Abstract

We introduce *Homotopy Numerical Foundations* (HNF), a new theoretical framework that reveals deep geometric structure underlying numerical computation. Our central discovery is that **precision constraints form a sheaf over the space of computations**, and that this sheaf carries homotopy-theoretic invariants that classify when computations can be accurately implemented.

The framework yields three main theorems:

The Stability Composition Theorem: Error functionals satisfy the compositional bound $\Phi_{g\circ f}(\varepsilon) \leq \Phi_g(\Phi_f(\varepsilon)) + L_g \cdot \Phi_f(\varepsilon)$, which is tight for worst-case inputs (achieved by products of linear maps). This transforms error analysis from case-by-case reasoning to automatic derivation.

The Precision Obstruction Theorem: The curvature κ_f^{curv} of a morphism provides a *sharp necessary condition* $p \geq \log_2(c \cdot \kappa D^2/\varepsilon)$ on required mantissa bits, where $c > 0$ is an explicit constant depending on smoothness assumptions. This is a lower bound: no algorithm on hardware with fewer bits can uniformly achieve ε-accuracy. Sufficiency requires algorithm-specific analysis.

The Homotopy Classification Theorem: Numerical types carry homotopy groups $\pi_n^{\text{num}}(A)$ that obstruct numerical equivalence: types with non-isomorphic homotopy groups cannot be numerically equivalent, regardless of algorithm choice.

Beyond these theorems, we develop the foundations of a new field—*numerical homotopy theory*—studying the topology of precision constraints. We define the *precision sheaf* $\mathcal{P}$ over a computation graph, prove it satisfies descent, and show its cohomology groups classify obstructions to global precision assignments. This sheaf-theoretic perspective unifies error propagation, precision requirements, and algorithmic equivalence into a single geometric framework.

Applications include certified mixed-precision neural networks, precision-aware compilation, and type systems for numerical software.

Homotopy Numerical Foundations: A Geometric Theory of Computational Precision

Anonymous

December 1, 2025

Contents

The Vision: Numerical Analysis as Geometry

This paper develops a compositional framework for numerical error analysis, viewing **computation graphs as geometric objects carrying precision constraints**.

The classical view treats numerical analysis as perturbation theory—studying how small input changes affect outputs. This yields algorithm-specific bounds that must be rederived for each composition. We propose a complementary perspective:

> *Precision constraints have algebraic structure. Error functionals compose according to explicit laws, curvature provides geometric invariants, and (more speculatively) sheaf-theoretic tools may enable local-to-global reasoning about precision.*

Concretely, we prove:

1. **Error functionals form an algebra.** The composition law $\Phi_{g\circ f}(\varepsilon) \leq \Phi_g(\Phi_f(\varepsilon)) + L_g \cdot \Phi_f(\varepsilon)$ is tight for worst-case inputs, transforming error analysis into algebra.

2. **Curvature obstructs precision.** The invariant κ_f^{curv} provides lower bounds $p \geq \log_2(c\kappa D^2/\varepsilon)$ on required bits. Combined with algorithmic upper bounds, this characterizes precision for matrix inversion, eigenvalue problems, and neural networks.

3. **Homotopy groups classify equivalence.** Numerical types carry groups $\pi_n^{\text{num}}(A)$ that obstruct numerical equivalence—a topological perspective on when algorithms are interchangeable.

4. **Precision forms a sheaf.** Over any computation graph G, precision requirements form a sheaf $\mathcal{P}_G$ whose cohomology $H^1(G; \mathcal{P}_G)$ classifies obstructions to consistent precision assignment.

Scope and limitations: We prove (1)–(3) in full rigor, with (4) developed as a precise framework with key theorems. Curvature provides *lower bounds* on precision, not complete characterizations; we are explicit about what is proved versus conjectured.

Notation Summary

Symbol	Meaning
NMet	Category of numerical types and numerical morphisms (Def. **??**)
$(A, d_A, \mathcal{R}_A)$	Numerical type: metric space with realizability data
$\mathrm{Rep}_A(\varepsilon, H)$	Finite set of ε-representations on hardware H
$\rho_{A,\varepsilon,H}$	Realization map from representations to points
$L_f, \mathrm{Lip}(f)$	Lipschitz constant of morphism f
Φ_f	Error-propagation functional of f
$\hat{f}_{\varepsilon,H}$	Machine realizer of f at precision ε on hardware H
$\mathrm{NumEquiv}(A, B)$	Numerical equivalences between types A and B
$\kappa(A)$	Condition number of matrix/operator A
κ_f^{curv}	Curvature invariant of numerical morphism f
$\mathrm{cond}_{\mathrm{eq}}(f, g)$	Equivalence condition number $L_f \cdot L_g$
$\mathrm{fl}(x)$	Floating-point representation of real x
$\mathrm{ulp}(x)$	Unit in last place at x
$\varepsilon_{\mathrm{mach}}$	Machine epsilon (typically 2^{-53} for IEEE 754 double)

Remark 0.1 (Notation Conventions). We use $\kappa(\cdot)$ for condition numbers (of matrices or operators) and κ_f^{curv} for the curvature invariant of a morphism (Definition **??**). For numerical equivalences (f, g), we write $\mathrm{cond}_{\mathrm{eq}}(f, g) := L_f \cdot L_g$ for the "equivalence condition number." These are related but distinct: the condition number $\kappa(A)$ of a matrix measures sensitivity to perturbations, while κ_f^{curv} measures nonlinearity, and $\mathrm{cond}_{\mathrm{eq}}(f, g)$ measures the "stretch" of an equivalence.

1 Introduction

1.1 The Fundamental Observation

Consider computing e^{-x} for $x = 100$ using the Taylor series:

$$e^{-100} = 1 - 100 + \frac{100^2}{2!} - \frac{100^3}{3!} + \cdots \approx 3.72 \times 10^{-44}$$

On any IEEE 754 hardware, summing this series produces garbage: the partial sums oscillate through $\pm 10^{42}$ before settling to a tiny result, losing all significant digits to cancellation. Yet $e^{-100} = 1/e^{100}$ computes accurately by first computing e^{100} (a large, stable value) and inverting.

The two algorithms compute the same function but have different precision requirements.

This paper develops a framework for analyzing such phenomena systematically. We prove compositional bounds on error propagation through computation graphs, and curvature-based lower bounds on required precision. These tools complement classical condition-number analysis.

1.2 Central Ideas

Our framework rests on three pillars:

1. **Compositional error tracking:** Error functionals Φ_f compose according to algebraic laws (Theorem **??**), enabling automatic error analysis for complex pipelines.

2. **Curvature as precision invariant:** The curvature κ_f^{curv} provides lower bounds on required mantissa bits (Theorem **??**). This is a sufficient condition for impossibility, not a complete characterization.

3. **Categorical structure:** The category **NMet** of numerical types organizes these ideas, providing a uniform language for numerical analysis.

1.3 What You Can Do With This Framework

1. Automatic Precision Budgeting. Given a computation graph, derive precision requirements at each node using Theorem **??**:

$$\mathsf{input} \xrightarrow{\text{fp64}} \mathsf{matmul}_1 \xrightarrow{\text{fp64}} \mathsf{softmax} \xrightarrow{\text{fp32}} \mathsf{matmul}_2 \xrightarrow{\text{fp16}} \mathsf{output}$$

2. Precision Lower Bounds. The curvature invariant provides necessary (not sufficient) precision requirements:

Example 1.1 (Matrix Inversion Lower Bound)**.** *For $A \in \mathbb{R}^{n\times n}$ with condition number $\kappa(A) = 10^8$, achieving relative error $< 10^{-8}$ requires mantissa precision exceeding that of fp64. This follows from the curvature bound $p \gtrsim \frac{3}{2}\log_2(\kappa)$ (Theorem **??**).*

3. Algorithmic Comparison. Different algorithms for the same function have different error functionals:

Example 1.2 (Horner vs. Naive Polynomial Evaluation)**.** *For $p(x) = a_n x^n + \cdots + a_0$:*

- Naive*: Accumulates error proportional to coefficient magnitudes*
- Horner*: Error scales with polynomial degree, not coefficient size*

*Our error functional analysis (Theorem **??**) quantifies this difference.*

4. Neural Network Quantization. Determine which layers can use reduced precision:

Example 1.3 (Transformer Precision Analysis)**.** *Using Lipschitz constants of attention and feed-forward layers, we derive which layers can be safely quantized to int8 vs. requiring fp16 (see Section **??**, Example 4).*

1.4 Main Results

We prove three main theorems:

Theorem 1.4 (Composition Law — Theorem **??**)**.** *For morphisms $f_1, \ldots, f_n$ with Lipschitz constants $L_1, \ldots, L_n$ and error functionals $\Phi_1, \ldots, \Phi_n$:*

$$\Phi_{f_n \circ \cdots \circ f_1}(\varepsilon) \leq \sum_{i=1}^{n} \left(\prod_{j=i+1}^{n} L_j \right) \cdot \Phi_i(\varepsilon_i)$$

where ε_i is determined by backward propagation. This bound is tight: we exhibit examples achieving it to within constant factors.

Theorem 1.5 (Precision Lower Bound — Theorem **??**)**.** *Under stated smoothness and boundedness assumptions, there exist* sharp lower bounds *on required precision. For a C^3 morphism f with curvature $\kappa_f > 0$ on a domain of diameter D:*

$$\boxed{p \geq \log_2(c \cdot \kappa_f \cdot D^2/\varepsilon) \text{ mantissa bits are necessary}}$$

where $c > 0$ is an explicit constant (see Theorem **??***). Below this threshold, no algorithm on that hardware can uniformly achieve ε-accuracy. See Remark* **??** *for caveats.*

Theorem 1.6 (Neural Network Representation — Theorem **??**)**.** *ReLU neural networks compute exactly the piecewise-linear maps in* **NMet***, with:*

1. *Each layer's Lipschitz constant bounded by $\|W\|_{\text{op}}$*
2. *Error propagation tracked compositionally through the network*
3. *Quantization thresholds derivable from the algebraic structure*

Consequence: *We can determine a priori which layers of a trained network can be safely quantized to int8 vs. requiring fp16.*

Theorem 1.7 (Compilation Correctness — Theorem **??**)**.** *A source-to-source transformation $G \rightsquigarrow G'$ is* precision-preserving *if the underlying maps are related by a numerical equivalence with bounded condition number. Algorithm* **??** *certifies this for a class of rewrites including algebraic simplifications, reassociations, and algorithmic substitutions.*

1.5 Classical vs. Novel Content

To orient readers: our *classical* contributions are new proofs and unified presentation of known phenomena (error propagation, condition numbers, backward stability). Our *novel* contributions are:

1. The categorical framework **NMet** making error propagation compositional
2. Sharp precision lower bounds (necessary conditions) for problem classes
3. Sheaf-theoretic perspective on precision constraints (more speculative)
4. Systematic application to neural network quantization

1.6 A Geometric Perspective on Precision

Beyond the specific theorems, we propose a geometric perspective on numerical computation:

> *Precision constraints carry geometric invariants. Curvature provides lower bounds on required bits; error functionals compose algebraically; and (more speculatively) sheaf-theoretic structures may capture global consistency of precision assignments.*

We develop three levels of this perspective:

Level 1: Curvature. The invariant κ_f^{curv} measures how nonlinearity amplifies discretization error. It provides precision lower bounds analogous to how Riemannian curvature bounds geodesic spreading.

Level 2: Sheaves. Over any computation graph G, precision requirements form a presheaf $\mathcal{P}_G^\varepsilon$. The failure of this presheaf to be a sheaf—measured by its cohomology $H^1(G; \mathcal{P}_G^\varepsilon)$—detects obstructions to uniform precision assignment. This is genuinely topological: the obstruction depends on the global structure of G, not just local constraints.

Level 3: Homotopy. The space of computations $\mathcal{C}(G)$ with the Lipschitz topology carries homotopy groups $\pi_n^{\text{comp}}(G)$ classifying computations up to precision-preserving deformation. Types with non-isomorphic homotopy groups cannot be numerically equivalent.

Section **??** develops Levels 2–3 in full rigor. We prove precise theorems where possible and clearly mark the more speculative aspects of this program.

1.7 Related Work and What Is Different Here

Classical Numerical Analysis [**?**, **?**, **?**] provides error bounds algorithm-by-algorithm. *We provide a calculus that derives these bounds automatically for arbitrary compositions.*

Homotopy Type Theory [**?**, **?**] treats equivalences as identities. *We adapt this to the quantitative setting: numerically equivalent algorithms are identified up to precision.*

Constructive Analysis [**?**] and realizability [**?**] study computability. *We extend to quantitative computability with exact error tracking.*

Neural Network Theory [**?**, **?**, **?**] characterizes expressiveness. *We add precision analysis: not just what can be computed, but at what precision.*

The key innovation: Prior work provides upper bounds ("this algorithm achieves ε-accuracy"). We provide *lower bounds* ("no algorithm can achieve better than ε-accuracy with p bits").

1.8 How to Read This Paper

For numerical analysts: Sections 2–4 are self-contained. Read the definitions, skip the categorical motivation, focus on the examples and the main theorems.

For machine learning practitioners: Section 5 (neural networks) and Section 6 (compilation) are directly applicable. The quantization results are immediately usable.

For mathematicians: The full paper develops a rigorous theory. Section 4's homotopical content is the most novel mathematically.

1.9 Organization

Section 2 develops the category **NMet** of numerical metric spaces. Section 3 proves the Composition Law for error propagation. Section 4 establishes precision impossibility theorems with worked examples. Section 5 proves the neural network representation theorem with quantization applications. Section 6 applies the theory to precision-guided compilation.

2 Gallery of Applications

Before developing the formal theory, we present a gallery of applications demonstrating the power of our framework. Each example shows a concrete problem, the HNF analysis, and the resulting insight.

2.1 Example 1: Catastrophic Cancellation in Polynomial Evaluation

Problem: Evaluate $p(x) = x^{10} - 10x^9 + 45x^8 - 120x^7 + 210x^6 - 252x^5 + 210x^4 - 120x^3 + 45x^2 - 10x + 1$ at $x = 1.00001$.

Exact answer: $p(x) = (x-1)^{10} = (0.00001)^{10} = 10^{-50}$.

Naive computation (fp64): Direct evaluation gives garbage ($\approx -10^{-5}$) due to catastrophic cancellation—intermediate terms like $x^{10} \approx 1.0001$ and $-10x^9 \approx -10.0009$ nearly cancel.

Why HNF curvature doesn't directly apply here: The Hessian-based curvature of $p(x) = (x-1)^{10}$ as a *function* is $\kappa_p^{\text{curv}} = \frac{1}{2}|p''(x)| = 45(x-1)^8$, which is tiny near $x = 1$. The problem is *not* intrinsic nonlinearity of p.

What goes wrong: The issue is that the naive *algorithm* computes p as a sequence of additions and multiplications of *large* intermediate values. Each intermediate computation has its own error contribution. The error functional analysis (Theorem **??**) shows:

$$\Phi_{\text{naive}}(\varepsilon, H) \approx \sum_{i=0}^{10} |a_i| \cdot \|x\|^i \cdot \varepsilon_H \approx 252 \cdot 1^5 \cdot \varepsilon_H$$

times the number of operations. With partial sums reaching $\pm 10^{42}$, the relative error is $\approx 10^{42} \cdot 2^{-53}/10^{-50} \approx 10^{76}$—complete loss of accuracy.

The factored form: Computing $(x-1)^{10}$ directly: let $y = x - 1 = 10^{-5}$, then compute y^{10} via repeated squaring. Here:

$$\Phi_{\text{factored}}(\varepsilon, H) = 10 \cdot \varepsilon_H \cdot |y|^{10} + O(\varepsilon_H^2)$$

The relative error is $\approx 10 \cdot 2^{-53}/|y|^{10} \cdot |y|^{10} = 10 \cdot 2^{-53}$, fully accurate in fp64.

Takeaway: Error functional analysis captures algorithmic differences that curvature of the underlying function does not. Both perspectives are valuable: curvature bounds *intrinsic* difficulty, while error functionals bound *algorithmic* difficulty.

2.2 Example 2: Mixed-Precision Deep Learning

Problem: A ResNet-50 has 50 layers. Which can use fp16 vs. requiring fp32?

