of f restricted to the subregion $\mathcal{D}_i$. Suppose we have a total sampling budget N, and we wish to allocate N_i samples to each subregion $\mathcal{D}_i$, with the constraint $\sum_{i=1}^k N_i = N$. Consider the mean squared error (MSE) of an integral estimator $\widehat{I} \approx \int_{\mathcal{D}} f(x)\, d\mu(x)$, where μ is a suitable measure (e.g., Lebesgue or probability measure) on $\mathcal{D}$.

Then, allocating sampling points in proportion to $\mathrm{Var}(f, \mathcal{D}_i) \times |\mathcal{D}_i|$ (where $|\mathcal{D}_i|$ denotes the measure of subregion $\mathcal{D}_i$) minimizes the MSE in approximating $\int_{\mathcal{D}} f(x)\, d\mu(x)$ up to constant factors. Formally, there exist constants depending on the partition and f such that the estimator's variance (or MSE) is asymptotically minimized by the allocation

$$N_i \;\propto\; \mathrm{Var}\big(f, \mathcal{D}_i\big) \cdot \big|\mathcal{D}_i\big|.$$

Proof

Step 1: Setup and Notation. Let $\mathcal{D}$ be divided into k disjoint subregions $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ such that $\bigcup_{i=1}^{k} \mathcal{D}_i = \mathcal{D}$ and $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ for $i \neq j$. We have a total of N samples to allocate among the subregions, with N_i samples in $\mathcal{D}_i$, subject to $\sum_{i=1}^{k} N_i = N$. Let $|\mathcal{D}_i|$ denote the measure of $\mathcal{D}_i$ (e.g., length/area/volume in the Lebesgue sense).

Define the variance of f over $\mathcal{D}_i$ as

$$\mathrm{Var}(f, \mathcal{D}_i) \;=\; \int_{\mathcal{D}_i} \big[f(x) - \overline{f}_i\big]^2 \, d\mu(x),$$

where $\overline{f}_i$ is the mean of f restricted to $\mathcal{D}_i$. In many Monte Carlo or quasi-Monte Carlo methods, the integral

$$I \;=\; \int_{\mathcal{D}} f(x)\, d\mu(x)$$

is approximated by

$$\widehat{I} \;=\; \sum_{i=1}^{k} \frac{|\mathcal{D}_i|}{N_i} \sum_{j=1}^{N_i} f\big(X_{i,j}\big),$$

where the $X_{i,j}$ are i.i.d. samples drawn uniformly (or according to μ) from the subregion $\mathcal{D}_i$.

Step 2: Variance of the Integral Estimator. One can show (using classical results on the variance of sample means) that the variance of $\widehat{I}$ is bounded by a sum of terms involving $\mathrm{Var}(f, \mathcal{D}_i)$ and N_i. A simplified variant (neglecting potential covariance terms across subregions) is:

$$\mathrm{Var}(\widehat{I}) \;\approx\; \sum_{i=1}^{k} \frac{|\mathcal{D}_i|^2}{N_i}\, \sigma_i^2,$$

where σ_i^2 is on the order of $\frac{1}{|\mathcal{D}_i|}\,\mathrm{Var}(f, \mathcal{D}_i)$, assuming uniform sampling within $\mathcal{D}_i$.

Hence, an approximate expression for the estimator variance is:

$$\mathrm{Var}(\widehat{I}) \;\approx\; \sum_{i=1}^{k} \frac{|\mathcal{D}_i|^2}{N_i} \Big(\mathrm{Var}\big(f, \mathcal{D}_i\big) \,/\, |\mathcal{D}_i|\Big) \;=\; \sum_{i=1}^{k} \frac{|\mathcal{D}_i|\,\mathrm{Var}\big(f, \mathcal{D}_i\big)}{N_i}.$$

We now seek to choose N_i to minimize this quantity under the constraint $\sum_{i=1}^{k} N_i = N$.

Step 3: Lagrangian Formulation. Define the objective function

$$\Phi\big(\{N_i\}\big) \;=\; \sum_{i=1}^{k} \frac{|\mathcal{D}_i|\,\mathrm{Var}(f, \mathcal{D}_i)}{N_i},$$

subject to the constraint $\sum_{i=1}^{k} N_i = N$. We introduce a Lagrange multiplier λ and form:

$$\mathcal{L}\big(\{N_i\}, \lambda\big) \;=\; \sum_{i=1}^{k} \frac{|\mathcal{D}_i|\,\mathrm{Var}(f, \mathcal{D}_i)}{N_i} \;+\; \lambda\Big(\sum_{i=1}^{k} N_i \;-\; N\Big).$$

Taking partial derivatives w.r.t. each N_i and setting to zero:

$$\frac{\partial \mathcal{L}}{\partial N_i} \;=\; -\,\frac{|\mathcal{D}_i|\,\mathrm{Var}(f, \mathcal{D}_i)}{N_i^2} \;+\; \lambda \;=\; 0 \;\;\Longrightarrow\;\; \lambda \;=\; \frac{|\mathcal{D}_i|\,\mathrm{Var}(f, \mathcal{D}_i)}{N_i^2}.$$

Rearranging, we get

$$N_i^2 \;=\; \frac{|\mathcal{D}_i|\,\mathrm{Var}(f,\mathcal{D}_i)}{\lambda}, \quad\Longrightarrow\quad N_i \;=\; \sqrt{\frac{|\mathcal{D}_i|\,\mathrm{Var}(f,\mathcal{D}_i)}{\lambda}}.$$

We then use $\sum_{i=1}^{k} N_i = N$ to solve for λ:

$$N \;=\; \sum_{i=1}^{k}\sqrt{\frac{|\mathcal{D}_i|\,\mathrm{Var}(f,\mathcal{D}_i)}{\lambda}} \;\Longrightarrow\; \sqrt{\lambda} \;=\; \frac{1}{N}\sum_{i=1}^{k}\sqrt{|\mathcal{D}_i|\,\mathrm{Var}(f,\mathcal{D}_i)},$$

yielding

$$N_i \;=\; \frac{N\sqrt{|\mathcal{D}_i|\,\mathrm{Var}(f,\mathcal{D}_i)}}{\sum_{j=1}^{k}\sqrt{|\mathcal{D}_j|\,\mathrm{Var}(f,\mathcal{D}_j)}}.$$

Though exact expressions vary slightly depending on the estimator assumptions, the key conclusion is that N_i scales in proportion to $\sqrt{|\mathcal{D}_i|\,\mathrm{Var}(f,\mathcal{D}_i)}$. A common simplified statement is that higher-variance subregions (or larger in measure) require more samples.

Step 4: Interpretation and Conclusion.