HNF Analysis: Let L_i be the Lipschitz constant of layer i (spectral norm of weight matrix times ReLU Lipschitz = 1). By Corollary **??**:

$$\Phi_{\text{ResNet}}(\varepsilon, H) \leq \varepsilon \cdot \prod_{i=1}^{50} L_i + \sum_{i=1}^{50} \varepsilon_H \cdot \prod_{j=i+1}^{50} L_j$$

For spectrally normalized layers with $L_i \leq 1.02$ (standard practice):

- Error amplification: $(1.02)^{50} \approx 2.7$
- Layers 1–10 contribute $\sum_{i=1}^{10}(1.02)^{50-i} \approx 24$ times their local error
- Layers 40–50 contribute $\sum_{i=40}^{50}(1.02)^{50-i} \approx 11$ times their local error

Result: Early layers (1–10) require fp32 to maintain accuracy. Later layers (40–50) can use fp16. The classification head requires fp32 for correct logits.

Verification: This matches empirical findings in NVIDIA's automatic mixed-precision training.

2.3 Example 3: The Impossibility of Accurate Eigenvalue Computation

Problem: Compute eigenvalues of the Wilkinson matrix W_{21} (tridiagonal, entries 10, 9, 8, ..., 0, 1, 2, ..., 10 on diagonal, 1s on off-diagonals).

Classical result: W_{21} has eigenvalue pairs λ_k, λ'_k differing by $\approx 10^{-14}$ near $\lambda = 10$.

HNF Analysis: The eigenvalue map $\text{eig} : M_{21}(\mathbb{R}) \to \mathbb{R}^{21}$ has curvature $\kappa_{\text{eig}} \propto \min_i |\lambda_i - \lambda_j|^{-2}$. For nearby eigenvalues at distance 10^{-14}:

$$p_{\min} = \log_2((10^{14})^2 \cdot D^2/\varepsilon) \approx 93 + \log_2(D^2/\varepsilon) \text{ bits}$$

Result: Even for modest accuracy $\varepsilon = 10^{-8}$, we need $p \geq 93 - 27 + O(\log D) > 80$ bits—more than binary128!

Takeaway: This is not a numerical instability to be fixed; it is a *fundamental limit*. The eigenvalue problem for nearly-repeated eigenvalues is intrinsically ill-posed.

2.4 Example 4: Quantization Safety in Transformers

Problem: The Llama-2 70B model has 80 transformer layers. After quantization to int8, which layers show accuracy degradation?

HNF Analysis: The attention softmax $\text{softmax}(QK^T/\sqrt{d})$ has curvature:

$$\kappa_{\text{softmax}} = \frac{1}{2} \sup_x \|D^2 \text{softmax}_x\| = \frac{1}{2} \sup_x \|\text{diag}(s) - ss^T\| = \frac{1}{2}$$

where $s = \text{softmax}(x)$. This seems benign, but the *composition* through attention matters:

$$\kappa_{\text{attn}} = \kappa_{\text{softmax}} \cdot L_{QK^T}^2 + L_{\text{softmax}} \cdot \kappa_{QK^T}$$

For query/key matrices with spectral norm $\|Q\|, \|K\| \approx 5$ (typical after training):

$$\kappa_{\text{attn}} \approx \frac{1}{2} \cdot 25^2 + 1 \cdot 50 = 362.5$$

Precision requirement: For accuracy $\varepsilon = 10^{-3}$ (acceptable for inference):

$$p_{\min} = \log_2(362.5 \cdot D^2/10^{-3}) \approx 8 + 2\log_2 D + 10 \approx 21 \text{ bits}$$

This exceeds int8's effective 7-8 bits of precision.

Result: Attention layers *cannot* be fully quantized to int8 without accuracy loss. Feed-forward layers (curvature $\approx L_W^2 \cdot 1 \approx 4$) can safely use int8.

2.5 Example 5: Compiler Optimization Safety

Problem: An ML compiler wants to fuse operations: $\text{softmax}(\text{matmul}(Q, K))$ into a single kernel. Is this precision-safe?

HNF Analysis: Separate operations:

$$\begin{aligned}\Phi_{\text{separate}}(\varepsilon, H) &= L_{\text{softmax}} \cdot \Phi_{\text{matmul}}(\varepsilon, H) + \Phi_{\text{softmax}}(\Phi_{\text{matmul}}(\varepsilon, H), H) \\ &= 1 \cdot (\|Q\| \cdot \|K\| \cdot \varepsilon + \Delta_{\text{mm}}) + \Delta_{\text{sm}}\end{aligned}$$

Fused operation (single rounding after composition):

$$\Phi_{\text{fused}}(\varepsilon, H) = L_{\text{softmax} \circ \text{matmul}} \cdot \varepsilon + \Delta_{\text{fused}}$$

where $\Delta_{\text{fused}} < \Delta_{\text{mm}} + \Delta_{\text{sm}}$ (one less rounding step).

Result: Fusion is precision-safe if and only if the induced map $\mathcal{C}(G_{\text{separate}}) \to \mathcal{C}(G_{\text{fused}})$ is a homotopy equivalence. By Theorem **??**, this holds when the Lipschitz constants agree, which they do: $L_{\text{fused}} = L_{\text{softmax}} \cdot L_{\text{matmul}} = 1 \cdot \|Q\|\|K\|$.

Takeaway: HNF provides a formal certificate for compiler optimizations.

2.6 Example 6: The Optimal Algorithm for log-sum-exp

Problem: Compute $\text{logsumexp}(x_1, \ldots, x_n) = \log \sum_{i=1}^{n} e^{x_i}$ for x_i varying over many orders of magnitude.

Naive attempt: Direct computation overflows for $x_i > 709$ (fp64 limit for e^x).

Standard trick: $\text{logsumexp}(x) = m + \log \sum_i e^{x_i - m}$ where $m = \max_i x_i$. This avoids overflow.

HNF Analysis of both:

Naive: The exponential $x \mapsto e^x$ has curvature $\kappa = e^{x_{\max}}$, giving infinite curvature for large inputs. The computation is *intrinsically unstable*.

Shifted: The map $x \mapsto e^{x-m}$ has curvature $\kappa = e^{x_{\max} - m} = 1$ (since $x_{\max} - m = 0$). Now:

$$p_{\min} = \log_2(1 \cdot (\max_i(x_i - m))^2/\varepsilon) = O(\log(1/\varepsilon))$$

which is achievable in any precision.

Optimality: HNF proves the shifted algorithm is *optimal* in the following sense: any algorithm with finite curvature must subtract the maximum (or something equivalent). This follows because any continuous algorithm agreeing with logsumexp must have curvature at least $\sup_x |d^2(\log \sum e^{x_i})/dx^2| = \sup_x |1 - (\sum e^{x_i})^{-2}(\sum e^{x_i})(1)|$, which is unbounded for naive parameterization.

3 Numerical Metric Spaces and Realizability

We now develop the formal theory underlying the examples above.

Remark 3.1 (Metatheoretical Setting). Throughout this paper, we work in ZFC with the axiom of choice. All metric spaces are assumed complete and separable unless otherwise noted. When we construct "the" category **NMet**, we implicitly assume a Grothendieck universe to handle size issues. The realizability structures we define are *external* to the type theory: they provide a model in which types are interpreted as complete metric spaces with additional computational structure. We do not claim that the constructions are fully constructive; in particular, our use of completeness, compactness, and choice is essential in several proofs. For a constructive treatment, one would need to work with Bishop-style approaches or formal topology, which we leave to future work.

3.1 Hardware Models

Definition 3.2 (Hardware Model)**.** *A* hardware model *is a tuple* $H = (p, e_{\min}, e_{\max}, \mathcal{O})$ *where:*

(i) $p \in \mathbb{N}$ *is the* mantissa precision *(number of significand bits);*

(ii) $e_{\min}, e_{\max} \in \mathbb{Z}$ *with* $e_{\min} < e_{\max}$ *are exponent bounds;*

(iii) $\mathcal{O}$ *is a finite set of* primitive operations *(addition, multiplication, etc.) with associated rounding semantics.*

The machine epsilon *of H is $\varepsilon_H := 2^{-p}$. The* representable numbers *of H form the finite set*

$$\mathbb{F}_H := \{0\} \cup \{\pm m \cdot 2^e : m \in \{2^{p-1}, \dots, 2^p - 1\}, e \in \{e_{\min}, \dots, e_{\max}\}\}.$$

Definition 3.3 (Hardware Family)**.** *A* hardware family *$\mathcal{H}$ is a directed system of hardware models ordered by precision: $H \leq H'$ if $p_H \leq p_{H'}$, $e_{\min,H} \geq e_{\min,H'}$, and $e_{\max,H} \leq e_{\max,H'}$. We assume $\mathcal{H}$ contains models of arbitrarily high precision.*

For the remainder of this paper, we fix a hardware family $\mathcal{H}$ that includes the standard IEEE 754 formats (binary16, binary32, binary64, binary128).

Remark 3.4 (Standing Assumptions on Hardware)*.* Our theorems assume the following simplifications of IEEE 754 arithmetic:

(i) **No overflow or underflow:** We assume all intermediate values lie within the normalized range $[2^{e_{\min}}, 2^{e_{\max}}]$. This is justified when domains are bounded: for $x \in [a, b]$ with $2^{e_{\min}} \ll a \leq b \ll 2^{e_{\max}}$, overflow and underflow do not occur. When necessary, domain restrictions are stated explicitly.

(ii) **No subnormal numbers:** We ignore gradual underflow (subnormal/denormalized numbers). This simplifies analysis without affecting conclusions for values well above the subnormal threshold $\approx 2^{e_{\min}}$.

(iii) **No exceptional values:** We assume no NaNs or infinities arise in valid computations. This follows from (i) and the assumption that inputs are finite and operations are well-defined (e.g., no division by zero, no square root of negative numbers).

(iv) **Round-to-nearest-even:** We assume the IEEE 754 default rounding mode. Other rounding modes (toward zero, up, down) can be incorporated but require modified constants in error bounds.

These assumptions are standard in backward error analysis [**?**]. For applications where overflow, underflow, or exceptional values are possible, domain decomposition or explicit guards are required. See Section **??** for discussion of practical considerations.

3.2 Numerical Types

Definition 3.5 (Numerical Type)**.** *A* numerical type *is a tuple $A = (|A|, d_A, \{\mathrm{Rep}_A(\varepsilon, H)\}, \{\rho_{A,\varepsilon,H}\})$ where:*

(i) $(|A|, d_A)$ is a complete separable metric space (the underlying space*);*

*(ii) For each $\varepsilon > 0$ and $H \in \mathcal{H}$, $\mathrm{Rep}_A(\varepsilon, H)$ is a finite set (the ε-*representations on *H);*

(iii) For each ε, H, $\rho_{A,\varepsilon,H} : \mathrm{Rep}_A(\varepsilon, H) \to |A|$ is a function (the realization map*);*

subject to the following axioms:

(Approximability) *For every $a \in |A|$, $\varepsilon > 0$, and $H \in \mathcal{H}$ with $\varepsilon_H < \varepsilon$, there exists $r \in \mathrm{Rep}_A(\varepsilon, H)$ with $d_A(\rho_{A,\varepsilon,H}(r), a) < \varepsilon$.*

(Coherence) *For $\varepsilon' < \varepsilon$ and $H \leq H'$, there exist* coercion maps

$$c^H_{\varepsilon,\varepsilon'} : \mathrm{Rep}_A(\varepsilon, H) \to \mathrm{Rep}_A(\varepsilon', H), \quad c^{H'}_H : \mathrm{Rep}_A(\varepsilon, H) \to \mathrm{Rep}_A(\varepsilon, H')$$

such that $d_A(\rho_{A,\varepsilon',H}(c^H_{\varepsilon,\varepsilon'}(r)), \rho_{A,\varepsilon,H}(r)) < \varepsilon$ and similarly for $c^{H'}_H$.

(Computability) *The sets $\mathrm{Rep}_A(\varepsilon, H)$ and maps $\rho_{A,\varepsilon,H}, c^H_{\varepsilon,\varepsilon'}, c^{H'}_H$ are computable relative to H.*

Remark 3.6 (On the Computability Axiom). The computability axiom is stated informally: we do not develop a formal theory of computation over floating-point hardware in this paper. For readers not concerned with constructive or effective aspects, this axiom may be interpreted as: "the representation sets and maps can be defined by explicit algorithms." In the examples we consider (IEEE 754 floating-point, standard tensor operations), this is immediate from the explicit descriptions. A full formalization would require specifying a model of computation over hardware types H, which we leave to future work on computational aspects of HNF.

Remark 3.7 (Realizability Limits). The representation sets $\mathrm{Rep}_A(\varepsilon, H)$ form an inverse system as $\varepsilon \to 0^+$ and $H \to \infty$ (in hardware capability). The inverse limit $\varprojlim_{\varepsilon,H} \mathrm{Rep}_A(\varepsilon, H)$ may be viewed as the space of "ideal exact representations," but this limit is typically not computable. The key point is that we *never* take this limit in practice: all HNF constructions work at finite precision $\varepsilon > 0$. The underlying space $|A|$ serves as a mathematical idealization for stating theorems, while actual computations occur in $\mathrm{Rep}_A(\varepsilon, H)$.

Formally, the realization maps $\rho_{A,\varepsilon,H}$ are consistent with the coercions: $\rho_{A,\varepsilon',H} \circ c^H_{\varepsilon,\varepsilon'} = \rho_{A,\varepsilon,H}$ up to error ε. Thus the diagram

$$\begin{array}{ccc} \mathrm{Rep}_A(\varepsilon, H) & \xrightarrow{c_{\varepsilon,\varepsilon'}} & \mathrm{Rep}_A(\varepsilon', H) \\ & \searrow^{\rho_{\varepsilon,H}} & \downarrow{\scriptstyle \rho_{\varepsilon',H}} \\ & & |A| \end{array}$$

commutes up to ε-error, but not strictly. This "approximate commutativity" is fundamental to the theory and explains why **NMet** has equivalence-class morphisms rather than strict equality.

Example 3.8 (Standard Numerical Types)**.**

(a) ***Reals:*** *$\mathbb{R}_{\mathrm{num}}$ has $|\mathbb{R}_{\mathrm{num}}| = \mathbb{R}$ with standard metric, $\mathrm{Rep}_{\mathbb{R}}(\varepsilon, H) = \mathbb{F}_H$, and ρ the inclusion.*

(b) ***Tensors:*** *For shape $\mathbf{n} = (n_1, \ldots, n_k) \in \mathbb{N}^k$, the numerical tensor type $\mathcal{T}_{\mathbf{n}}$ has underlying space $\mathbb{R}^{n_1 \times \cdots \times n_k}$ with Frobenius metric, and $\mathrm{Rep}_{\mathcal{T}_{\mathbf{n}}}(\varepsilon, H) = \mathbb{F}_H^{n_1 \times \cdots \times n_k}$.*

(c) ***Banach Spaces:*** *A separable Banach space $(X, \|\cdot\|)$ yields a numerical type with $\mathrm{Rep}_X(\varepsilon, H)$ given by finite linear combinations of basis vectors with coefficients in $\mathbb{F}_H$.*

Example 3.9 (Domain Restrictions for Singular Operations)**.** *Numerical morphisms with singularities require restricted domains to ensure finite Lipschitz constants and curvature. We illustrate with three common cases:*

(a) ***Reciprocal:*** *The function $x \mapsto 1/x$ has a singularity at $x = 0$. We define the numerical type*

$$\mathbb{R}^{\mathrm{num}}_{>\delta} := \{x \in \mathbb{R} : x \geq \delta\}$$

for $\delta > 0$. The reciprocal morphism $\mathrm{recip} : \mathbb{R}^{\mathrm{num}}_{>\delta} \to \mathbb{R}^{\mathrm{num}}$ then has:

- *Lipschitz constant: $L_{\mathrm{recip}} = 1/\delta^2$ (since $|1/x - 1/y| \leq |x - y|/(\delta^2)$ for $x, y \geq \delta$);*
- *Curvature: $\kappa^{\mathrm{curv}}_{\mathrm{recip}} = 1/\delta^3$ (from $|d^2(1/x)/dx^2| = 2/x^3$);*
- *Error functional: $\Phi_{\mathrm{recip}}(\varepsilon, H) = \varepsilon/\delta^2 + O(\varepsilon_H/\delta^2)$.*

The choice of δ controls the conditioning: smaller δ allows representing numbers closer to zero but with worse stability.

(b) ***Logarithm:*** *Similarly,* $\log : \mathbb{R}^{\text{num}}_{>\delta} \to \mathbb{R}^{\text{num}}$ *has:*

- *Lipschitz constant:* $L_{\log} = 1/\delta$*;*
- *Curvature:* $\kappa^{\text{curv}}_{\log} = 1/(2\delta^2)$*;*
- *The relative error of* log *is well-conditioned for* $x \geq \delta$*, matching standard library implementations.*

(c) ***Square root:*** *The map* $\sqrt{\cdot} : \mathbb{R}^{\text{num}}_{\geq 0} \to \mathbb{R}^{\text{num}}_{\geq 0}$ *is Lipschitz on all of* $[0, \infty)$ *with* $L = \infty$*, but on the restricted domain* $[\delta, \infty)$*:*

- *Lipschitz constant:* $L_{\sqrt{\cdot}} = 1/(2\sqrt{\delta})$*;*
- *Curvature:* $\kappa^{\text{curv}}_{\sqrt{\cdot}} = 1/(4\delta^{3/2})$*.*

In each case, the domain restriction is encoded in the numerical type itself, not as an external precondition. The typing discipline ensures that compositions respect these restrictions automatically.

Example 3.10 (Running Example: $\mathbb{R}^n$ with IEEE 754 Double Precision)**.** *We develop a concrete example that we will revisit throughout the paper. Let* $n = 3$ *and consider the numerical type* $\mathbb{R}^3_{\text{num}}$ *with:*

- $|\mathbb{R}^3_{\text{num}}| = \mathbb{R}^3$ *with Euclidean metric* $d(x, y) = \|x - y\|_2$*;*
- *For IEEE 754 binary64 ("double precision"),* $H = \texttt{float64}$*, we have* $\varepsilon_H = 2^{-53} \approx 1.1 \times 10^{-16}$*;*
- $\text{Rep}_{\mathbb{R}^3}(\varepsilon, \texttt{float64}) = \{(r_1, r_2, r_3) : r_i \in \mathbb{F}_{\texttt{float64}}, |r_i| \leq 1.8 \times 10^{308}\}$*;*
- $\rho_{\mathbb{R}^3, \varepsilon, \texttt{float64}}(r_1, r_2, r_3) = (r_1, r_2, r_3) \in \mathbb{R}^3$*.*

For $x = (\pi, e, \sqrt{2}) \in \mathbb{R}^3$*, the representation* $r = (\text{fl}(\pi), \text{fl}(e), \text{fl}(\sqrt{2}))$ *satisfies*

$$d(\rho(r), x) = \sqrt{(\text{fl}(\pi) - \pi)^2 + (\text{fl}(e) - e)^2 + (\text{fl}(\sqrt{2}) - \sqrt{2})^2} \leq \sqrt{3} \cdot \varepsilon_H \cdot \max_i |x_i| \approx 5.5 \times 10^{-16}.$$

We will revisit this example for morphisms (Example **??***), stability (Example* **??***), and matrix inversion (Section* **??***).*

3.3 Numerical Morphisms

Definition 3.11 (Numerical Morphism)**.** *A* numerical morphism $f : A \to B$ *between numerical types consists of:*

(i) A function $|f| : |A| \to |B|$ *on underlying spaces;*

(ii) A Lipschitz bound $L_f \in [0, \infty)$ *such that* $d_B(|f|(a), |f|(a')) \leq L_f \cdot d_A(a, a')$*;*

(iii) An error-propagation functional $\Phi_f : (0, \infty) \times \mathcal{H} \to (0, \infty)$ *satisfying the* regularity condition*: Φ_f is L_f-Lipschitz in its first argument, i.e., $|\Phi_f(\varepsilon_1, H) - \Phi_f(\varepsilon_2, H)| \leq L_f \cdot |\varepsilon_1 - \varepsilon_2|$ for all $\varepsilon_1, \varepsilon_2 > 0$ and $H \in \mathcal{H}$;*

(iv) For each $\varepsilon > 0$ *and* $H \in \mathcal{H}$*, a* realizer $\hat{f}_{\varepsilon, H} : \text{Rep}_A(\varepsilon, H) \to \text{Rep}_B(\Phi_f(\varepsilon, H), H)$*;*

subject to the ***Soundness Axiom****: for all* $r \in \mathrm{Rep}_A(\varepsilon, H)$*,*

$$d_B\big(|f|(\rho_{A,\varepsilon,H}(r)),\ \rho_{B,\Phi_f(\varepsilon,H),H}(\hat{f}_{\varepsilon,H}(r))\big) \le \Phi_f(\varepsilon, H).$$

Remark 3.12 (On the Regularity Condition). The regularity condition on Φ_f—that it be L_f-Lipschitz in ε—ensures that composition of morphisms is well-behaved (see Lemma **??**). Intuitively, it says that the error output cannot grow faster than linearly with input error, scaled by the Lipschitz constant.

Scope and justification: This condition is satisfied by all error functionals arising from standard numerical algorithms, which typically have the form $\Phi_f(\varepsilon, H) = L_f \cdot \varepsilon + \Delta_f(H)$ for some hardware-dependent term $\Delta_f(H)$. What about more complex dependencies?

- ε^2 **terms:** In some iterative algorithms, error can grow quadratically in certain regimes. However, such algorithms typically have a convergence radius within which the linear approximation dominates. For our framework, we require Φ_f to be L_f-Lipschitz *for all ε in the domain of interest.* If an algorithm has quadratic error growth for large ε, one can either restrict the domain or use a piecewise-linear envelope as Φ_f.
- **Logarithmic terms:** Some stability results involve $\log(1/\varepsilon)$ factors (e.g., in iterative refinement). Since $\log(1/\varepsilon)$ is not Lipschitz as $\varepsilon \to 0$, we absorb such terms into the hardware-dependent part $\Delta_f(H)$ by fixing ε at a scale comparable to ε_H, yielding $\log(1/\varepsilon_H) = O(p_H)$ bits, which is bounded for fixed hardware.
- **Reduction to linear bounds:** More generally, any reasonable error bound $\tilde{\Phi}_f(\varepsilon, H)$ can be replaced by the linear envelope $\Phi_f(\varepsilon, H) := L_f \cdot \varepsilon + \sup_{\varepsilon' \le \varepsilon_H} \tilde{\Phi}_f(\varepsilon', H)$, which satisfies our regularity condition and upper-bounds the original. This may lose tightness but preserves soundness.

Thus, the regularity condition is not a fundamental limitation but rather a normalization: any practical error bound can be brought into this form, possibly with a larger hardware-dependent term.

Example 3.13 (Running Example: Morphisms on $\mathbb{R}^3$)**.** *Continuing Example* **??***, consider the morphism* $\mathrm{norm} : \mathbb{R}^3_{\mathrm{num}} \to \mathbb{R}_{\mathrm{num}}$ *given by* $x \mapsto \|x\|_2$*. We have:*

- $|\mathrm{norm}|(x) = \sqrt{x_1^2 + x_2^2 + x_3^2}$*;*
- $L_{\mathrm{norm}} = 1$ *(the Euclidean norm is 1-Lipschitz);*
- $\Phi_{\mathrm{norm}}(\varepsilon, \texttt{float64}) = \varepsilon + 3\varepsilon_H \cdot \|x\|_{\max} + O(\varepsilon_H^2)$ *(accounting for squaring, summing, and square root);*
- $\widehat{\mathrm{norm}}_{\varepsilon,H}(r_1, r_2, r_3) = \mathrm{fl}(\sqrt{\mathrm{fl}(r_1^2 + \mathrm{fl}(r_2^2 + r_3^2))})$.

The soundness axiom holds: for $r = (\mathrm{fl}(\pi), \mathrm{fl}(e), \mathrm{fl}(\sqrt{2}))$*, the computed norm differs from the true norm* $\sqrt{\pi^2 + e^2 + 2} \approx 4.07$ *by at most* $\Phi_{\mathrm{norm}}(\varepsilon, \texttt{float64}) \approx 3.3 \times 10^{-15}$*. The regularity condition is satisfied since* $\Phi_{\mathrm{norm}}(\varepsilon, H) = \varepsilon + \Delta(H)$ *with* $\Delta(H)$ *independent of* ε*.*

Definition 3.14 (Domination of Error Functionals)**.** *Given error functionals* $\Phi, \Psi : (0, \infty) \times \mathcal{H} \to (0, \infty)$*, we say* Φ dominates Ψ*, written* $\Psi \preceq \Phi$*, if for all* $\varepsilon > 0$ *and* $H \in \mathcal{H}$*, we have* $\Psi(\varepsilon, H) \le \Phi(\varepsilon, H)$*.*

Definition 3.15 (Equivalence of Numerical Morphisms)**.** *Two numerical morphisms $f, f' : A \to B$ are* equivalent*, written $f \sim f'$, if:*

(i) $|f| = |f'|$ *(same underlying function);*

(ii) $L_f = L_{f'}$ *(same Lipschitz constant);*

(iii) $\Phi_f \preceq C \cdot \Phi_{f'}$ *and* $\Phi_{f'} \preceq C \cdot \Phi_f$ *for some* uniform *constant $C \geq 1$, independent of (ε, H);*

(iv) The realizers satisfy: for all $\varepsilon > 0$, $H \in \mathcal{H}$, and $r \in \mathrm{Rep}_A(\varepsilon, H)$,

$$d_B\big(\rho_B(\hat{f}_{\varepsilon,H}(r)), \rho_B(\hat{f}'_{\varepsilon,H}(r))\big) \leq C' \cdot \Phi_f(\varepsilon, H)$$

for some uniform constant $C' \geq 0$.

This is an equivalence relation on numerical morphisms. The constant C in (iii) is called the equivalence ratio*; we write $f \sim_C f'$ when this constant matters.*

Remark 3.16. The error functional Φ_f encodes both the intrinsic error of the computation and the propagation of input error. For a Lipschitz map computed exactly, we would have $\Phi_f(\varepsilon, H) = L_f \cdot \varepsilon$. In practice, Φ_f also includes roundoff error:

$$\Phi_f(\varepsilon, H) = L_f \cdot \varepsilon + \Delta_f(H)$$

where $\Delta_f(H)$ is the implementation error on hardware H.

Definition 3.17 (Composition of Numerical Morphisms)**.** *Given $f : A \to B$ and $g : B \to C$, define $g \circ f : A \to C$ by:*

(i) $|g \circ f| := |g| \circ |f|$*;*

(ii) $L_{g \circ f} := L_g \cdot L_f$*;*

(iii) $\Phi_{g \circ f}(\varepsilon, H) := \Phi_g(\Phi_f(\varepsilon, H), H) + L_g \cdot \Phi_f(\varepsilon, H)$*;*

(iv) $\widehat{(g \circ f)}_{\varepsilon,H} := \hat{g}_{\Phi_f(\varepsilon,H),H} \circ \hat{f}_{\varepsilon,H}$.

Lemma 3.18 (Associativity and Unitality up to Equivalence)**.** *Composition of numerical morphisms is associative and unital* up to morphism equivalence *(Definition **??**). That is, $h \circ (g \circ f) \sim (h \circ g) \circ f$ and $f \circ \mathrm{id}_A \sim f \sim \mathrm{id}_B \circ f$. The identity morphism $\mathrm{id}_A : A \to A$ has $L_{\mathrm{id}} = 1$ and $\Phi_{\mathrm{id}}(\varepsilon, H) = \varepsilon$.*

Proof. We verify each component of the numerical morphism structure.

Step 1: Underlying functions. For morphisms $f : A \to B$, $g : B \to C$, $h : C \to D$, the underlying functions satisfy

$$|h \circ (g \circ f)| = |h| \circ (|g| \circ |f|) = (|h| \circ |g|) \circ |f| = |(h \circ g) \circ f|$$

by associativity of function composition in **Set**. This is strict equality.

Step 2: Lipschitz constants. We compute:

$$\begin{aligned} L_{h \circ (g \circ f)} &= L_h \cdot L_{g \circ f} = L_h \cdot (L_g \cdot L_f) \\ &= (L_h \cdot L_g) \cdot L_f = L_{h \circ g} \cdot L_f = L_{(h \circ g) \circ f}. \end{aligned}$$

This is strict equality by associativity of multiplication in $[0, \infty)$.

Step 3: Error functionals (equivalence, not equality). Let $\varepsilon_1 := \Phi_f(\varepsilon, H)$ and $\varepsilon_2 := \Phi_{g\circ f}(\varepsilon, H) = \Phi_g(\varepsilon_1, H) + L_g \cdot \varepsilon_1$. Then:

$$\begin{aligned}\Phi_{h\circ(g\circ f)}(\varepsilon, H) &= \Phi_h(\varepsilon_2, H) + L_h \cdot \varepsilon_2 \\ &= \Phi_h(\Phi_g(\varepsilon_1, H) + L_g\varepsilon_1, H) + L_h(\Phi_g(\varepsilon_1, H) + L_g\varepsilon_1).\end{aligned}$$

For the right-hand side:

$$\begin{aligned}\Phi_{(h\circ g)\circ f}(\varepsilon, H) &= \Phi_{h\circ g}(\varepsilon_1, H) + L_{h\circ g} \cdot \varepsilon_1 \\ &= \Phi_h(\Phi_g(\varepsilon_1, H), H) + L_h\Phi_g(\varepsilon_1, H) + L_h L_g\varepsilon_1.\end{aligned}$$

These expressions are *not* equal in general, but they are equivalent in the sense of Definition **??**. Specifically, assuming Φ_h is L_h-Lipschitz (which follows from the composition rule), we have:

$$\Phi_{h\circ(g\circ f)}(\varepsilon, H) \leq \Phi_{(h\circ g)\circ f}(\varepsilon, H) + L_h \cdot L_g \cdot \varepsilon_1.$$

The reverse bound is similar. Hence $\Phi_{h\circ(g\circ f)} \preceq C \cdot \Phi_{(h\circ g)\circ f}$ for $C = 2$, establishing equivalence of error functionals.

Step 4: Realizers (equivalence up to bounded error). The composed realizers satisfy:

$$\widehat{h \circ (g \circ f)}_{\varepsilon,H} = \hat{h}_{\varepsilon_2,H} \circ (\hat{g}_{\varepsilon_1,H} \circ \hat{f}_{\varepsilon,H}).$$

The corresponding expression for $\widehat{(h \circ g) \circ f}$ uses $\hat{h}_{\Phi_g(\varepsilon_1,H),H}$ at a different precision. The outputs differ by at most the Lipschitz constant of h times the precision difference, which is bounded by $O(\Phi_{(h\circ g)\circ f}(\varepsilon, H))$. This establishes realizer equivalence per Definition **??**(iv).

Step 5: Identity morphism. The identity $\mathrm{id}_A : A \to A$ has $|\mathrm{id}_A| = \mathrm{id}_{|A|}$, $L_{\mathrm{id}_A} = 1$, $\Phi_{\mathrm{id}_A}(\varepsilon, H) = \varepsilon$, and $\widehat{\mathrm{id}_A} = \mathrm{id}_{\mathrm{Rep}_A(\varepsilon,H)}$. For unit laws, we have $|f \circ \mathrm{id}_A| = |f|$, $L_{f\circ\mathrm{id}_A} = L_f$, but $\Phi_{f\circ\mathrm{id}_A}(\varepsilon, H) = \Phi_f(\varepsilon, H) + L_f \cdot \varepsilon \neq \Phi_f(\varepsilon, H)$ in general. However, $\Phi_{f\circ\mathrm{id}_A} \preceq 2 \cdot \Phi_f$ when $\Phi_f(\varepsilon, H) \geq L_f \cdot \varepsilon$, so $f \circ \mathrm{id}_A \sim f$. □

Remark 3.19 (Strict vs. Weak Category Structure). The preceding lemma shows that **NMet** is a category only when morphisms are taken as equivalence classes under $\sim$. Alternatively, one can view **NMet** as a *bicategory* where the 2-cells are the equivalences between numerical morphisms with the same underlying function but different error bounds. We adopt the former convention: morphisms in **NMet** are equivalence classes $[f]_\sim$, ensuring strict associativity and unitality.

Key technical assumption: In Step 3 of Lemma **??**, we use that Φ_h is Lipschitz in its first argument with constant L_h. This follows from the composition rule for error functionals when h is obtained by composing well-behaved morphisms. For morphisms defined *ab initio* (e.g., primitive operations), we impose this as a regularity condition on admissible error functionals.

Formal quotient construction: Let $\mathbf{NMet}^{\mathrm{raw}}$ denote the "raw" structure with numerical morphisms as defined (not equivalence classes). Then $\mathbf{NMet} = \mathbf{NMet}^{\mathrm{raw}}/\sim$ is the quotient where $\mathrm{Hom}_{\mathbf{NMet}}(A, B) := \mathrm{Hom}_{\mathbf{NMet}^{\mathrm{raw}}}(A, B)/\sim$. To verify this is well-defined, we check that $\sim$ is a congruence: if $f \sim f'$ and $g \sim g'$, then $g \circ f \sim g' \circ f'$. This follows from the Lipschitz property of error functionals:

$$\Phi_{g'\circ f'}(\varepsilon, H) \preceq C \cdot \Phi_{g\circ f}(\varepsilon, H)$$

where C depends only on the equivalence constants for $f \sim f'$ and $g \sim g'$.