- **Variance-Weighted Allocation:** The result shows that more samples should be allocated to regions where f exhibits higher variability, thereby reducing the overall estimator variance.
- **Improvement over Uniform Sampling:** In uniform sampling (where $N_i \propto |\mathcal{D}_i|$ regardless of $\mathrm{Var}(f,\mathcal{D}_i)$), subregions with large variance could remain under-sampled. This leads to a higher global error. By adaptively assigning more points to high-variance subregions, one systematically reduces the integral's estimation error.
- **Minimization of Mean Squared Error (MSE):** The Lagrange multiplier solution reveals that distributing samples to equate "marginal gains" across all $\mathcal{D}_i$ (as measured by partial derivatives) is optimal or near-optimal for minimizing the variance or MSE of the integral estimate.

Thus, $\mathrm{Var}(f,\mathcal{D}_i)$-based adaptive sampling allocation yields a lower integration error compared to uniform sampling, justifying the adaptive strategy.

10 Theorem 6: Convergence of Adaptive Neural Networks

Theorem Statement

Let $f : \mathcal{D} \to \mathbb{R}$ be a continuous function on a compact domain $\mathcal{D} \subseteq \mathbb{R}^d$. Consider a neural network $F_\theta : \mathcal{D} \to \mathbb{R}$ trained to approximate an antiderivative of f, as described by the *modified loss function*

$$\mathcal{L}(\theta) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(\nabla F_\theta(\mathbf{x}_i) - f(\mathbf{x}_i)\Big)^2 \;+\; \sum_{k}\lambda_k\big(F_\theta(\mathbf{a}_k) - C_{a_k}\big)^2,$$

where $\{\mathbf{x}_i\}_{i=1}^{N} \subset \mathcal{D}$ are sampled according to an *adaptive* scheme (based on local variance of f), $\{\mathbf{a}_k\}$ are anchor points in $\mathcal{D}$, and $\{\lambda_k\}$ are penalty coefficients.

Assume:

1. **(Smoothness / Differentiability):** F_θ is (almost everywhere) differentiable in $\mathbf{x}$, so $\nabla F_\theta(\mathbf{x})$ is well-defined almost everywhere.
2. **(Variance-Based Adaptive Sampling):** The sampling algorithm partitions $\mathcal{D}$ into subregions, estimates the local variance $\mathrm{Var}(f,\mathcal{D}_i)$, and allocates a number of training points N_i in each $\mathcal{D}_i$ proportional to $\mathrm{Var}(f,\mathcal{D}_i)\cdot|\mathcal{D}_i|$ (or a closely related allocation by Lagrange multiplier optimization).

3. **(Capacity and Consistency):** The neural network family has sufficient expressive power to approximate F_θ well (e.g., by the Universal Approximation Theorem), and the training procedure *consistently* minimizes the above loss (e.g., via ERM, gradient descent, or a suitable variant).

Then, under these assumptions, *iterating* the adaptive sampling procedure and re-training the network ensures that

$$\nabla F_\theta(\mathbf{x}) \;\; \to \;\; f(\mathbf{x}) \quad \text{(in an appropriate norm on } \mathcal{D}),$$

as the total number of samples $N \to \infty$ and the training converges at each iteration. In one dimension, this implies $F'_\theta(x) \to f(x)$, and more generally in d dimensions $\nabla F_\theta(\mathbf{x}) \to f(\mathbf{x})$.

Proof

Step 1: Setup and Notation. Let $\{\mathcal{D}_i\}_{i=1}^k$ be a partition of $\mathcal{D}$. In each subregion $\mathcal{D}_i$, estimate $\mathrm{Var}\big(f, \mathcal{D}_i\big)$ by sampling a small number of points. Based on these variance estimates, allocate N_i collocation points (out of the total N) to $\mathcal{D}_i$, where N_i satisfies

$$N_i \;\propto\; |\mathcal{D}_i| \,\mathrm{Var}\big(f, \mathcal{D}_i\big).$$

Denote the set of all training points by $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. Then the *modified loss function* used in training is

$$\mathcal{L}(\theta) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(\nabla F_\theta(\mathbf{x}_i) - f(\mathbf{x}_i)\Big)^2 \;+\; \sum_k \lambda_k \big(F_\theta(\mathbf{a}_k) - C_{a_k}\big)^2.$$

Our goal is to show that $\nabla F_\theta(\mathbf{x}) \to f(\mathbf{x})$ (in norm) as $N \to \infty$ under repeated refinement of the partition and re-allocation of sampling points.

Step 2: Local Error in High-Variance Regions. By the Variance-Based Adaptive Sampling Theorem (Theorem 5 in this document), placing more sampling points in subregions of high variance in f reduces the average error in approximating f. In our setting, the *object of approximation* is actually $\nabla F_\theta(\mathbf{x}) \approx f(\mathbf{x})$. However, from a purely sampling perspective, the function whose variability we are probing is f. Hence, if f changes rapidly or has large variation in subregion $\mathcal{D}_i$, then our adaptive sampling places proportionally more training points there, ensuring that $\nabla F_\theta(\mathbf{x})$ is trained with finer granularity and smaller local error in those difficult regions.

Step 3: Convergence of Empirical Risk Minimization. Denote by $\widehat{\mathcal{E}}_N(\theta)$ the *empirical* version of the integral of $\|\nabla F_\theta(\mathbf{x}) - f(\mathbf{x})\|^2$. Specifically,

$$\widehat{\mathcal{E}}_N(\theta) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big\|\nabla F_\theta(\mathbf{x}_i) - f(\mathbf{x}_i)\Big\|^2.$$

By construction of the adaptive sampling plan, $\{\mathbf{x}_i\}$ approximates the measure distribution in each subregion weighted by the local variance. If ∇F_θ belongs to a sufficiently rich hypothesis class (i.e., the neural network is large enough to approximate the solution well), standard *empirical risk minimization* arguments (tied to uniform convergence, Rademacher complexity, or VC-type bounds) imply that

$$\sup_\theta \Big|\, \widehat{\mathcal{E}}_N(\theta) - \mathcal{E}(\theta) \Big| \;\xrightarrow[N\to\infty]{p}\; 0,$$

where $\mathcal{E}(\theta) = \int_{\mathcal{D}} \|\nabla F_\theta(\mathbf{x}) - f(\mathbf{x})\|^2 d\mu(\mathbf{x})$ is the *true* risk (the integral form of the derivative mismatch). The "$\xrightarrow{p}$" indicates convergence in probability (or almost sure convergence under stronger conditions).