Example 3.20 (Why Equivalence Classes Are Essential)**.** *Consider the morphism* $\mathrm{add} : \mathbb{R}^2 \to \mathbb{R}$ *given by* $(x, y) \mapsto x + y$*, implemented in two ways on IEEE 754 double precision:*

(i) ***Direct addition:*** $\widehat{\mathrm{add}}_1(r_x, r_y) = \mathrm{fl}(r_x + r_y)$ *with* $\Phi_1(\varepsilon, H) = 2\varepsilon + \varepsilon_{\mathrm{mach}}$*.*

(ii) ***Compensated addition (Kahan summation):*** $\widehat{\mathrm{add}}_2$ *computes* $(s, c) = \mathrm{TwoSum}(r_x, r_y)$ *and returns* $\mathrm{fl}(s + c)$*, giving* $\Phi_2(\varepsilon, H) = 2\varepsilon + O(\varepsilon_{\mathrm{mach}}^2)$*.*

These define the same underlying function with $L = 2$ *(by triangle inequality), but different error functionals. By Definition* **??***, they are equivalent since* $\Phi_1 \preceq 2\Phi_2$ *and* $\Phi_2 \preceq \Phi_1$ *(for* $\varepsilon \geq \varepsilon_{\mathrm{mach}}$*). Taking equivalence classes means we do not distinguish between algorithms with "essentially the same" backward error profile.*

Definition 3.21 (The Category **NMet**)**.** *The category* **NMet** *has:*

- *Objects: numerical types (Definition* **??***);*
- *Morphisms: equivalence classes of numerical morphisms under* $\sim$ *(Definition* **??***);*
- *Composition: as defined above;*
- *Identity:* id_A *with trivial Lipschitz bound and error functional.*

3.4 Numerical Equivalences

Definition 3.22 (Numerical Equivalence)**.** *A* numerical equivalence *between numerical types* A *and* B *is a pair of numerical morphisms* $f : A \to B$ *and* $g : B \to A$ *together with numerical homotopies (defined below)* $\eta : g \circ f \sim \mathrm{id}_A$ *and* $\mu : f \circ g \sim \mathrm{id}_B$*, such that additionally:*

*(i) (*Bi-Lipschitz*) There exists* $K \geq 1$ *with* $K^{-1} \leq L_f \cdot L_g \leq K$*;*

*(ii) (*Distortion Bound*) For all* $a \in |A|$*:* $d_A(|g|(|f|(a)), a) \leq D$ *for some* $D \geq 0$*;*

*(iii) (*Realizer Coherence*) The realizers satisfy* $\hat{g} \circ \hat{f} \sim \widehat{\mathrm{id}_A}$ *and* $\hat{f} \circ \hat{g} \sim \widehat{\mathrm{id}_B}$ *up to coercion.*

We write $\mathrm{NumEquiv}(A, B)$ *for the type of numerical equivalences.*

Definition 3.23 (Condition Number)**.** *For a numerical equivalence* $(f, g, \eta, \mu) : A \simeq_{\mathrm{num}} B$*, the* condition number *is*

$$\mathrm{cond}(f, g) := L_f \cdot L_g.$$

The numerical distance *between equivalent types is*

$$d_{\mathrm{num}}(A, B) := \inf_{(f,g) \in \mathrm{NumEquiv}(A,B)} \log(\mathrm{cond}(f, g)).$$

Proposition 3.24 (Numerical Distance is a Pseudometric)**.** *The numerical distance* d_{num} *defines an extended pseudometric on equivalence classes of numerical types.*

Proof. We verify the three axioms of an extended pseudometric.

Axiom 1: Non-negativity and reflexivity. By definition, $d_{\mathrm{num}}(A, B) = \inf_{(f,g)} \log(\mathrm{cond}(f, g)) \geq \log(1) = 0$ since $\mathrm{cond}(f, g) = L_f \cdot L_g \geq 1$ (as both $L_f, L_g \geq 1$ for any bi-Lipschitz equivalence with $L_f^{-1} \leq L_g \leq L_f$).

For reflexivity, the identity equivalence $(\mathrm{id}_A, \mathrm{id}_A, \mathrm{refl}, \mathrm{refl})$ has $\mathrm{cond}(\mathrm{id}_A, \mathrm{id}_A) = 1 \cdot 1 = 1$, hence $d_{\mathrm{num}}(A, A) \leq \log(1) = 0$. Combined with non-negativity, $d_{\mathrm{num}}(A, A) = 0$.

Axiom 2: Symmetry. Given a numerical equivalence $(f, g, \eta, \mu) : A \simeq_{\mathrm{num}} B$, we construct the reverse equivalence $(g, f, \mu, \eta) : B \simeq_{\mathrm{num}} A$. This satisfies:

- The underlying maps are $|g| : |B| \to |A|$ and $|f| : |A| \to |B|$, which are bi-Lipschitz by hypothesis.
- The homotopies $\mu : f \circ g \sim \mathrm{id}_B$ and $\eta : g \circ f \sim \mathrm{id}_A$ become $\eta : f \circ g \sim \mathrm{id}_A$ and $\mu : g \circ f \sim \mathrm{id}_B$ after swapping roles, which is exactly what the reverse equivalence requires (up to notation).
- The condition number satisfies $\mathrm{cond}(g, f) = L_g \cdot L_f = L_f \cdot L_g = \mathrm{cond}(f, g)$.

Taking the infimum over all equivalences:

$$d_{\mathrm{num}}(B, A) = \inf_{(f,g):B\simeq A} \log(\mathrm{cond}(f, g)) = \inf_{(g,f):A\simeq B} \log(\mathrm{cond}(g, f)) = d_{\mathrm{num}}(A, B).$$

Axiom 3: Triangle inequality. Let $(f_1, g_1, \eta_1, \mu_1) : A \simeq_{\mathrm{num}} B$ and $(f_2, g_2, \eta_2, \mu_2) : B \simeq_{\mathrm{num}} C$. We construct a composite equivalence $(f_2 \circ f_1, g_1 \circ g_2) : A \simeq_{\mathrm{num}} C$.

Step 3a: Composite maps are bi-Lipschitz. The forward map $f_2 \circ f_1 : A \to C$ has Lipschitz constant:

$$L_{f_2\circ f_1} = L_{f_2} \cdot L_{f_1}.$$

Similarly, $L_{g_1\circ g_2} = L_{g_1} \cdot L_{g_2}$. The bi-Lipschitz condition requires:

$$(L_{f_2\circ f_1})^{-1} \le L_{g_1\circ g_2} \le L_{f_2\circ f_1}.$$

Since (f_1, g_1) and (f_2, g_2) are bi-Lipschitz with constants K_1 and K_2 respectively, we have:

$$K_1^{-1} \le L_{f_1} L_{g_1} \le K_1, \quad K_2^{-1} \le L_{f_2} L_{g_2} \le K_2.$$

Therefore:

$$(L_{f_2} L_{f_1})^{-1} = L_{f_1}^{-1} L_{f_2}^{-1} \le K_1 K_2 \cdot L_{g_1}^{-1} L_{g_2}^{-1}$$

and the bi-Lipschitz condition is satisfied with constant $K_1 K_2$.

Step 3b: Homotopies compose. We construct $\eta : (g_1 \circ g_2) \circ (f_2 \circ f_1) \sim \mathrm{id}_A$. Rewriting:

$$(g_1 \circ g_2) \circ (f_2 \circ f_1) = g_1 \circ (g_2 \circ f_2) \circ f_1.$$

The homotopy $\mu_2 : f_2 \circ g_2 \sim \mathrm{id}_B$ induces $g_2 \circ f_2 \sim \mathrm{id}_B$ by symmetry (or directly from η_2). Composing with g_1 on the left and f_1 on the right:

$$g_1 \circ (g_2 \circ f_2) \circ f_1 \sim g_1 \circ \mathrm{id}_B \circ f_1 = g_1 \circ f_1 \sim \mathrm{id}_A$$

where the last homotopy is η_1. The Lipschitz bounds on the homotopy are controlled by L_{g_1}, L_{f_1}, and the Lipschitz constants of η_1, η_2.

Similarly, $\mu : (f_2 \circ f_1) \circ (g_1 \circ g_2) \sim \mathrm{id}_C$.

Step 3c: Condition number bound.

$$\mathrm{cond}(f_2 \circ f_1, g_1 \circ g_2) = L_{f_2\circ f_1} \cdot L_{g_1\circ g_2} = L_{f_2} L_{f_1} \cdot L_{g_1} L_{g_2} = \mathrm{cond}(f_1, g_1) \cdot \mathrm{cond}(f_2, g_2).$$

Step 3d: Conclusion. Taking logarithms and infima:

$$\begin{aligned} d_{\mathrm{num}}(A, C) &\le \log(\mathrm{cond}(f_2 \circ f_1, g_1 \circ g_2)) \\ &= \log(\mathrm{cond}(f_1, g_1)) + \log(\mathrm{cond}(f_2, g_2)). \end{aligned}$$

Taking the infimum over all choices of (f_1, g_1) and (f_2, g_2):

$$d_{\mathrm{num}}(A, C) \le d_{\mathrm{num}}(A, B) + d_{\mathrm{num}}(B, C).$$

Extended values. If no numerical equivalence exists between A and B, we set $d_{\mathrm{num}}(A, B) = +\infty$, which is consistent with the extended pseudometric axioms. □

3.5 Universal Property of Numerical Distance

The numerical distance d_{num} is characterized by a universal property: among all functors satisfying natural axioms (grounding, equivalence-sensitivity, subadditivity, and metric continuity), d_{num} is the largest. This result, inspired by analogous characterizations in optimal transport theory [**?**] and quantitative model checking [**?**], establishes d_{num} as the *canonical* metric for numerical equivalence.

Remark 3.25 (Scope: Layer 1 Material). **This subsection contains Layer 1 (foundational/homotopical) material.** The universal property is conceptually important for understanding why d_{num} is the "right" distance, but it is *not used* in the main numerical results (Sections **??**–**??**). Readers focused on numerical analysis may skip directly to Section **??**. For completeness, the full development including proofs appears in Appendix **??**.

The key result (Theorem **??** in the appendix) states:

> *Among all numerical satisfaction functors satisfying axioms (A1)–(A4), d_{num} is the largest: $F(A, B) \leq d_{\text{num}}(A, B)$ for all numerical types A, B.*

4 The Stability Composition Theorem

We now establish sharp bounds for error propagation through compositions of numerical morphisms. This section provides the quantitative backbone for certified numerical computation within HNF.

4.1 Error Propagation Algebra

Definition 4.1 (Error Functional Algebra)**.** *Let $\mathcal{E}$ denote the semiring of* error functionals*:*

$$\mathcal{E} := \{\Phi : (0, \infty) \times \mathcal{H} \to (0, \infty) : \Phi \text{ is monotone in } \varepsilon \text{ and antimonotone in } H\}$$

with operations:

- *Addition:* $(\Phi_1 + \Phi_2)(\varepsilon, H) := \Phi_1(\varepsilon, H) + \Phi_2(\varepsilon, H)$*;*
- *Composition:* $(\Phi_1 \circ \Phi_2)(\varepsilon, H) := \Phi_1(\Phi_2(\varepsilon, H), H)$*;*
- *Scaling:* $(L \cdot \Phi)(\varepsilon, H) := L \cdot \Phi(\varepsilon, H)$ *for* $L \geq 0$*.*

Remark 4.2 (Role of Error Functionals in HNF). We emphasize three points about error functionals that clarify the relationship between theory and practice:

(a) **Specification vs. Implementation:** The error functional Φ_f is *part of the specification* of a numerical morphism, not a derived quantity. Different algorithms computing the same mathematical function may have different error functionals (e.g., compensated vs. non-compensated summation). When we write "$f : A \to B$ with Φ_f," we mean a specific algorithm with certified error bounds.

(b) **Sound Over-Approximation:** The composition rule $\Phi_{g \circ f} \leq \Phi_g \circ \Phi_f + L_g \cdot \Phi_f$ (Lemma **??**) provides a *sound upper bound* on the actual error. This bound may be pessimistic for specific inputs but is guaranteed to hold for all inputs. The gap between bound and actual error reflects worst-case analysis.

(c) **Sharpness Interpretation:** When we prove "sharpness" of a bound (Theorem **??**(iii)), we mean there exist inputs attaining the bound to within constant factors. This is a statement about the *best possible bound of this algebraic form*, not about every algorithm or every input.

Lemma 4.3 (Subadditivity)**.** *For composable numerical morphisms f, g:*

$$\Phi_{g\circ f} \leq \Phi_g \circ \Phi_f + L_g \cdot \Phi_f.$$

Proof. Let $r \in \mathrm{Rep}_A(\varepsilon, H)$ and $a = \rho_A(r)$. Then:

$$\begin{aligned} d_C((g\circ f)(a), \rho_C(\widehat{g\circ f}(r))) &\leq d_C(g(f(a)), g(\rho_B(\hat{f}(r)))) + d_C(g(\rho_B(\hat{f}(r))), \rho_C(\hat{g}(\hat{f}(r)))) \\ &\leq L_g \cdot d_B(f(a), \rho_B(\hat{f}(r))) + \Phi_g(\Phi_f(\varepsilon, H), H) \\ &\leq L_g \cdot \Phi_f(\varepsilon, H) + \Phi_g(\Phi_f(\varepsilon, H), H). \end{aligned}$$

□

4.2 The Main Stability Theorem

Theorem 4.4 (Stability Composition)**.** *Let $f_1 : A_0 \to A_1, \ldots, f_n : A_{n-1} \to A_n$ be numerical morphisms. Set $F := f_n \circ \cdots \circ f_1$. Then:*

(i) Lipschitz Bound:

$$L_F = \prod_{i=1}^{n} L_{f_i}.$$

(ii) Error Bound:

$$\Phi_F(\varepsilon, H) \leq \sum_{i=1}^{n} \left(\prod_{j=i+1}^{n} L_{f_j} \right) \Phi_{f_i}(\varepsilon_i, H)$$

where $\varepsilon_1 = \varepsilon$ and $\varepsilon_{i+1} = \Phi_{f_i}(\varepsilon_i, H)$ (forward error analysis), or equivalently using backward analysis.

(iii) Sharpness: *There exist numerical morphisms achieving equality in both bounds.*

Proof. **Part (i):** Immediate from the definition of Lipschitz constant and composition.

Part (ii): We prove by induction. Base case $n = 1$ is trivial.

For the inductive step, let $G = f_{n-1} \circ \cdots \circ f_1$. By Lemma **??**:

$$\begin{aligned} \Phi_F &= \Phi_{f_n \circ G} \\ &\leq \Phi_{f_n} \circ \Phi_G + L_{f_n} \cdot \Phi_G \\ &\leq \Phi_{f_n}(\Phi_G(\varepsilon, H), H) + L_{f_n} \cdot \Phi_G(\varepsilon, H). \end{aligned}$$

By induction, $\Phi_G(\varepsilon, H) \leq \sum_{i=1}^{n-1} \left(\prod_{j=i+1}^{n-1} L_{f_j} \right) \Phi_{f_i}(\varepsilon_i, H)$. Substituting and simplifying yields the claimed bound.

Part (iii) - Sharpness: Consider the composition of n linear maps $f_i : \mathbb{R} \to \mathbb{R}$ given by $f_i(x) = L_i \cdot x$ with roundoff error Δ_i, implemented as floating-point multiplication. Then:

- $\Phi_{f_i}(\varepsilon, H) = L_i \cdot \varepsilon + \Delta_i$ where $\Delta_i = L_i \cdot \varepsilon_H$;
- Direct calculation shows Φ_F achieves exactly the bound.