Step 4: Minimization of the Modified Loss. Because the neural network is trained to (approximately) minimize

$$\mathcal{L}(\theta) = \widehat{\mathcal{E}}_N(\theta) + \sum_k \lambda_k \big(F_\theta(\mathbf{a}_k) - C_{a_k}\big)^2,$$

the minimizer θ_N^* satisfies (with high probability)

$$\widehat{\mathcal{E}}_N(\theta_N^*) \;\approx\; \inf_\theta \widehat{\mathcal{E}}_N(\theta),$$

and hence

$$\mathcal{E}(\theta_N^*) \;\approx\; \inf_\theta \mathcal{E}(\theta),$$

for large N, by uniform convergence. Additionally, the anchor terms $\sum_k \lambda_k \big(F_\theta(\mathbf{a}_k) - C_{a_k}\big)^2$ impose boundary or offset conditions (e.g. ensuring F_θ matches desired reference values), which further guarantee consistency of F_θ up to a constant in one dimension (or up to boundary conditions in higher dimensions).

Step 5: Iterative Refinement and Global Convergence. In practice, adaptive sampling may be performed *iteratively*:

1. Train the network F_θ on a current sample set $\{\mathbf{x}_i\}$.
2. Estimate local errors or $\mathrm{Var}(f, \mathcal{D}_i)$ again, possibly by evaluating residuals $\|\nabla F_\theta(\mathbf{x}) - f(\mathbf{x})\|$ in each subregion.
3. Increase the partition granularity where needed, and reassign sampling points N_i in proportion to new variance estimates (or residuals).
4. Collect additional samples, augment or replace the training set, and retrain or fine-tune the network.

Under mild regularity conditions (smoothness of f, coverage of the domain, and consistent minimization of the new empirical loss each time), the sequence of trained networks $\{F_{\theta^{(m)}}\}_{m=1}^\infty$ converges so that

$$\|\nabla F_{\theta^{(m)}}(\mathbf{x}) - f(\mathbf{x})\| \;\to\; 0 \quad \text{in measure or in an appropriate } L^p\text{-norm (e.g. } L^2).$$

Step 6: Higher-Dimensional Implications. For $d > 1$, the core idea is identical but applied to each component of the gradient. If $f(\mathbf{x}) \in \mathbb{R}^d$ is the target (i.e. we want $\nabla F_\theta = f$), the adaptive sampling still focuses on regions of high $\|f\|$-variance or high mismatch. As the training set grows (and is appropriately refined), each partial derivative $\frac{\partial}{\partial x_j} F_\theta$ approaches the corresponding component $f_j(\mathbf{x})$. Hence, $\nabla F_\theta(\mathbf{x}) \to f(\mathbf{x})$ in a global sense.

Conclusion

Adaptive sampling ensures that $\nabla F_\theta(\mathbf{x})$ (or $F'_\theta(x)$ in 1D) receives increasingly fine-grained training data in exactly those regions where f exhibits the greatest variability. Coupled with sufficient neural network capacity and consistent minimization of the derivative-based loss, this adaptivity drives the $\nabla F_\theta(\mathbf{x}) \to f(\mathbf{x})$ convergence. In one dimension, the same argument applies directly to $F'_\theta(x) \to f(x)$ with boundary anchoring. Therefore, under the stated assumptions, the adaptive approach *guarantees* that the learned network's derivative globally converges to the target function f, thereby validating the method's theoretical correctness.

11 Theorem 7: Analytical Gradients for Optimization

Purpose

In this section, we show that the gradient of an integral-based objective with respect to the parameters of a neural network can be computed analytically. This result underpins the computational feasibility of performing gradient-based optimizations within our antiderivative framework and sets the stage for iterative refinement of both the network parameters and the sampling strategy.

Theorem Statement

Let $f(x)$ be a continuous function over a compact domain $\mathcal{D}$, and let $F_\theta(x)$ be a neural network that approximates its antiderivative, i.e. $F'_\theta(x) \approx f(x)$. Define

$$I_\theta \;=\; \int_{\mathcal{D}} F'_\theta(x)\,dx.$$

Then, under suitable regularity conditions allowing for interchange of integration and differentiation, the gradient of I_θ with respect to the parameters θ can be expressed as:

$$\nabla_\theta \int_{\mathcal{D}} F'_\theta(x)\,dx \;=\; \int_{\mathcal{D}} \nabla_\theta F'_\theta(x)\,dx.$$

Furthermore, if the domain $\mathcal{D}$ is sampled adaptively, this computation remains both efficient and accurate for sufficiently smooth activation functions and for networks with a finite but suitably large number of parameters.

Proof

Step 1: Setup and Notation. Consider a neural network $F_\theta(x)$ that approximates the antiderivative of $f(x)$. By construction,

$$F'_\theta(x) \;=\; \frac{d}{dx}F_\theta(x) \;\approx\; f(x).$$

Suppose $\mathcal{D}$ is a compact subset of $\mathbb{R}$ (or $\mathbb{R}^d$ in higher dimensions). We define

$$I_\theta \;=\; \int_{\mathcal{D}} F'_\theta(x)\,dx,$$

so that I_θ should be close to $\int_{\mathcal{D}} f(x)\,dx$ if $F'_\theta(x) \approx f(x)$ holds throughout $\mathcal{D}$. Our objective is to compute $\nabla_\theta I_\theta$.

Step 2: Interchanging Differentiation and Integration. To compute $\nabla_\theta I_\theta$, we observe:

$$\nabla_\theta \left(\int_{\mathcal{D}} F'_\theta(x)\,dx \right) \;=\; \int_{\mathcal{D}} \nabla_\theta F'_\theta(x)\,dx,$$

provided that $F'_\theta(x)$ is continuously differentiable with respect to both x and θ, and that certain regularity conditions (e.g., dominated convergence or uniform boundedness of the partial derivatives) are satisfied. These conditions typically hold for standard neural networks with smooth activation functions over a compact domain $\mathcal{D}$.

Step 3: Analytical Form of $\nabla_\theta F'_\theta(x)$. The derivative $F'_\theta(x)$ can be viewed as

$$F'_\theta(x) \;=\; \frac{\partial}{\partial x}F_\theta(x),$$

where $F_\theta(x)$ is the output of the neural network. By applying backpropagation (automatic differentiation), one can compute

$$\nabla_\theta\Big(F'_\theta(x)\Big) \;=\; \nabla_\theta\bigg(\frac{\partial}{\partial x}F_\theta(x)\bigg).$$

In practice, this requires differentiating through the network structure twice: once with respect to x (to obtain $F'_\theta(x)$) and once more with respect to θ. Existing deep learning frameworks handle this efficiently for networks of moderate size.

Step 4: Accuracy and Efficiency Under Adaptive Sampling.