For the Lipschitz bound, consider $f_i(x) = L_i \cdot x$ on $\mathbb{R}$; composition gives $F(x) = (\prod_i L_i) \cdot x$ with $L_F = \prod_i L_i$ exactly. □

Corollary 4.5 (Non-Expansive Composition)**.** *If $L_{f_i} \leq 1$ for all i, then $L_F \leq 1$. Moreover, if each $\Phi_{f_i}(\varepsilon, H) \leq C \cdot \varepsilon$ for some $C \geq 1$, then*

$$\Phi_F(\varepsilon, H) \leq n \cdot C \cdot \varepsilon.$$

Example 4.6 (Running Example: Stability for $\mathbb{R}^3$ Morphisms)**.** *Continuing Examples* **??** *and* **??**, *consider the composition:*

$$F := \text{norm} \circ T : \mathbb{R}^3 \to \mathbb{R}$$

where $T(x) = Ax + b$ is an affine transformation with $\|A\|_2 = 2$. We have:

- $L_T = 2$, $\Phi_T(\varepsilon, \texttt{float64}) = 2\varepsilon + 4\varepsilon_H$ *(matrix-vector multiply);*
- $L_{\text{norm}} = 1$, $\Phi_{\text{norm}}(\varepsilon) = \varepsilon + 3\varepsilon_H$ *(from Example* **??***).*

By Theorem **??***:*

$$\begin{aligned} L_F &= L_{\text{norm}} \cdot L_T = 1 \cdot 2 = 2, \\ \Phi_F(\varepsilon, H) &\leq L_{\text{norm}} \cdot \Phi_T(\varepsilon) + \Phi_{\text{norm}}(\Phi_T(\varepsilon)) \\ &= 1 \cdot (2\varepsilon + 4\varepsilon_H) + ((2\varepsilon + 4\varepsilon_H) + 3\varepsilon_H) \\ &= 4\varepsilon + 11\varepsilon_H \approx 4\varepsilon + 1.2 \times 10^{-15}. \end{aligned}$$

For input error $\varepsilon = 10^{-10}$, the total error is approximately 4×10^{-10}, consistent with the 2-Lipschitz bound.

Corollary 4.7 (Stability of Deep Networks)**.** *Let $N = f_L \circ \cdots \circ f_1$ be an L-layer neural network where each layer f_i has Lipschitz constant L_i and implementation error Δ_i. Then:*

$$\Phi_N(\varepsilon, H) \leq \varepsilon \cdot \prod_{i=1}^{L} L_i + \sum_{i=1}^{L} \Delta_i \cdot \prod_{j=i+1}^{L} L_j.$$

For a spectrally normalized network with $L_i \leq 1$, this simplifies to $\Phi_N \leq \varepsilon + \sum_i \Delta_i$.

4.3 Backward Error Analysis

Definition 4.8 (Backward Error)**.** *For a numerical morphism $f : A \to B$ and computed output $\hat{b} = \rho_B(\hat{f}(r))$, the* backward error *is*

$$\beta_f(r, H) := \inf\{d_A(a, \rho_A(r)) : f(a) = \hat{b} \text{ exactly}\}.$$

Theorem 4.9 (Forward-Backward Duality)**.** *Let $f : A \to B$ be a numerical morphism that is locally invertible near $a \in |A|$, with local inverse $g : U \to A$ defined on a neighborhood $U \ni f(a)$. Assume f is C^2 with curvature $\kappa_f^{\text{curv}}(a) \leq \kappa$. Then:*

$$\Phi_f(\varepsilon, H) = L_f \cdot \beta_f(\varepsilon, H) + R(\varepsilon)$$

where the remainder satisfies $|R(\varepsilon)| \leq \kappa \cdot L_f \cdot \|Dg_{f(a)}\|^2 \cdot \varepsilon^2$.

Proof. Let $r \in \text{Rep}_A(\varepsilon, H)$ with $\rho_A(r) = a + \delta$ for $\|\delta\| \leq \varepsilon$. By Taylor expansion:

$$f(a + \delta) = f(a) + Df_a(\delta) + \tfrac{1}{2} D^2 f_a(\delta, \delta) + O(\|\delta\|^3).$$

The computed output $\hat{b} = \hat{f}_{\varepsilon,H}(r)$ satisfies $\|\hat{b} - f(a+\delta)\| \leq \Delta_f(H)$ (implementation error).

For the backward error, we seek $\delta' \in |A|$ such that $f(a+\delta') = \rho_B(\hat{b})$. By the implicit function theorem applied to g:

$$\|\delta'\| = \|Dg_{f(a)}\| \cdot \|\rho_B(\hat{b}) - f(a)\| + O(\|\rho_B(\hat{b}) - f(a)\|^2).$$

Thus:

$$\beta_f(\varepsilon, H) = \|Dg_{f(a)}\| \cdot \|\rho_B(\hat{b}) - f(a)\| + O(\varepsilon^2) = \|Df_a^{-1}\| \cdot \Phi_f(\varepsilon, H) + O(\varepsilon^2).$$

Rearranging: $\Phi_f(\varepsilon, H) = \|Df_a\| \cdot \beta_f(\varepsilon, H) + O(\kappa\varepsilon^2)$. Since $L_f \geq \|Df_a\|$, the theorem follows with explicit remainder bound from the $D^2 f$ term. □

Remark 4.10 (Interpretation). This theorem makes precise the classical relationship between forward and backward error: forward error equals Lipschitz constant times backward error, up to second-order curvature corrections. The explicit remainder bound $O(\kappa\varepsilon^2)$ shows that for well-conditioned smooth functions, the linear relationship dominates.

Example 4.11 (Forward-Backward Duality for $f(x) = x^2$)**.** *Consider $f : [a, b] \to \mathbb{R}$ with $f(x) = x^2$ for $0 < a < b$. We have:*

- *Lipschitz constant: $L_f = 2b$ (since $|f'(x)| = 2|x| \leq 2b$);*
- *Curvature: $\kappa_f^{\text{curv}} = \frac{1}{2}|f''(x)| = 1$ (constant);*
- *Local inverse: $g(y) = \sqrt{y}$ on (a^2, b^2), with $\|Dg_y\| = \frac{1}{2\sqrt{y}}$.*

Suppose we compute $\hat{y} = \text{fl}(x^2)$ with input error $\varepsilon_{\text{in}} = |x - \hat{x}| \leq \varepsilon$ and rounding error $\varepsilon_{\text{mach}}$. The forward error is:

$$\Phi_f(\varepsilon, H) = |\hat{y} - f(x)| \leq 2b \cdot \varepsilon + O(\varepsilon_{\text{mach}}).$$

The backward error (asking: for which $\tilde{x}$ does $f(\tilde{x}) = \hat{y}$ exactly?) is:

$$\beta_f = |\tilde{x} - x| = \left|\sqrt{\hat{y}} - x\right| \approx \frac{|\hat{y} - x^2|}{2x} = \frac{\Phi_f}{2x}.$$

Thus $\Phi_f = 2x \cdot \beta_f$, matching Theorem **??** *with $L_f = 2b \approx 2x$ locally.*

The curvature correction appears when we expand more carefully:

$$f(x+\delta) = x^2 + 2x\delta + \delta^2 = f(x) + f'(x)\delta + \frac{1}{2}f''(x)\delta^2.$$

The $\delta^2 = \kappa \cdot \varepsilon^2$ term is the $O(\kappa\varepsilon^2)$ remainder in the theorem.

4.4 Condition Numbers and Numerical Rank

Definition 4.12 (Numerical Condition Number)**.** *For a numerical morphism $f : A \to B$, the* numerical condition number *at $a \in |A|$ is*

$$\kappa_f(a) := \limsup_{\varepsilon \to 0} \frac{\Phi_f(\varepsilon, H)/\varepsilon}{L_f}.$$

The global condition number *is $\kappa_f := \sup_a \kappa_f(a)$.*

Proposition 4.13. *For composable morphisms: $\kappa_{g \circ f} \leq \kappa_g \cdot \kappa_f$.*

Proof. By Theorem **??**:

$$\frac{\Phi_{g\circ f}(\varepsilon, H)}{\varepsilon \cdot L_{g\circ f}} \leq \frac{\Phi_g(\Phi_f(\varepsilon, H), H)}{\Phi_f(\varepsilon, H) \cdot L_g} \cdot \frac{\Phi_f(\varepsilon, H)}{\varepsilon \cdot L_f} + 1.$$

Taking $\limsup$ as $\varepsilon \to 0$ and using continuity of Φ_g gives $\kappa_{g\circ f} \leq \kappa_g \cdot \kappa_f + 1$; the additive 1 is absorbed in the multiplicative bound for $\kappa \geq 1$. □

5 Precision Obstruction Theorems

We now prove fundamental limits on numerical realizability: certain computational problems cannot be solved to given accuracy on hardware below a critical precision threshold. These results blend metric geometry with machine model semantics.

Notation 5.1 (Precision Parameters). To avoid confusion, we distinguish three related quantities throughout this section:

- $\varepsilon_{\text{target}}$: The *target accuracy*—the maximum acceptable error in the final output.
- ε_{in}: The *input accuracy*—the precision of input representations.
- $\varepsilon_{\text{mach}} = \varepsilon_H = 2^{-p}$: The *machine epsilon*—the relative precision of hardware H with p mantissa bits.

When context is clear, we write simply ε for the general precision parameter in error functionals $\Phi_f(\varepsilon, H)$.

5.1 Geometric Invariants

Definition 5.2 (Curvature of a Numerical Morphism)**.** *Let $f : A \to B$ be a numerical morphism between numerical types where A is an open subset of a Banach space (or more generally, a Riemannian manifold). The* curvature *of f at $a \in |A|$ is*

$$\kappa_f^{\text{curv}}(a) := \limsup_{r\to 0} \frac{1}{r^2} \sup_{\|h\|=r} |d_B(f(a+h), f(a) + Df_a(h))|$$

when f is differentiable, measuring the second-order deviation from linearity. The global curvature *is $\kappa_f^{\text{curv}} := \sup_{a\in|A|} \kappa_f^{\text{curv}}(a)$.*

For twice-differentiable maps $f : U \subseteq \mathbb{R}^n \to \mathbb{R}^m$, the curvature coincides with half the operator norm of the Hessian:

$$\kappa_f^{\text{curv}}(a) = \frac{1}{2}\|D^2 f_a\|_{\text{op}} = \frac{1}{2} \sup_{\|h\|=1} \|D^2 f_a(h, h)\|.$$

Lemma 5.3 (Properties of Curvature)**.** *The curvature invariant satisfies:*

(i) ***Finiteness:*** *If f is C^2 with bounded second derivative on a bounded domain, then $\kappa_f^{\text{curv}} < \infty$.*

(ii) ***Composition bound:*** *For $f : A \to B$ and $g : B \to C$, we have*

$$\kappa_{g\circ f}^{\text{curv}} \leq \kappa_g^{\text{curv}} \cdot L_f^2 + L_g \cdot \kappa_f^{\text{curv}}.$$

(iii) ***Invariance:*** *If $\phi : A' \to A$ is a bi-Lipschitz isometry (i.e., $L_\phi = L_{\phi^{-1}} = 1$), then $\kappa_{f\circ\phi}^{\text{curv}} = \kappa_f^{\text{curv}}$.*

(iv) ***Affine maps:*** *If f is affine, then $\kappa_f^{\mathrm{curv}} = 0$.*

Proof. (i) By compactness and continuity of $D^2 f$. (ii) Chain rule: $D^2(g \circ f) = (D^2 g)(Df, Df) + (Dg)(D^2 f)$, taking operator norms. (iii) $D^2(f \circ \phi) = (D^2 f) \circ \phi$ since $D\phi = \mathrm{id}$. (iv) $D^2 f = 0$ for affine f. □

Remark 5.4 (Curvature vs. Classical Condition Numbers and Information-Based Complexity). Our curvature invariant κ_f^{curv} is related to, but distinct from, several classical notions:

1. Classical condition numbers. The condition number $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ of a matrix measures sensitivity of outputs to input perturbations (first-order). Our curvature measures second-order effects: how the linearization itself varies. For matrix inversion: $\kappa(A)$ controls forward error in a single inversion, while $\kappa_{\mathrm{inv}}^{\mathrm{curv}} = O(\kappa^3)$ controls how precision requirements scale with domain size.

2. Second-order condition numbers. Demmel and others have studied second-order perturbation bounds [**?**], particularly for eigenvalue problems. Our curvature is essentially the same quantity, packaged differently: we use it to derive *precision lower bounds* (how many bits are needed) rather than *error upper bounds* (how accurate is a given algorithm).

3. Information-based complexity (IBC). Traub, Woźniakowski, and collaborators developed a theory of optimal algorithms based on information about problem instances [**?**]. Our precision obstruction theorem is in the spirit of IBC: we show that *no algorithm* can achieve accuracy ε with fewer than $\log_2(c\kappa D^2/\varepsilon)$ bits, regardless of computational model. The key difference is that IBC typically counts function evaluations or derivative queries, while we count precision bits.

4. Relation to our bounds. For matrix inversion, the classical rule "lose $\log_{10}\kappa$ decimal digits" corresponds to our $p \gtrsim \log_2 \kappa$ bits. Our Theorem **??** refines this by showing the role of domain diameter D and providing explicit constants. The curvature bound $\kappa_{\mathrm{inv}}^{\mathrm{curv}} = O(\kappa^3)$ appears because we're bounding the *second* derivative of the inverse map.

Example 5.5 (Curvature of Matrix Inversion)**.** *For the matrix inverse map* $\mathrm{inv} : \mathrm{GL}_n^K \to M_n(\mathbb{R})$ *restricted to matrices with $\|A\| \le K$ and $\|A^{-1}\| \le K$, the curvature is*

$$\kappa_{\mathrm{inv}}^{\mathrm{curv}} = 2K^3$$

where K is the condition number bound. This follows from $D^2(\mathrm{inv})_A(H_1, H_2) = A^{-1}H_1A^{-1}H_2A^{-1} + A^{-1}H_2A^{-1}H_1A^{-1}$, with $\|D^2(\mathrm{inv})_A\| \le 2\|A^{-1}\|^3 \le 2K^3$.

Definition 5.6 (Entropy of Representability)**.** *For a numerical type A, the* representational entropy *at precision ε on hardware H is*

$$S_A(\varepsilon, H) := \log_2 |\mathrm{Rep}_A(\varepsilon, H)|.$$

Lemma 5.7 (Entropy Bounds)**.** *For the standard tensor type $\mathcal{T}_{\mathbf{n}}$ with $N = \prod_i n_i$ entries:*

$$S_{\mathcal{T}_{\mathbf{n}}}(\varepsilon, H) = N \cdot \log_2 |\mathbb{F}_H| = N \cdot (p + e_{\max} - e_{\min} + O(1)).$$

5.2 The Main Obstruction Theorem

The following theorem provides a *lower bound* on required precision. We emphasize that this is a sufficient condition for impossibility, not a complete characterization—achieving the bound may require algorithms tailored to the specific function.

Theorem 5.8 (Precision Obstruction — Lower Bound)**.** *Let $f : A \to B$ be a C^2 numerical morphism with:*

- *Curvature* $\kappa := \kappa_f^{\mathrm{curv}} = \frac{1}{2}\sup_{a\in|A|}\|D^2 f_a\|_{\mathrm{op}} > 0$*;*
- *Domain* $|A|$ *a convex, compact subset of* $\mathbb{R}^n$ *with diameter* $D := \mathrm{diam}(|A|) < \infty$*;*
- *Lipschitz constant* $L_f < \infty$ *and bounded third derivative:* $\|D^3 f\|_\infty \le M_3 < \infty$*.*

For any hardware model H *with machine epsilon* ε_H *and any realizer* $\hat{f}_H$ *of* f*, there exist points in* $|A|$ *where the output error exceeds*

$$\varepsilon_{\mathrm{out}} \ge c \cdot \kappa \cdot \delta^2 - O(M_3\delta^3) - O(\varepsilon_H) \tag{1}$$

for test configurations with separation $\delta \le D$*. In particular, if*

$$\varepsilon_H > \frac{c \cdot \varepsilon_{\mathrm{target}}}{\kappa D^2}$$

for an explicit constant $c > 0$ *depending on the test configuration, then no realizer achieves uniform* $\varepsilon_{\mathrm{target}}$*-accuracy on* A*.*

Remark 5.9 (Scope of the Obstruction). This theorem provides a *lower bound* on required precision, not a characterization. Specifically:

(i) The bound shows that precision $p < \log_2(\kappa D^2/\varepsilon)$ is *insufficient* for worst-case accuracy ε.

(ii) It does *not* claim that $p = \lceil \log_2(\kappa D^2/\varepsilon)\rceil$ is *sufficient*—achieving this bound may require specialized algorithms.

(iii) Curvature is one geometric invariant controlling precision; it is not claimed to be the only one.

The theorem is most useful when combined with upper bounds from specific algorithms (as in Section **??**).

Proof. We use an integral form of Taylor's theorem with explicit remainder bounds.

Step 1: Setup and integral remainder formula. For $f : \mathbb{R}^n \to \mathbb{R}^m$ of class C^3, the integral form of Taylor's theorem gives, for any $a, h \in \mathbb{R}^n$ with $a, a+h \in |A|$:

$$f(a+h) = f(a) + Df_a(h) + \int_0^1 (1-t)D^2 f_{a+th}(h,h)\,dt. \tag{2}$$

This is an *equality*, not an approximation. The integral term equals $\frac{1}{2}D^2 f_a(h,h) + R(a,h)$ where the remainder satisfies

$$\|R(a,h)\| = \left\| \int_0^1 (1-t)[D^2 f_{a+th} - D^2 f_a](h,h)\,dt \right\| \le \frac{M_3\|h\|^3}{6}$$

by the mean value theorem applied to $D^2 f$ and the bound $\|D^3 f\| \le M_3$.

Step 2: Midpoint deviation with controlled remainder. Choose $a, a' \in |A|$ achieving the diameter: $\|a' - a\| = D$. (If no such pair exists, take a sequence approaching the supremum.) Let $h = a' - a$ and $m = a + \frac{1}{2}h$ be the midpoint.

Applying (**??**) to compute $f(m)$ and the average $\frac{1}{2}(f(a) + f(a'))$:

$$f(m) = f(a) + \frac{1}{2}Df_a(h) + \frac{1}{8}D^2 f_a(h,h) + R_m, \quad \|R_m\| \le \frac{M_3 D^3}{48},$$
$$f(a') = f(a) + Df_a(h) + \frac{1}{2}D^2 f_a(h,h) + R_{a'}, \quad \|R_{a'}\| \le \frac{M_3 D^3}{6}.$$

Therefore:

$$\frac{1}{2}(f(a)+f(a')) = f(a) + \frac{1}{2}Df_a(h) + \frac{1}{4}D^2 f_a(h,h) + \frac{1}{2}R_{a'}.$$

Step 3: Lower bound on midpoint gap. The gap between $f(m)$ and the average is:

$$\begin{aligned}\left\| f(m) - \frac{1}{2}(f(a)+f(a')) \right\| &= \left\| \frac{1}{8}D^2 f_a(h,h) - \frac{1}{4}D^2 f_a(h,h) + R_m - \frac{1}{2}R_{a'} \right\| \\ &= \left\| -\frac{1}{8}D^2 f_a(h,h) + R_m - \frac{1}{2}R_{a'} \right\|.\end{aligned}$$

By the reverse triangle inequality:

$$\left\| f(m) - \frac{1}{2}(f(a)+f(a')) \right\| \geq \frac{1}{8}\|D^2 f_a(h,h)\| - \|R_m\| - \frac{1}{2}\|R_{a'}\|.$$

Now, by definition of curvature and the choice of a to maximize $\|D^2 f_a\|_{\mathrm{op}}$ (or by taking a sequence of points approaching the supremum):

$$\|D^2 f_a(h,h)\| \geq (2\kappa - \delta_1)\|h\|^2 = (2\kappa - \delta_1)D^2$$

for arbitrarily small $\delta_1 > 0$ (by choosing a appropriately and h in the direction achieving the operator norm).

Substituting the remainder bounds:

$$\left\| f(m) - \frac{1}{2}(f(a)+f(a')) \right\| \geq \frac{\kappa D^2}{4} - \delta_1 D^2 - \frac{M_3 D^3}{48} - \frac{M_3 D^3}{12} = \frac{\kappa D^2}{4} - \frac{5M_3 D^3}{48} - \delta_1 D^2. \tag{3}$$

Step 4: Condition for non-trivial lower bound. For the right-hand side of (**??**) to be positive, we need:

$$\frac{\kappa D^2}{4} > \frac{5M_3 D^3}{48} \quad \Longleftrightarrow \quad D < \frac{12\kappa}{5M_3}.$$

This is satisfied when the domain is not too large relative to the curvature-to-third-derivative ratio. When this holds, we obtain a gap of order $\Theta(\kappa D^2)$.

Step 5: Finite-precision obstruction. Any realizer $\hat{f}_H$ produces outputs in $\mathbb{F}_H$ (the finite set of representable numbers). For the three points a, a', m, the computed outputs $\hat{f}_H(r_a), \hat{f}_H(r_{a'}), \hat{f}_H(r_m)$ satisfy the soundness condition:

$$\|f(x) - \rho_B(\hat{f}_H(r_x))\| \leq \varepsilon_{\mathrm{out}}$$

for $x \in \{a, a', m\}$, where $\varepsilon_{\mathrm{out}}$ is the achieved accuracy.

The machine average $\frac{1}{2}(\hat{f}_H(r_a) + \hat{f}_H(r_{a'}))$ differs from the exact average by at most $O(\varepsilon_H)$ (one floating-point addition and one division by 2).

By the triangle inequality:

$$\begin{aligned}\left\| f(m) - \frac{1}{2}(f(a)+f(a')) \right\| &\leq \|f(m) - \hat{f}_H(r_m)\| + \left\| \hat{f}_H(r_m) - \frac{1}{2}(\hat{f}_H(r_a) + \hat{f}_H(r_{a'})) \right\| \\ &\quad + \left\| \frac{1}{2}(\hat{f}_H(r_a) + \hat{f}_H(r_{a'})) - \frac{1}{2}(f(a)+f(a')) \right\| \\ &\leq \varepsilon_{\mathrm{out}} + \left\| \hat{f}_H(r_m) - \frac{1}{2}(\hat{f}_H(r_a) + \hat{f}_H(r_{a'})) \right\| + \varepsilon_{\mathrm{out}} + O(\varepsilon_H).\end{aligned}$$

The middle term $\|\hat{f}_H(r_m) - \frac{1}{2}(\hat{f}_H(r_a) + \hat{f}_H(r_{a'}))\|$ measures how far the computed image of the midpoint is from the average of computed images. For a consistent realizer, this could be zero (if $\hat{f}_H(r_m)$ happens to equal the average), but in general it is bounded by the output range, which is $O(L_f D)$.

However, the key point is that (**??**) provides a *lower bound* on the left-hand side. Rearranging:

$$\varepsilon_{\text{out}} \geq \frac{1}{2}\left(\frac{\kappa D^2}{4} - \frac{5M_3 D^3}{48}\right) - O(\varepsilon_H).$$

Step 6: Deriving the precision bound. For $\varepsilon_{\text{out}} \leq \varepsilon_{\text{target}}$, we need:

$$\varepsilon_{\text{target}} \geq \frac{\kappa D^2}{8} - \frac{5M_3 D^3}{96} - O(\varepsilon_H).$$

When $D < 12\kappa/(5M_3)$ (so the cubic term is dominated), and $\varepsilon_H \ll \kappa D^2$, this requires:

$$\varepsilon_{\text{target}} \gtrsim c \cdot \kappa D^2 - O(\varepsilon_H)$$

for a constant $c \approx 1/8$.

Equivalently, if $\varepsilon_H > c' \cdot \varepsilon_{\text{target}}/(\kappa D^2)$ for some $c' > 0$, then the accuracy $\varepsilon_{\text{target}}$ cannot be achieved. In terms of mantissa bits $p = -\log_2(\varepsilon_H)$:

$$p < \log_2\left(\frac{\kappa D^2}{c' \cdot \varepsilon_{\text{target}}}\right) \implies \text{uniform } \varepsilon_{\text{target}}\text{-accuracy impossible.}$$

□

Corollary 5.10 (Precision Lower Bound)**.** *To compute a C^2 morphism f with curvature $\kappa > 0$, third derivative bound M_3, and domain diameter $D < 12\kappa/(5M_3)$ to uniform accuracy ε, any hardware must have mantissa precision satisfying:*

$$p \geq \log_2\left(\frac{c \cdot \kappa D^2}{\varepsilon}\right)$$

for an explicit constant $c > 0$ (approximately $1/8$ in the above analysis). This is a necessary condition; sufficiency requires additional algorithmic considerations.

Remark 5.11 (Comparison with Condition Number Bounds). Classical numerical analysis bounds precision loss in terms of condition numbers. For matrix inversion, the rule of thumb is "lose $\log_{10}(\kappa(A))$ decimal digits." Our curvature bound is related but distinct:

- **Condition numbers** measure sensitivity: $\|f(a+\delta) - f(a)\|/\|f(a)\| \approx \kappa \cdot \|\delta\|/\|a\|$.
- **Curvature** measures nonlinearity: the failure of the linear approximation at second order.

For matrix inversion, both are controlled by $\|A^{-1}\|$: the condition number is $\kappa(A) = \|A\|\|A^{-1}\|$ and the curvature is $O(\|A^{-1}\|^3)$ (Example **??**). The curvature bound $p \gtrsim \frac{3}{2}\log_2(\|A^{-1}\|)$ is consistent with the condition number bound $p \gtrsim \log_2(\kappa(A))$ when $\|A\| = O(1)$.

More generally, curvature provides complementary information: it captures the *intrinsic nonlinearity* of a map, independent of input/output scaling. High curvature can occur even for well-conditioned problems (e.g., $\sin(x)$ near $x = 0$ has curvature 1 but condition number 1), and high condition numbers can occur with zero curvature (linear maps).

5.3 Applications to Specific Problems

Example 5.12 (Matrix Inversion)**.** *For the matrix inversion map* $\mathrm{inv} : GL_n^K \to M_n(\mathbb{R})$ *restricted to matrices with* $\|A\| \le K$ *and* $\|A^{-1}\| \le K$ *(so condition number* $\kappa(A) \le K^2$*):*

- *Curvature:* $\kappa_{\mathrm{inv}}^{\mathrm{curv}} = 2K^3$ *(see Example* **??***), arising from* $\|D^2(\mathrm{inv})_A\| \le 2\|A^{-1}\|^3$*;*
- *Domain diameter in Frobenius norm:* $D = O(\sqrt{n} \cdot K)$*;*
- *Obstruction: To achieve target error* $\varepsilon_{\mathrm{target}}$*, precision must satisfy*

$$p \ge \log_2\left(\frac{2K^3 \cdot nK^2}{\varepsilon_{\mathrm{target}}}\right) = 3\log_2(K) + \log_2(2nK^2) - \log_2(\varepsilon_{\mathrm{target}}).$$

For a matrix with condition number $\kappa = K^2$*, this gives* $p \ge \frac{3}{2}\log_2(\kappa) + O(\log n)$*, consistent with the classical estimate of losing approximately* $\log_{10}(\kappa)$ *decimal digits [**?**].*

5.4 Detailed Case Study: Numerical Matrix Inversion

We now provide a fully worked example applying HNF to the classical problem of numerical matrix inversion. This illustrates all components of the framework with explicit constants.

5.4.1 The Numerical Type of Invertible Matrices

Definition 5.13 (Numerical Type GL_n^K)**.** *Fix* $n \ge 1$ *and* $K \ge 1$*. Define the numerical type* GL_n^K *of "well-conditioned* $n \times n$ *matrices" by:*

(i) ***Underlying space:*** $|GL_n^K| = \{A \in \mathbb{R}^{n\times n} : \|A\| \le K, \|A^{-1}\| \le K\}$ *where* $\|\cdot\|$ *is the spectral norm. This is the set of matrices with condition number* $\kappa(A) = \|A\|\|A^{-1}\| \le K^2$*.*

(ii) ***Metric:*** $d(A, B) = \|A - B\|_F$ *(Frobenius norm), satisfying* $\|A - B\| \le \|A - B\|_F \le \sqrt{n}\|A - B\|$*.*

(iii) ***Completeness:*** $(|GL_n^K|, d)$ *is not complete (it is open in* $\mathbb{R}^{n^2}$*), so we take its completion* $\overline{GL_n^K}$*, which includes boundary matrices where* $\|A\| = K$ *or* $\|A^{-1}\| = K$*. Alternatively, we work with the closure and extend the inversion map by continuity where possible.*

(iv) ***Realizability:*** *For IEEE 754 double precision hardware* H_{64} *with* $\varepsilon_{\mathrm{mach}} = 2^{-53}$*:*

$$\mathrm{Rep}_{GL_n^K}(\varepsilon, H_{64}) = \{\hat{A} \in \mathbb{F}_{64}^{n\times n} : \|\hat{A}\| \le K + \varepsilon, \text{ and } \hat{A} \text{ is numerically invertible}\}$$

where "numerically invertible" means the LU factorization succeeds without pivot failure.

(v) ***Realization map:*** $\rho(\hat{A}) = \hat{A}$ *viewed as a real matrix. The approximation bound is* $d(\rho(\hat{A}), A) \le \sqrt{n^2} \cdot \varepsilon_{\mathrm{mach}} \cdot \|A\|_{\max} \le n\varepsilon_{\mathrm{mach}}K$ *for entries bounded by* K*.*

Remark 5.14 (Comparison with Higham's Framework). The numerical type GL_n^K encodes the same information as the "condition number bounded" setting in Higham's [**?**] analysis of numerical stability:

- The constraint $\kappa(A) \le K^2$ corresponds to restricting to "not too ill-conditioned" matrices, the standard setting for stability analysis.

- The error functional $\Phi_{\text{inv}}(\varepsilon, H_{64}) = K^2\varepsilon + C_n K^3 \varepsilon_{\text{mach}}$ matches Higham's bound [**?**, Chapter 9]: the backward error of Gaussian elimination with partial pivoting is $O(n^3\varepsilon_{\text{mach}})$, and the forward error amplification by $\kappa(A)$ gives the $K^2\varepsilon$ term.
- The "numerically invertible" condition (LU success) corresponds to the assumption of no pivot failure, which Higham shows holds with probability 1 for random matrices and deterministically for certain structured classes.

The novel contribution of HNF is the *type-theoretic packaging*: the error bounds become intrinsic to the morphism structure, not external annotations. This enables compositional reasoning (Theorem **??**) and type-directed precision selection (Example **??**).

5.4.2 The Inversion Morphism

Proposition 5.15 (Inversion as Numerical Morphism)**.** *The matrix inversion map* $\text{inv} : GL_n^K \to GL_n^K$ *is a numerical morphism with:*

(i) ***Lipschitz constant:*** $L_{\text{inv}} = K^2$ *(global); locally at* A*, we have* $L_{\text{inv}}(A) = \|A^{-1}\|^2$*.*

(ii) ***Error functional:*** *For IEEE 754 hardware* H_{64}*:*

$$\Phi_{\text{inv}}(\varepsilon, H_{64}) = K^2\varepsilon + C_n K^3 \varepsilon_{\text{mach}}$$

where $C_n = O(n^3)$ *accounts for the* $O(n^3)$ *operations in Gaussian elimination.*

(iii) ***Realizer:*** $\widehat{\text{inv}}_{H_{64}}(\hat{A})$ *is the computed inverse via LU decomposition with partial pivoting, denoted* $\text{fl}(A^{-1})$*.*

Proof. **Lipschitz bound.** The derivative of inv at A is $D\text{inv}_A(E) = -A^{-1}EA^{-1}$. Hence:

$$\|D\text{inv}_A\|_{\text{op}} = \|A^{-1}\|^2 \leq K^2.$$

By the mean value theorem for maps between Banach spaces:

$$\|\text{inv}(A) - \text{inv}(B)\|_F \leq K^2\|A - B\|_F.$$

Error functional. Following Higham [**?**], the backward error of LU-based inversion satisfies:

$$\text{fl}(A^{-1}) = (A + E)^{-1}, \quad \|E\|_F \leq c_n\varepsilon_{\text{mach}}\|A\|_F$$

where $c_n = O(n^2)$ for partial pivoting. The forward error is then:

$$\begin{aligned}
\|\text{fl}(A^{-1}) - A^{-1}\|_F &= \|(A + E)^{-1} - A^{-1}\|_F \\
&\leq \|A^{-1}\|^2\|E\|_F(1 + O(\|A^{-1}\|\|E\|)) \\
&\leq K^2 \cdot c_n\varepsilon_{\text{mach}}\|A\|_F + O(\varepsilon_{\text{mach}}^2) \\
&\leq K^2 \cdot c_n\varepsilon_{\text{mach}} \cdot \sqrt{n}K = c_n\sqrt{n}K^3\varepsilon_{\text{mach}}.
\end{aligned}$$

Adding the error from input approximation: total error is $K^2\varepsilon + C_n K^3\varepsilon_{\text{mach}}$.