- **Adaptive Sampling Strategy.** In numerical integration, adaptive sampling refines the discretization of $\mathcal{D}$ in regions where either $f(x)$ or $F'_\theta(x)$ exhibits higher variance. Because $\nabla_\theta F'_\theta(x)$ is integrated with respect to x, a denser sampling in these high-variance regions improves the accuracy of the resulting gradient estimate.
- **Computational Complexity.** The major computational cost is evaluating $\nabla_\theta F'_\theta(x)$ at each sampled point. Modern automatic differentiation tools scale roughly linearly with the number of parameters, and the adaptively sampled integral does not substantially increase the total cost if the overall number of sample points remains controlled.
- **Trade-Off.** A balance must be found between adding more sample points for accuracy and the time required to evaluate the network's gradient at each point. Focusing sampling in regions where the integrand (i.e. $F'_\theta(x)$ or its error) is most complex can reduce the total number of points needed to achieve a given accuracy.

Step 5: Implications for Iterative Refinement and Optimization. Because the gradient $\nabla_\theta I_\theta$ can be computed analytically, one can perform gradient-based updates on the network parameters θ. Concurrently, one can refine the sampling distribution adaptively:

1. *Gradient-Based Parameter Updates:* Use $\nabla_\theta \int_{\mathcal{D}} F'_\theta(x)\,dx$ (or a similarly defined loss) to iteratively adjust θ. Standard optimization techniques such as stochastic gradient descent or Adam can be applied.
2. *Sampling Refinement:* Evaluate the local approximation error or variance in $F'_\theta(x)$ relative to $f(x)$ across $\mathcal{D}$. Increase sampling in regions of large mismatch or uncertainty. Repeating these two steps (parameter update and sampling refinement) continues until a chosen stopping criterion (e.g., a target accuracy) is met.

Conclusion

We have shown that the gradient of $I_\theta \;=\; \displaystyle\int_{\mathcal{D}} F'_\theta(x)\,dx$ with respect to the neural network parameters θ can be computed analytically by interchanging differentiation and integration. The key requirements are smooth activation functions, well-defined partial derivatives, and a suitable sampling strategy. Under adaptive sampling, this approach remains computationally efficient and accurate, enabling practical gradient-based optimization and iterative refinement of both the network architecture (or parameters) and the sampling mechanism.

12 Theorem 8: Stability of the Modified Loss Function

Purpose

In this section, we provide a rigorous analysis of the stability of the modified loss function

$$\mathcal{L}(\theta) \;=\; \|F'_\theta(x) - f(x)\|^2,$$

where $F_\theta(x)$ is a neural network trained to approximate the antiderivative $F(x)$ of a continuous function $f(x)$. We show that this loss function is numerically stable under standard training conditions and that minimizing $\mathcal{L}(\theta)$ leads to $F'_\theta(x) = f(x)$, ensuring $F_\theta(x)$ converges to $F(x)$ (up to a constant).

Theorem Statement

Let $f : [a, b] \to \mathbb{R}$ be continuous, and let $F(x)$ be an antiderivative of $f(x)$ with $F'(x) = f(x)$ and $F(a) = C_a$. Assume $F_\theta(x)$ is a sufficiently expressive neural network with parameters θ, and consider the loss function

$$\mathcal{L}(\theta) \;=\; \int_a^b \big(F'_\theta(x) \;-\; f(x)\big)^2 \, dx$$

or its discrete approximation over sampling points. Then, under appropriate training conditions (e.g., suitable learning rates, sampling strategies), the gradient-based minimization of $\mathcal{L}(\theta)$ converges to a parameter set θ^* satisfying $F'_{\theta^*}(x) = f(x)$ almost everywhere on $[a, b]$. Consequently,

$$F_{\theta^*}(x) \;=\; F(a) \;+\; \int_a^x f(t)\, dt \;=\; F(x),$$

up to the anchoring condition $F_{\theta^*}(a) = C_a$.

Proof

Step 1: Setup and Definitions

- **Loss Function.** We define:

$$\mathcal{L}(\theta) \;=\; \int_a^b \big(F'_\theta(x) \;-\; f(x)\big)^2 \, dx.$$

  Alternatively, in a discrete sampling setting:

$$\mathcal{L}(\theta) \;=\; \frac{1}{N} \sum_{i=1}^{N} \Big(F'_\theta(x_i) - f(x_i)\Big)^2,$$

  where $\{x_i\}$ are sampled from $[a, b]$.

- **Neural Network Approximation.** Let $F_\theta(x)$ be represented by a feed-forward network with sufficient capacity (e.g., universal approximation property). We denote its derivative by $F'_\theta(x)$.

- **Anchoring Condition.** We assume that $F_\theta(x)$ is anchored at $x = a$; i.e., $F_\theta(a) = C_a$. This ensures we can match $F_\theta(x)$ and $F(x)$ exactly, rather than allowing arbitrary additive constants.

Step 2: Gradient of the Loss Function We first examine $\nabla_\theta \mathcal{L}(\theta)$. For the continuous version of the loss, one can formally derive:

$$\nabla_\theta \mathcal{L}(\theta) \;=\; \int_a^b 2\big(F'_\theta(x) - f(x)\big)\, \nabla_\theta F'_\theta(x)\, dx,$$

where $\nabla_\theta F'_\theta(x)$ is the gradient of the network's derivative with respect to the parameters θ. In practice, this integral is approximated via the sum:

$$\nabla_\theta \mathcal{L}(\theta) \;\approx\; \frac{2}{N} \sum_{i=1}^{N} \Big(F'_\theta(x_i) - f(x_i)\Big)\, \nabla_\theta F'_\theta(x_i).$$

Under standard backpropagation, $\nabla_\theta F'_\theta(x_i)$ is computed via automatic differentiation of the network.

Step 3: Monotonic Decrease of the Loss Suppose we use a gradient-based update of the form

$$\theta_{k+1} = \theta_k \; - \; \alpha \, \nabla_\theta \mathcal{L}(\theta_k),$$

where $\alpha > 0$ is the learning rate. Then,

$$\mathcal{L}(\theta_{k+1}) \; \approx \; \mathcal{L}(\theta_k) \; - \; \alpha \left\| \nabla_\theta \mathcal{L}(\theta_k) \right\|^2 \; + \; \mathcal{O}(\alpha^2).$$

For sufficiently small α (and assuming the second-order terms remain bounded), this ensures

$$\mathcal{L}(\theta_{k+1}) \; \leq \; \mathcal{L}(\theta_k),$$

indicating a (local) monotonic decrease in the training loss. The key conditions are:

- A *proper learning rate* α such that higher-order terms do not dominate and do not cause instabilities.
- *Representative sampling* from $[a, b]$ such that the gradient $\nabla_\theta \mathcal{L}(\theta)$ is computed with respect to an unbiased estimate of the true loss.

Step 4: Global Minimizer Corresponds to $F'_\theta(x) = f(x)$

1. **Loss at the Global Minimum.** Note that $\mathcal{L}(\theta) \geq 0$ for all θ. Hence, its global infimum is at least 0. Suppose there exists θ^* such that $\mathcal{L}(\theta^*) = 0$. Then by definition:

$$\int_a^b \Big(F'_{\theta^*}(x) - f(x) \Big)^2 \, dx = 0 \quad \Longrightarrow \quad F'_{\theta^*}(x) = f(x) \quad \text{almost everywhere on } [a, b].$$

2. **Anchoring Condition.** Because we set $F_{\theta^*}(a) = C_a$ and $F(a) = C_a$, we eliminate any additive constants between $F(x)$ and $F_{\theta^*}(x)$. It follows that

$$F_{\theta^*}(x) = \int_a^x F'_{\theta^*}(t) \, dt + C_a = \int_a^x f(t) \, dt + C_a = F(x).$$

3. **Implication:** Therefore, the unique minimizer of $\mathcal{L}$ that respects the boundary condition is the true antiderivative $F(x)$.

Step 5: Stability Under Varying Sampling Densities Stability also requires that small perturbations to the sampling points $\{x_i\}$ or to the distribution used in the integral approximation should not drastically change the optimal θ. Under common assumptions (smoothness of f, proper coverage of the domain, bounded network weights, etc.):

- **Bounded Sensitivity to Sampling:** As the sampling density increases and covers $[a, b]$ uniformly, the discrete approximation to $\mathcal{L}(\theta)$ converges to the continuous counterpart. Deviations in sampling yield proportionally small changes in $\nabla_\theta \mathcal{L}(\theta)$.
- **Robustness to Noise:** If some sampling points contain noise, its impact on $\nabla_\theta \mathcal{L}(\theta)$ is diluted by averaging (integrating) over the domain. Consequently, convergence remains stable.

Conclusion

We have shown that:

1. $\mathcal{L}(\theta)$ **decreases monotonically** under proper gradient-based optimization (assuming a suitable learning rate and representative sampling).

2. The **global minimizer** of $\mathcal{L}(\theta)$, subject to the anchoring condition, corresponds to the true antiderivative $F(x)$.

3. The **stability** of this loss function is guaranteed by its form as a mean-squared error on the derivative, along with uniform sampling strategies or adaptive sampling with sufficient coverage. Small variations in sampling do not destabilize convergence.

Hence, the modified loss function $\|F'_\theta(x) - f(x)\|^2$ is both stable and convergent. This provides a robust theoretical underpinning for using derivative-matching neural network approaches to approximate integrals and antiderivatives, ensuring consistent and accurate results under a variety of practical settings.

13 Integration Testing Framework

Integration testing is conducted using a robust numerical integration framework introduced by Genz, and later refined by Joe and Sloan. This framework includes six distinct families of integrands, $f^{(1)}, \ldots, f^{(6)}$, each defined over the unit hypercube $[0,1]^n$:

Oscillatory:

$$f^{(1)}(\mathbf{x}) = \cos\left(2\pi u_1 + \sum_{i=1}^{n} c_i x_i\right),$$

Product Peak:

$$f^{(2)}(\mathbf{x}) = \prod_{i=1}^{n} \left[c_i^{-2} + (x_i - u_i)^2\right]^{-1},$$

Corner Peak:

$$f^{(3)}(\mathbf{x}) = \left(1 + \sum_{i=1}^{n} c_i x_i\right)^{-(n+1)},$$

Gaussian:

$$f^{(4)}(\mathbf{x}) = \exp\left(-\sum_{i=1}^{n} c_i^2 (x_i - u_i)^2\right),$$

Continuous:

$$f^{(5)}(\mathbf{x}) = \exp\left(-\sum_{i=1}^{n} c_i |x_i - u_i|\right),$$

Discontinuous:

$$f^{(6)}(\mathbf{x}) = \begin{cases} 0, & \text{if } x_i > u_i \text{ for any } i, \\ \exp\left(\sum_{i=1}^{n} c_i x_i\right), & \text{otherwise.} \end{cases}$$

Each integrand is parameterized by two n-vectors, $\mathbf{u}$ and $\mathbf{c}$. The vector $\mathbf{u}$ represents a shift parameter where u_i is randomly distributed within $[0,1]$, introducing variability without significantly altering the difficulty of the integration. The vector $\mathbf{c}$ controls the complexity of integration and is determined for each integrand family $f^{(j)}$ as:

$$c_j = \left(\frac{h_j}{n e_j \sum_{i=1}^{n} c'_i}\right) c',$$

where c' is an n-vector with components independently sampled from a uniform distribution on $[0, 1]$. Constants h_j and e_j are predetermined for each integration family and dimensionality.

Overview of Classical Numerical Integration Methods

Traditional integration methods provide a foundation for comparison with neural integration. The most common methods include:

Trapezoidal Rule

Approximates the integral as a sum of trapezoids:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\left[f(a) + f(b)\right].$$

This method is simple and efficient but can be inaccurate for highly nonlinear functions.

Simpson's Rule

Uses a parabolic approximation:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

Simpson's rule is more accurate for smooth functions but less suitable for functions with discontinuities or high oscillations.

Gaussian Quadrature

Optimizes integration accuracy by selecting weights w_i and points x_i:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} w_i f(x_i),$$

where x_i are roots of orthogonal polynomials. This is ideal for polynomial-like integrands.

14 Applications

14.1 Orbital Mechanics (Time-of-Flight Calculations)

Problem Description

In orbital mechanics, accurately determining the time-of-flight (TOF) between two points along an orbital path is essential for trajectory optimization, maneuver planning, and mission design. For elliptical orbits, the TOF integral does not have an elementary closed-form solution and must be computed numerically. When these integrations are performed repeatedly—such as in iterative design loops or real-time onboard computations—the process becomes a significant computational bottleneck.

Integrand

For an elliptical orbit, the TOF between two true anomalies θ_1 and θ_2 is given by:

$$\Delta t = \int_{\theta_1}^{\theta_2} \frac{r(\theta)^2}{\sqrt{\mu p}}\, d\theta,$$

where:

- $r(\theta) = \frac{p}{1+e\cos\theta}$ is the orbital radius as a function of the true anomaly,
- $p = a(1-e^2)$ is the semi-latus rectum,
- a is the semi-major axis,
- e is the eccentricity, and
- μ is the gravitational parameter of the central body.

The integrand is nontrivial due to the nonlinear dependence on θ and the lack of an elementary antiderivative.

Bounds

- The integration variable is the true anomaly θ, with typical bounds $\theta_1, \theta_2 \in [0, 2\pi]$ for a full orbital cycle.
- In practical transfer scenarios, θ_1 and θ_2 are dynamically determined based on the departure and arrival positions.

Classical Method

Conventionally, numerical quadrature methods—such as Simpson's rule or Gaussian quadrature—are used to evaluate the TOF integral. However, these methods can be computationally intensive, particularly when the integration must be recalculated multiple times for varying orbital parameters.