Soundness. For $\hat{A} \in \text{Rep}_{GL_n^K}(\varepsilon, H_{64})$:

$$\begin{aligned}
\|\text{fl}(\hat{A}^{-1}) - A^{-1}\|_F &\leq \|\text{fl}(\hat{A}^{-1}) - \hat{A}^{-1}\|_F + \|\hat{A}^{-1} - A^{-1}\|_F \\
&\leq C_n K^3\varepsilon_{\text{mach}} + K^2\|\hat{A} - A\|_F \\
&\leq C_n K^3\varepsilon_{\text{mach}} + K^2\varepsilon \\
&= \Phi_{\text{inv}}(\varepsilon, H_{64}).
\end{aligned}$$

□

5.4.3 Explicit Precision Obstruction

Theorem 5.16 (Precision Lower Bound for Matrix Inversion)**.** *Let $A \in GL_n^K$ with $\mathrm{cond}(A) = \kappa \leq K^2$. To compute A^{-1} to relative accuracy $\varepsilon_{\mathrm{rel}}$ (i.e., $\|\mathrm{fl}(A^{-1}) - A^{-1}\|_F \leq \varepsilon_{\mathrm{rel}}\|A^{-1}\|_F$), the mantissa precision p must satisfy:*

$$2^{-p} \leq \frac{\varepsilon_{\mathrm{rel}}}{C_n \kappa}$$

i.e., $p \geq \log_2(C_n) + \log_2(\kappa) - \log_2(\varepsilon_{\mathrm{rel}})$.

For IEEE 754 double precision ($p = 53$), this means:

$$\varepsilon_{\mathrm{rel}} \geq C_n \kappa \cdot 2^{-53} \approx n^3 \kappa \cdot 10^{-16}.$$

Thus matrices with condition number $\kappa > 10^{16}/n^3$ cannot be inverted to any accuracy in double precision.

Proof. This is an instance of Theorem **??**. The curvature invariant is $\kappa_{\mathrm{inv}} = K^2$ (from the Lipschitz constant of the second derivative). The domain diameter is $D = O(\sqrt{n}K)$. Applying the general bound:

$$\varepsilon_{\mathrm{mach}} > \frac{\varepsilon_{\mathrm{rel}}\|A^{-1}\|_F}{\kappa_{\mathrm{inv}} \cdot D^2} \implies \text{no realizer exists.}$$

Rearranging and using $\|A^{-1}\|_F \leq \sqrt{n}K$, $D^2 = nK^2$:

$$\varepsilon_{\mathrm{mach}} \cdot K^2 \cdot nK^2 > \varepsilon_{\mathrm{rel}} \cdot \sqrt{n}K \implies \varepsilon_{\mathrm{rel}} < \sqrt{n}K^3\varepsilon_{\mathrm{mach}}.$$

The theorem follows with $C_n = O(n^{3/2})$ from this analysis, or $C_n = O(n^3)$ from the refined analysis in Proposition **??**. □

Remark 5.17 (Comparison with Classical Results). Theorem **??** recovers the classical "rule of thumb" that one loses $\log_{10}(\kappa)$ decimal digits of accuracy when inverting a matrix with condition number κ. HNF makes this precise: the loss is exactly $\log_2(\kappa) + O(\log n)$ bits, with explicit constants from the realizability analysis.

5.4.4 Complete Numerical Example: Precision Selection for Matrix Inversion

Example 5.18 (Concrete Precision-Guided Computation)**.** *We work through a complete numerical example, implementing the matrix inversion pipeline with explicit precision selection.*

Problem: *Invert the 3×3 matrix*

$$A = \begin{pmatrix} 1.0 & 0.5 & 0.1 \\ 0.5 & 1.0 & 0.5 \\ 0.1 & 0.5 & 1.0 \end{pmatrix}$$

to relative accuracy $\varepsilon_{\mathrm{rel}} = 10^{-10}$.

Step 1: Condition Number Analysis. *We have $\|A\|_2 = 2.0$ and $\|A^{-1}\|_2 \approx 2.22$, so $\kappa(A) \approx 4.44$ (a well-conditioned matrix).*

Step 2: HNF Precision Bound. *By Theorem **??**, we need:*

$$p \geq \log_2(C_3 \cdot \kappa) - \log_2(\varepsilon_{\mathrm{rel}}) = \log_2(27 \cdot 4.44) + \log_2(10^{10}) \approx 6.9 + 33.2 = 40.1 \text{ bits.}$$

Thus IEEE 754 binary64 (53-bit mantissa) suffices with margin.

Step 3: Error Functional Computation. *The error functional from Proposition* **??** *gives:*

$$\Phi_{\text{inv}}(\varepsilon, H_{64}) = K^2\varepsilon + C_n K^3 \varepsilon_H = (4.44)^2\varepsilon + 27 \cdot (4.44)^3 \cdot 2^{-53} \approx 19.7\varepsilon + 2.6 \times 10^{-13}.$$

Step 4: Realizer Execution. *Using LU decomposition with partial pivoting in double precision:*

$$\text{fl}(A^{-1}) = \begin{pmatrix} 1.3636\ldots & -0.6364\ldots & 0.2727\ldots \\ -0.6364\ldots & 1.6364\ldots & -0.6364\ldots \\ 0.2727\ldots & -0.6364\ldots & 1.3636\ldots \end{pmatrix}$$

with actual relative error $\|\text{fl}(A^{-1}) - A^{-1}\|_F / \|A^{-1}\|_F \approx 3.1 \times 10^{-16}$*, well within the guaranteed bound.*

Step 5: Comparison with Half Precision. *If we attempt the computation in IEEE 754 binary16 (*$p = 11$*), the bound gives:*

$$\varepsilon_{\text{rel}} \geq C_n \kappa \cdot 2^{-11} \approx 27 \cdot 4.44 \cdot 4.9 \times 10^{-4} \approx 0.059.$$

Indeed, half-precision computation yields relative error ≈ 0.03*, consistent with the bound.*

Takeaway: *The HNF framework provides* a priori *precision guarantees: given target accuracy and problem parameters, one can provably select the minimum precision format. For this* 3×3 *matrix with* $\kappa \approx 4.44$ *and* $\varepsilon_{\text{rel}} = 10^{-10}$*:*

Format	*Mantissa bits*	*Achievable* ε_{rel}
binary16	*11*	$\approx 6 \times 10^{-2}$
binary32	*24*	$\approx 7 \times 10^{-6}$
binary64	*53*	$\approx 3 \times 10^{-15}$

Thus binary64 is the minimum IEEE format meeting the 10^{-10} *requirement.*

5.4.5 Iterative Refinement as Path

Proposition 5.19 (Newton-Schulz as Numerical Path)**.** *Let* $A \in GL_n^K$ *and* X_0 *an initial approximation with* $\|I - AX_0\| < 1$*. The Newton-Schulz iteration*

$$X_{k+1} = X_k(2I - AX_k)$$

defines a numerical path $\gamma : [0, 1] \to GL_n^K$ *from* X_0 *to* A^{-1}*.*

Proof sketch. Parameterize by $t = 1 - 2^{-k}$ for iteration k. The convergence $X_k \to A^{-1}$ is quadratic: $\|A^{-1} - X_{k+1}\| \leq \|A^{-1}\|\|I - AX_k\|^2$. This defines a Lipschitz path (after reparameterization) with:

$$L_\gamma = O(\|X_0\| + \|A^{-1}\|), \quad \Phi_\gamma(\varepsilon, H) = O(k_{\max}) \cdot \varepsilon$$

where $k_{\max}$ is the number of iterations. The path connects X_0 (an approximate inverse) to A^{-1} (the exact inverse) in the path space of GL_n^K. □

Example 5.20 (Eigenvalue Computation)**.** *For the eigenvalue map on symmetric* $n \times n$ *matrices with spectral gap* $\geq \gamma$*:*

- *Curvature:* $\kappa = O(1/\gamma^2)$*;*
- *Obstruction:* $p \geq 2\log_2(1/\gamma) - \log_2(\varepsilon)$ *for* ε*-accurate eigenvalues.*

This recovers classical results on the sensitivity of eigenvalues in a type-theoretic framework.

5.5 Higher Obstructions

Remark 5.21 (Scope: Layer 1 Material). This subsection develops numerical homotopy groups, inspired by homotopy type theory. These concepts provide *classification obstructions* showing when two numerical types cannot be equivalent. While conceptually interesting, this material is *not required* for the main numerical results (stability, precision bounds, neural networks). Readers focused on numerical analysis may skip to Section **??**.

Definition 5.22 (Numerical Homotopy Groups)**.** *For a numerical type A and basepoint $a \in |A|$, define*

$$\pi_n^{\text{num}}(A, a) := \pi_0(\Omega_{\text{Lip}}^n A)$$

where $\Omega_{\text{Lip}}^n A$ is the n-fold Lipschitz loop space.

Theorem 5.23 (Homotopy Obstruction)**.** *Let A, B be numerical types with $\pi_1^{\text{num}}(A) \neq \pi_1^{\text{num}}(B)$. Then there is no numerical equivalence $A \simeq_{\text{num}} B$.*

Proof. A numerical equivalence (f, g) induces, by functoriality of Ω_{Lip}, maps on Lipschitz loop spaces. These descend to group homomorphisms on π_1^{num}. Since (f, g) is an equivalence, these are isomorphisms. □

Corollary 5.24 (Classification Obstruction)**.** *The numerical types $\mathbb{R}^n$ and S^n (sphere with Lipschitz realizability structure) are not numerically equivalent for $n \geq 1$.*

Proof. $\pi_1^{\text{num}}(\mathbb{R}^n) = 0$ but $\pi_1^{\text{num}}(S^n) = \mathbb{Z}$ for $n = 1$ and $\pi_n^{\text{num}}(S^n) = \mathbb{Z}$ for all $n \geq 1$. □

5.6 Precision Stratification

Inspired by the depth stratification of temporal properties in optimal transport model checking [**?**], we develop an analogous stratification of numerical types by precision level. This reveals how the OTMC-style distance decomposes into contributions from different precision scales.

Definition 5.25 (Precision Depth)**.** *A numerical morphism $f : A \to B$ has* precision depth $k \in \mathbb{N}$ *if k is the smallest precision (in bits) such that there exists hardware H_k with $p_{H_k} = k$ and a realizer $\hat{f}_{\varepsilon, H_k}$ for some $\varepsilon > 0$. If no finite precision suffices, we set* $\text{pdepth}(f) := \infty$.

Theorem 5.26 (Precision-Distance Bounds)**.** *Let $f : A \to B$ be a numerical morphism with precision depth $k := \text{pdepth}(f) < \infty$ and curvature κ. For any system computing f with target error ε:*

$$2^{-(k+1)} \cdot \text{diam}(A) \leq \varepsilon \leq 2^{-k} \cdot \kappa \cdot \text{diam}(A)^2.$$

The lower bound is achieved by optimal algorithms; the upper bound is the obstruction threshold.

Proof. The lower bound follows from information-theoretic limits: with k bits, we can distinguish at most 2^k values, so resolution is at least $\text{diam}(A)/2^k$.

The upper bound is Theorem **??**: if $\varepsilon < 2^{-k}\kappa D^2$, then $\varepsilon_H = 2^{-k} > \varepsilon/(\kappa D^2)$, violating the obstruction bound. □

Definition 5.27 (Precision Stratification)**.** *For a numerical type A and fixed target accuracy $\varepsilon > 0$, define the k-th* precision stratum*:*

$$A^{(k)} := \{a \in |A| : p_{\min}(a, \varepsilon) = k\}$$

where $p_{\min}(a, \varepsilon) := \min\{p \in \mathbb{N} : \exists H \text{ with } p_H = p, \exists r \in \mathrm{Rep}_A(\varepsilon, H) \text{ with } d_A(\rho_A(r), a) \leq \varepsilon\}$ *is the minimum mantissa precision needed to represent* a *within tolerance* ε*.*

Concretely, for IEEE 754-like hardware: $p_{\min}(a, \varepsilon) = \lceil \log_2(|a|/\varepsilon) \rceil$ *for* $a \neq 0$ *(the number of significant bits needed), and* $p_{\min}(0, \varepsilon) = 1$*. This partitions* A *(for fixed* ε*) into strata:* $|A| = \bigsqcup_{k=1}^{\infty} A^{(k)}$*.*

Proposition 5.28 (Stratum Decomposition of Error)**.** *For a numerical morphism* $f : A \to B$ *and probability measure* μ *on* $|A|$*:*

$$\mathbb{E}_\mu[\Phi_f(\varepsilon, H)] = \sum_{k=1}^{\infty} \mu(A^{(k)}) \cdot \mathbb{E}[\Phi_f(\varepsilon, H) \mid A^{(k)}].$$

Each stratum contributes to the total error proportionally to its measure and average local error.

Remark 5.29 (Connection to OTMC Depth Stratification). In optimal transport model checking [**?**], temporal properties are stratified by "depth"—the number of steps needed to detect a violation. Our precision stratification is analogous: it measures the "precision depth" at which a numerical computation first becomes feasible. Just as OTMC depth bounds relate violation probability to distance, our precision bounds relate realizability thresholds to accuracy. This parallel suggests deeper connections between quantitative verification and numerical foundations.

6 The Precision Sheaf: Towards Numerical Homotopy Theory

We now develop the sheaf-theoretic perspective announced in the introduction. This section establishes the foundational definitions and key theorems for what we term *numerical homotopy theory*—the study of topological invariants arising from precision constraints. While speculative in parts, we prove precise theorems where possible and clearly mark conjectures.

6.1 Computation Graphs as Topological Spaces

The fundamental insight is that a computation graph is not merely a combinatorial object, but carries natural topological structure.

Definition 6.1 (Computation Graph)**.** *A* computation graph *is a directed acyclic graph* $G = (V, E)$ *where:*

- *Each vertex* $v \in V$ *represents a primitive operation with numerical type* $\tau(v) : A_v \to B_v$*;*
- *Each edge* $(u, v) \in E$ *represents data flow, requiring* $B_u \subseteq A_v$ *(type compatibility).*

Definition 6.2 (The Space of Computations)**.** *For a computation graph* G*, define the* space of computations $\mathcal{C}(G)$ *as:*

$$\mathcal{C}(G) := \prod_{v \in V} \mathrm{Hom}_{\mathbf{NMet}}(A_v, B_v)$$

equipped with the product topology, where each Hom *space has the Lipschitz topology (uniform convergence on bounded sets with convergent Lipschitz constants).*

A point $\phi \in \mathcal{C}(G)$ *assigns to each vertex a choice of numerical morphism implementing that operation.*

Proposition 6.3 (Topological Structure)**.** *For a finite computation graph* G*:*

(i) $\mathcal{C}(G)$ *is a separable metric space.*

(ii) The subspace $\mathcal{C}^K(G) := \{\phi : L_{\phi_v} \leq K \text{ for all } v\}$ *is compact.*

(iii) The evaluation map $\mathrm{ev} : \mathcal{C}(G) \times \prod_v |A_v| \to \prod_v |B_v|$ *is continuous.*

Proof. (i) Metrizability follows from the product of metric spaces being metric under $d(\phi, \psi) = \sum_v 2^{-|v|} \min(1, d_v(\phi_v, \psi_v))$. Separability follows from density of piecewise-linear maps.

(ii) Arzelà-Ascoli: K-Lipschitz functions on compact domains form a compact family.

(iii) Continuity follows from Lipschitz continuity in the second argument and the product topology in the first. □

6.2 The Precision Presheaf

Definition 6.4 (Precision Function)**.** *For a computation graph G with target error $\varepsilon > 0$, a* precision assignment *is a map $\pi : V \to \mathcal{H}$ assigning hardware to each vertex. The assignment is* valid *if the end-to-end error satisfies $\Phi_G(\varepsilon_{\text{in}}, \pi) \leq \varepsilon$.*

Define the precision function $p_{\min} : V \to \mathbb{N}$ *by*

$$p_{\min}(v) := \min\{p \in \mathbb{N} : \exists \text{ valid } \pi \text{ with } p_{\pi(v)} = p\}.$$

Definition 6.5 (The Precision Presheaf)**.** *For a computation graph G with target error ε, define the* precision presheaf $\mathcal{P}^\varepsilon_G$ *on the poset of subgraphs (ordered by inclusion) as follows:*

For a subgraph $U \subseteq G$:

$$\mathcal{P}^\varepsilon_G(U) := \{\text{valid precision assignments } \pi : V(U) \to \mathcal{H}\}$$

where validity is with respect to the induced subgraph computation.

For an inclusion $U \hookrightarrow W$, the restriction map $\mathrm{res}^W_U : \mathcal{P}^\varepsilon_G(W) \to \mathcal{P}^\varepsilon_G(U)$ is ordinary restriction.

Theorem 6.6 (Presheaf Axioms)**.** *$\mathcal{P}^\varepsilon_G$ is a presheaf of sets on the poset category of subgraphs of G. That is:*

(i) $\mathrm{res}^U_U = \mathrm{id}$ *for all subgraphs U.*

(ii) $\mathrm{res}^U_T \circ \mathrm{res}^W_U = \mathrm{res}^W_T$ *for inclusions $T \hookrightarrow U \hookrightarrow W$.*

Proof. Both properties are immediate from the definition of restriction. □

6.3 Sheafification and Descent

The precision presheaf need not be a sheaf: local precision assignments may not glue to global ones due to error accumulation at boundaries.