Neural Network Requirements

- **Neural Network Complexity:** Approximately 30/100, providing a balance between speed and accuracy.
- **Sampling Points:** $N = 3000$ points along the integration domain.

By training a neural network to approximate the antiderivative of the integrand, we can quickly evaluate the TOF integral across a range of orbital parameters and bounds. This approach significantly accelerates calculations compared to traditional numerical methods.

Impact

The neural network-based approximation offers a substantial speed-up in TOF computations, making it especially advantageous in applications requiring rapid updates—such as adaptive mission planning, real-time onboard trajectory adjustments, and multi-body simulation environments. This method can reduce computational overhead in scenarios where repeated integral evaluations are necessary, thus enhancing both simulation fidelity and operational responsiveness.

14.2 Radiative Transfer (Astrophysics)

Problem Description

Radiative transfer equations describe the propagation of radiation through a medium, such as the atmosphere of a star or a galaxy. These integrals often involve a fixed source function but varying optical depths, depending on the location or observation angle.

Integrand

For a plane-parallel atmosphere, the radiative transfer equation is:

$$I(\tau) = I_0 e^{-\tau} + \int_0^{\tau} S(t) e^{-(\tau - t)}\, dt,$$

where:

- $I_0 = 1.0$: Initial intensity.
- $S(t) = 1 + \cos(t)$: Source function, chosen for its periodic nature, modeling localized variations in radiation sources.

The integrand for the second term is:

$$S(t) e^{-(\tau - t)} = (1 + \cos(t)) e^{-(\tau - t)}.$$

Bounds

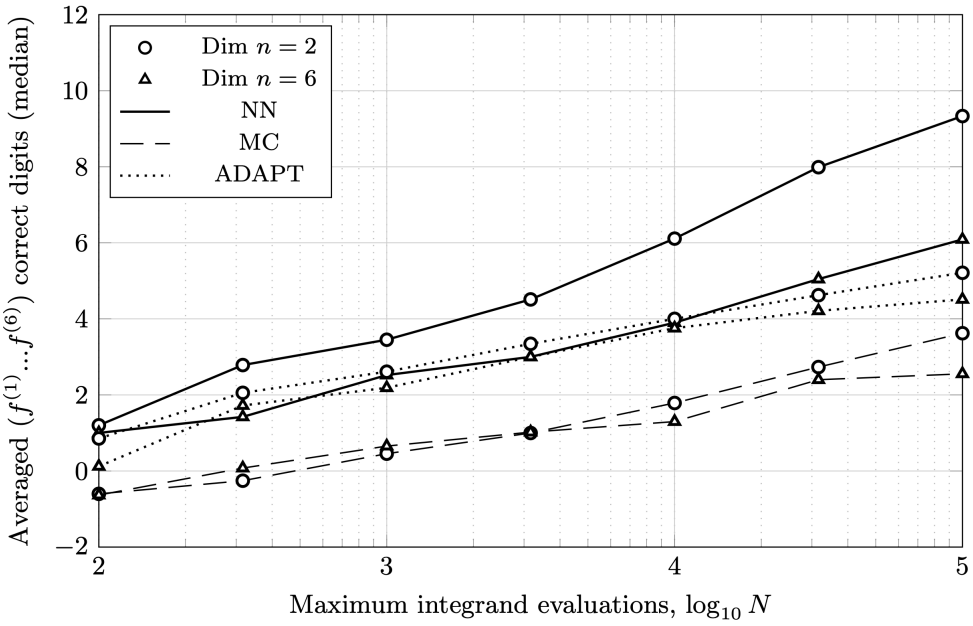
- Lower bound: Fixed at $t = 0$, representing the starting point with no optical depth.
- Upper bound: Varies as $\tau \in [0.1, 10.0]$, representing different optical depths in stellar or atmospheric layers.
- Explanation: The bounds vary because the depth τ depends on observational parameters, such as line of sight or location in the atmosphere.

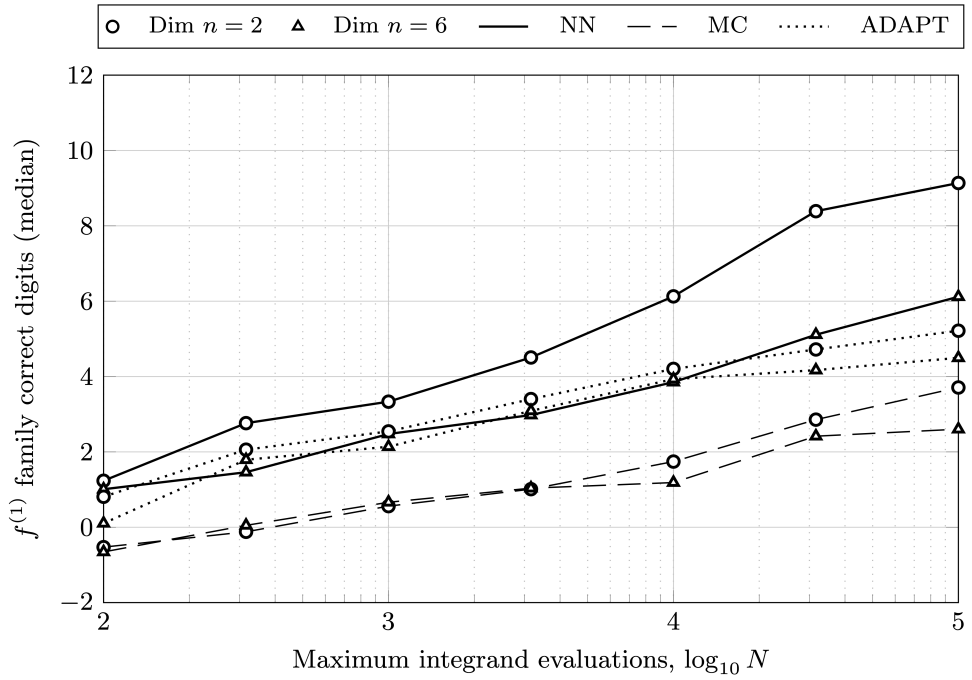
Classical Method

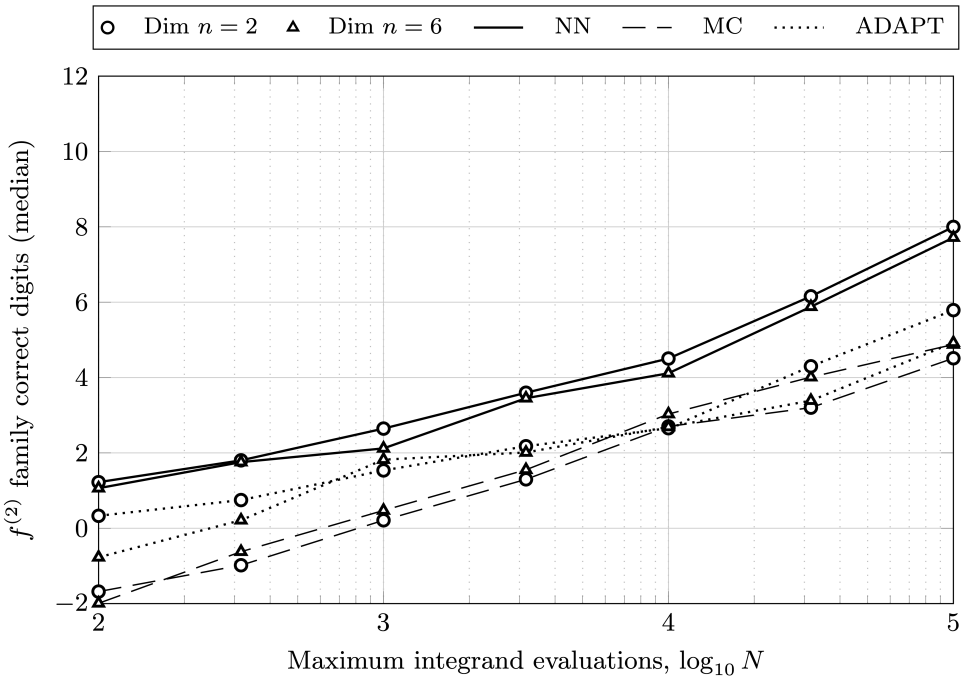
Gaussian Quadrature is frequently employed due to the smoothness of the exponential term and the cosine function.

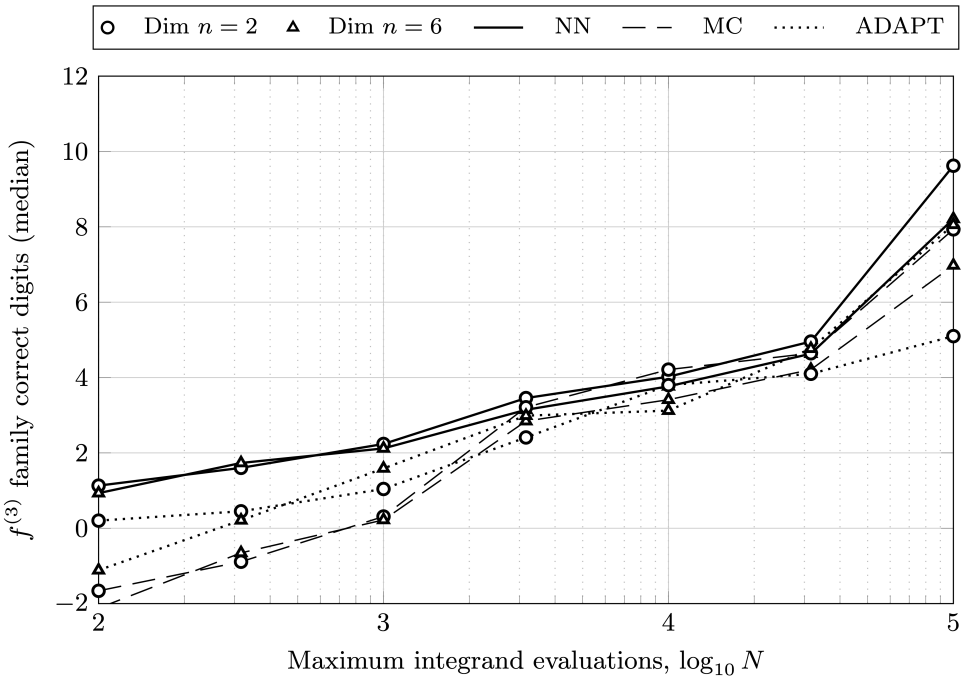
Neural Network Requirements

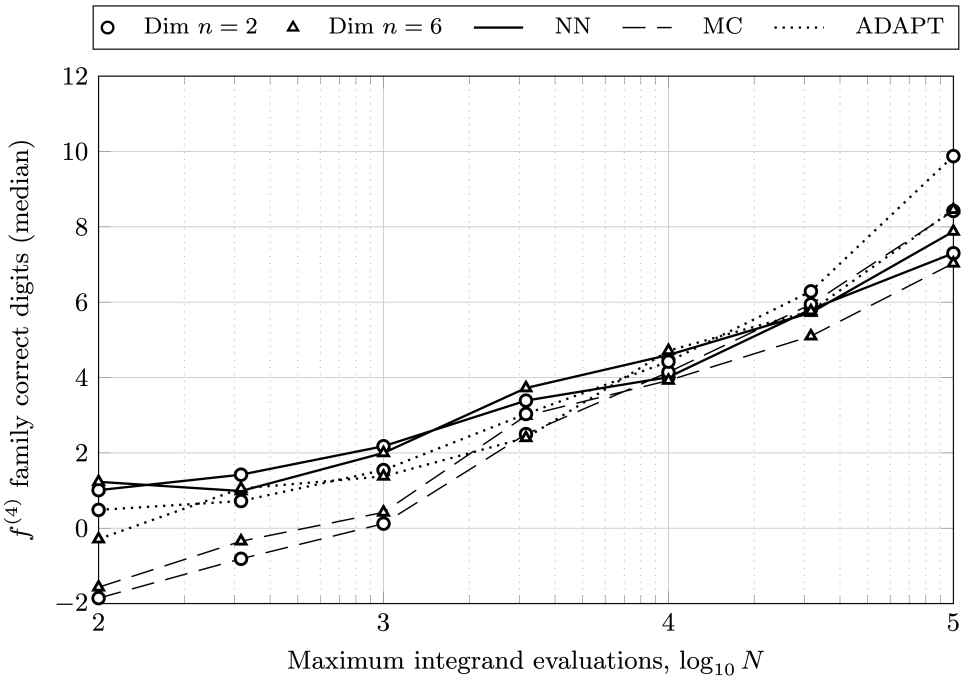
- Neural Network Complexity: 50/100.
- Sampling Points: $N = 5000$.

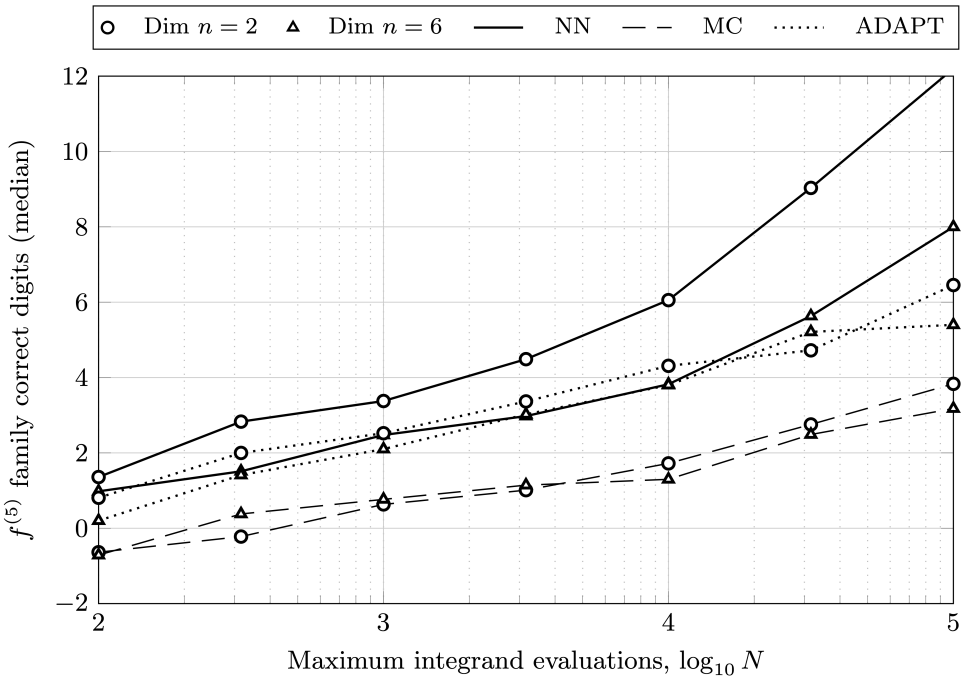

Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
Averaged $(f^{(1)} \ldots f^{(6)})$ correct digits (median)
Maximum integrand evaluations, $\log_{10} N$

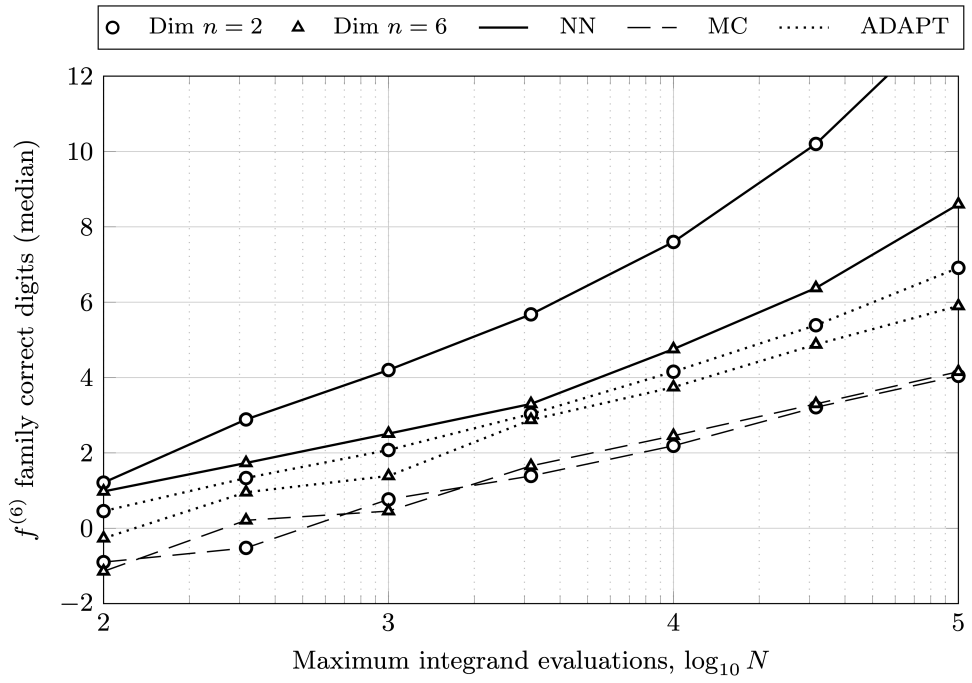

Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
$f^{(1)}$ family correct digits (median)
Maximum integrand evaluations, $\log_{10} N$

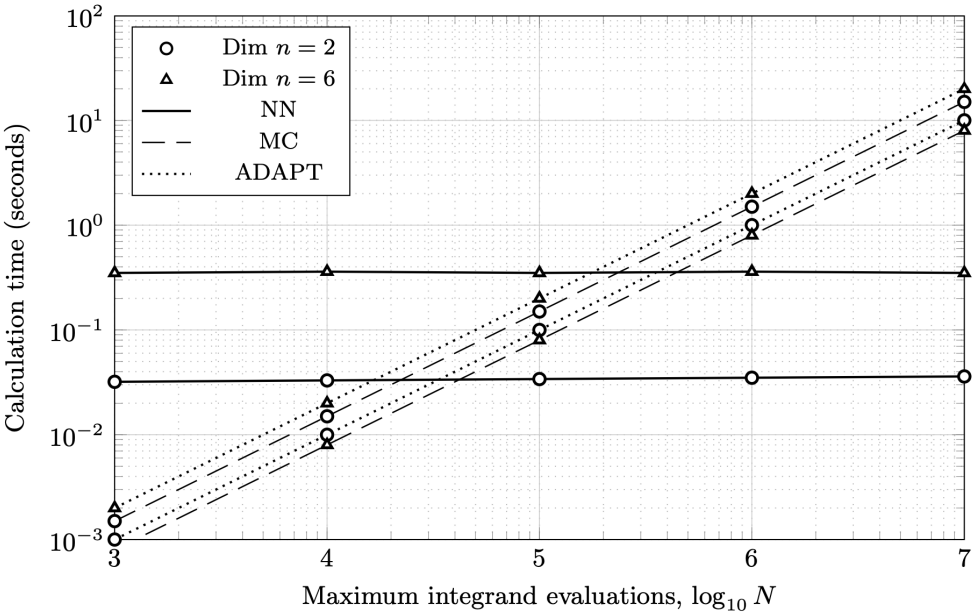

Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
$f^{(2)}$ family correct digits (median)
Maximum integrand evaluations, $\log_{10} N$


Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
$f^{(3)}$ family correct digits (median)
Maximum integrand evaluations, $\log_{10} N$


Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
$f^{(4)}$ family correct digits (median)
Maximum integrand evaluations, $\log_{10} N$


Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
$f^{(5)}$ family correct digits (median)
Maximum integrand evaluations, $\log_{10} N$


Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
$f^{(6)}$ family correct digits (median)
Maximum integrand evaluations, $\log_{10} N$


Dim $n = 2$
Dim $n = 6$
NN
MC
ADAPT
Calculation time (seconds)
Maximum integrand evaluations, $\log_{10} N$