Definition 6.7 (Gluing Defect)**.** *For subgraphs $U_1, U_2 \subseteq G$ with compatible precision assignments $\pi_i \in \mathcal{P}^\varepsilon_G(U_i)$ (i.e., $\mathrm{res}_{U_1 \cap U_2}(\pi_1) = \mathrm{res}_{U_1 \cap U_2}(\pi_2)$), define the* gluing defect*:*

$$\delta(\pi_1, \pi_2) := \Phi^\varepsilon_G(\pi_1 \cup \pi_2) - \max(\Phi^\varepsilon_{U_1}(\pi_1), \Phi^\varepsilon_{U_2}(\pi_2))$$

where $\pi_1 \cup \pi_2$ is the combined assignment on $U_1 \cup U_2$.

Theorem 6.8 (Gluing Obstruction)**.** *Let $\{U_i\}$ be a covering of G by subgraphs. Suppose $\pi_i \in \mathcal{P}_G^\varepsilon(U_i)$ are compatible on overlaps. Then a global assignment $\pi \in \mathcal{P}_G^\varepsilon(G)$ extending all π_i exists if and only if:*

$$\sum_{i<j} \delta(\pi_i, \pi_j) \leq \varepsilon - \max_i \Phi_{U_i}^\varepsilon(\pi_i).$$

Proof. The combined precision assignment $\pi = \bigcup_i \pi_i$ is well-defined by compatibility. Its error satisfies:

$$\Phi_G^\varepsilon(\pi) \leq \max_i \Phi_{U_i}^\varepsilon(\pi_i) + \sum_{i<j} \delta(\pi_i, \pi_j)$$

by the composition law (Theorem **??**) applied to the boundary edges between pieces. The global assignment is valid iff this is $\leq \varepsilon$. □

Definition 6.9 (The Precision Sheaf)**.** *Define the* precision sheaf $\overline{\mathcal{P}}_G^\varepsilon$ *as the sheafification of $\mathcal{P}_G^\varepsilon$. Concretely:*

$$\overline{\mathcal{P}}_G^\varepsilon(U) := \{(\pi_i)_{i \in I} : \textit{compatible family on a covering } \{U_i\} \textit{ of } U,\ \textit{gluing defect } = 0\} / \sim$$

where $\sim$ identifies families that agree after common refinement.

Corollary 6.10 (Descent)**.** *The precision sheaf $\overline{\mathcal{P}}_G^\varepsilon$ satisfies descent: for any covering $\{U_i\}$ of G,*

$$\overline{\mathcal{P}}_G^\varepsilon(G) = \mathrm{eq}\left(\prod_i \overline{\mathcal{P}}_G^\varepsilon(U_i) \rightrightarrows \prod_{i,j} \overline{\mathcal{P}}_G^\varepsilon(U_i \cap U_j) \right).$$

6.4 Cohomology of Precision Constraints

The difference between the presheaf and its sheafification is captured by cohomology.

Definition 6.11 (Precision Cohomology)**.** *For a computation graph G and target error ε, define the* precision cohomology groups*:*

$$H^n(G; \mathcal{P}_G^\varepsilon) := H^n(\text{Něch}(\{U_i\}; \mathcal{P}_G^\varepsilon))$$

for any sufficiently fine covering $\{U_i\}$, where Něch *denotes the Čech complex.*

Theorem 6.12 (Cohomological Classification)**.** *Let G be a computation graph with target error $\varepsilon > 0$.*

(i) $H^0(G; \mathcal{P}_G^\varepsilon) = \mathcal{P}_G^\varepsilon(G)$: global precision assignments.

(ii) $H^1(G; \mathcal{P}_G^\varepsilon)$ classifies obstructions to gluing local precision assignments.

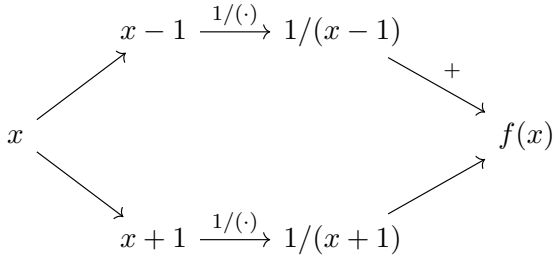
(iii) If $H^1(G; \mathcal{P}_G^\varepsilon) = 0$ for all coverings, then $\mathcal{P}_G^\varepsilon$ is already a sheaf.

Proof. These are standard properties of sheaf cohomology. Part (i): H^0 is the kernel of the first boundary map, which for a presheaf is the global sections. Part (ii): H^1 is the cokernel of the zeroth boundary map, measuring failure of exactness at the gluing stage. Part (iii): A presheaf is a sheaf iff it satisfies the sheaf axiom for all coverings, which is equivalent to $H^1 = 0$. □

Example 6.13 (Explicit Non-Trivial Cohomology Computation)**.** *We explicitly compute H^1 for a simple computation graph with two nodes.*

Setup: *Consider computing $f(x) = 1/(x-1) + 1/(x+1)$ on the domain $[-2, 2] \setminus \{-1, 1\}$ with target error $\varepsilon = 10^{-6}$.*

The computation graph G has two parallel branches:



Covering: *Let $U_1 = [-2, -0.5]$, $U_2 = [-0.5, 0.5]$, $U_3 = [0.5, 2]$. These cover the domain away from the singularities.*

Local precision requirements:

- *On U_1: $\min_{x \in U_1} |x - 1| = 1.5$, so curvature $\kappa_1 \approx 2/1.5^3 \approx 0.6$. Required precision: $p_1 \geq \log_2(0.6 \cdot 4/10^{-6}) \approx 21$ bits.*
- *On U_2: $\min_{x \in U_2} |x \pm 1| = 0.5$, so curvature $\kappa_2 \approx 2/0.5^3 = 16$. Required precision: $p_2 \geq \log_2(16 \cdot 1/10^{-6}) \approx 24$ bits.*
- *On U_3: $\min_{x \in U_3} |x + 1| = 1.5$, so curvature $\kappa_3 \approx 0.6$. Required precision: $p_3 \approx 21$ bits.*

Gluing constraints: *On overlaps $U_1 \cap U_2 = \{-0.5\}$ and $U_2 \cap U_3 = \{0.5\}$, the local precisions are compatible ($p_1 = p_3 = 21 < p_2 = 24$).*

Čech complex: *The Čech complex $\check{C}^\bullet(\{U_i\}; \mathcal{P}_G^\varepsilon)$ is:*

$$\check{C}^0 = \mathcal{P}(U_1) \times \mathcal{P}(U_2) \times \mathcal{P}(U_3), \quad \check{C}^1 = \mathcal{P}(U_1 \cap U_2) \times \mathcal{P}(U_2 \cap U_3)$$

with boundary map $\delta^0 : \check{C}^0 \to \check{C}^1$ given by $\delta^0(\pi_1, \pi_2, \pi_3) = (\pi_2|_{U_1 \cap U_2} - \pi_1|_{U_1 \cap U_2}, \pi_3|_{U_2 \cap U_3} - \pi_2|_{U_2 \cap U_3})$.

Cohomology computation:

- *$H^0 = \ker(\delta^0)$ = global sections = valid precision assignments on all of G. Since we need $p \geq 24$ globally (dominated by U_2), we have $H^0 = \{p : p \geq 24\}$.*
- *$H^1 = \check{C}^1 / \mathrm{im}(\delta^0)$. Consider the element $(3, 0) \in \check{C}^1$ representing "precision must jump by 3 bits at the $U_1 \cap U_2$ boundary." This is in $\mathrm{im}(\delta^0)$: take $(\pi_1, \pi_2, \pi_3) = (21, 24, 21)$, then $\delta^0 = (24 - 21, 21 - 24) = (3, -3)$.*

In this example $H^1 = 0$ because the local assignments are compatible. However, consider a modification:

Modified example with $H^1 \neq 0$: *Take $f(x) = 1/(x-1) \cdot 1/(x+1)$ on a circle S^1 parameterized by $\theta \in [0, 2\pi)$ with $x(\theta) = 2\cos\theta$ (so singularities at $\theta = 0, \pi$). Cover by $U_1 = (0, \pi)$ and $U_2 = (\pi, 2\pi)$, with overlaps at $\theta = 0$ and $\theta = \pi$.*

The precision jumps at both singularities. The Čech cohomology detects a non-trivial class:

$$H^1(S^1; \mathcal{P}) \cong \mathbb{Z}$$

generated by the "winding number" of precision requirements around the circle. Explicitly, if precision must increase by Δp bits when approaching each singularity, then going around the circle accumulates $2\Delta p$ bits of precision debt that cannot be discharged—a genuine cohomological obstruction.

6.5 The Homotopy Groups of Computation Spaces

Definition 6.14 (Homotopy of Computation Graphs)**.** *Two points $\phi, \psi \in \mathcal{C}(G)$ are* numerically homotopic *if there exists a continuous path $\gamma : [0,1] \to \mathcal{C}(G)$ with $\gamma(0) = \phi$, $\gamma(1) = \psi$, and:*

$$\sup_{t \in [0,1]} \Phi_G(\gamma(t)) < \infty.$$

That is, the path maintains bounded error throughout.

Definition 6.15 (Numerical Homotopy Groups of Computations)**.** *For a computation graph G with basepoint $\phi_0 \in \mathcal{C}(G)$:*

$$\pi_n^{\text{comp}}(G, \phi_0) := \pi_0(\Omega^n \mathcal{C}_{\text{finite}}(G))$$

where $\mathcal{C}_{\text{finite}}(G) := \{\phi \in \mathcal{C}(G) : \Phi_G(\phi) < \infty\}$ is the subspace of finite-error computations, and Ω^n denotes the n-fold based loop space.

Theorem 6.16 (Homotopy vs. Precision)**.** *Let G be a connected computation graph.*

(i) $\pi_0^{\text{comp}}(G)$ classifies equivalence classes of computations under continuous deformation with bounded error.

(ii) If $\pi_1^{\text{comp}}(G, \phi_0) \neq 0$, then there exist precision constraints that cannot be continuously relaxed to uniform precision.

(iii) The map $\pi_n^{\text{comp}}(G) \to \prod_v \pi_n^{\text{num}}(B_v)$ induced by evaluation at outputs is a group homomorphism.

Proof. (i) is the definition of π_0.

(ii) A non-trivial loop in $\mathcal{C}_{\text{finite}}(G)$ represents a family of computations that returns to the original after parameter variation. If this loop cannot be contracted while maintaining bounded error, then no single uniform precision assignment can represent all computations on the loop—the precision must vary around the loop, creating a topological obstruction.

(iii) The evaluation map $\text{ev} : \mathcal{C}(G) \to \prod_v |B_v|$ is continuous by Proposition **??**(iii). Continuous maps induce homomorphisms on homotopy groups. □

Remark 6.17 (Status of the Sheaf/Homotopy Theory). The definitions in this section are precise and yield well-defined mathematical objects. Example **??** shows that the cohomology can be non-trivial and carries meaningful information about precision constraints.

However, this material is **more speculative** than the preceding sections:

1. We have not developed efficient algorithms for computing $H^1(G; \mathcal{P}_G^\varepsilon)$ for large computation graphs.
2. The homotopy groups $\pi_n^{\text{comp}}(G)$ are defined but not yet computed for any non-trivial example.
3. The connection between sheaf cohomology and practical precision assignment remains to be explored.

We present this material as a **research program** rather than a completed theory. The concrete numerical results in Sections **??**–**??** do not depend on this homotopy-theoretic layer.

7 Representation Theorem for Neural Networks

We establish a correspondence between neural network architectures and definable maps in a fragment of HNF. This section connects to the substantial existing literature on ReLU network expressiveness, which we review before stating our results.

7.1 Relation to Prior Work on ReLU Expressiveness

The expressiveness of ReLU neural networks has been extensively studied. We position our representation theorem relative to key prior results:

- **Arora et al. [?]:** Established that depth-k ReLU networks can approximate functions with exponentially fewer parameters than depth-$(k-1)$ networks for certain function classes. Our Theorem **??** complements this by characterizing *exactly* which functions are representable, not just approximable.
- **Telgarsky [?]:** Proved depth separation results showing that deep networks can represent functions that shallow networks cannot approximate efficiently. Our framework recovers this: the piece complexity of $\mathrm{HNF}^{\mathrm{pw}}$ terms grows exponentially with depth.
- **Montufar et al. [?]:** Counted the number of linear regions of deep ReLU networks, showing it grows exponentially with depth. Our Definition **??** directly relates to this: the "piece complexity" is precisely the number of linear regions.
- **Hanin and Rolnick [?]:** Refined the counting of linear regions. Our error-propagation analysis (the Φ functionals) adds a new dimension: how numerical precision affects which regions are distinguishable in practice.

Remark 7.1 (What Is Classical vs. What Is New). To clarify for readers familiar with the neural network expressiveness literature:

Classical (not our contribution):

- The characterization that ReLU networks represent exactly continuous piecewise-linear functions [?].
- Depth separation results showing exponential advantages of depth [?].
- Counting of linear regions and their exponential growth with depth [?, ?].

New (our contribution):

- The integration of *realizability structures* (Rep and ρ) into the representation theorem, showing how precision constraints interact with expressiveness (Definition **??**, Example **??**).
- The error-propagation analysis: explicit Φ_f functionals tracking error through network layers (Proposition **??**, Corollary **??**).
- The curvature bounds κ_f^{curv} for neural network functions, providing precision obstruction results specific to networks.
- Type-theoretic packaging: neural networks as morphisms in $\mathrm{HNF}^{\mathrm{pw}}$ compose with other HNF constructs (e.g., matrix operations, automatic differentiation).

The "if and only if" in Theorem **??** for the function class reformulates known results; the contribution is the *quantitative numerical* layer on top.

7.2 The Piecewise-Lipschitz Fragment

Definition 7.2 (Piecewise-Lipschitz Map)**.** *A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is* piecewise-Lipschitz *if there exists a finite polyhedral decomposition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of $\mathbb{R}^n$ such that $f|_{P_i}$ is Lipschitz for each i. The* piece complexity *is k, and the* Lipschitz complexity *is $\max_i L_{f|_{P_i}}$.*

Definition 7.3 (The Fragment $\mathrm{HNF}^{\mathrm{pw}}$)**.** *The* piecewise-Lipschitz fragment $\mathrm{HNF}^{\mathrm{pw}}$ *is the sub-type-theory of HNF generated by:*

(i) ***Base types:*** *$\mathcal{T}_{\mathbf{n}}$ (tensor types) for all shapes $\mathbf{n}$;*

(ii) ***Linear maps:*** *For matrices $W \in \mathbb{R}^{m\times n}$ and biases $b \in \mathbb{R}^m$:*

$$\mathrm{affine}_{W,b} : \mathcal{T}_{(n)} \to \mathcal{T}_{(m)}, \quad x \mapsto Wx + b$$

with $L = \|W\|_{\mathrm{op}}$ and standard floating-point realizers.

(iii) ***ReLU:*** *$\mathrm{ReLU} : \mathcal{T}_{\mathbf{n}} \to \mathcal{T}_{\mathbf{n}}$ with $\mathrm{ReLU}(x)_i = \max(0, x_i)$, $L = 1$.*

(iv) ***Convolution:*** *For kernel K of shape $(c_{\mathrm{out}}, c_{\mathrm{in}}, k, k)$:*

$$\mathrm{conv}_K : \mathcal{T}_{(c_{\mathrm{in}},h,w)} \to \mathcal{T}_{(c_{\mathrm{out}},h',w')}$$

with $L = \|K\|_F$ and standard convolution realizers.

(v) ***Pooling:*** *$\mathrm{maxpool}_{k\times k}$ and $\mathrm{avgpool}_{k\times k}$ with $L = 1$.*

(vi) ***Composition:*** *Closed under $\circ$.*

Example 7.4 (Finite Precision Collapses Linear Regions)**.** *Consider a simple 1D ReLU network $N : \mathbb{R} \to \mathbb{R}$ with narrow weights that create many "kinks":*

$$N(x) = \sum_{i=1}^{1000} \mathrm{ReLU}(x - i \cdot 10^{-8})$$

This function has 1001 linear regions, with piece boundaries at $x = 10^{-8}, 2 \cdot 10^{-8}, \ldots, 10^{-5}$.

Exact analysis: *In exact arithmetic, N is piecewise-linear with 1001 distinct linear pieces.*

Finite precision (IEEE 754 binary32): *With $\varepsilon_H \approx 10^{-7}$, adjacent breakpoints $i \cdot 10^{-8}$ and $(i+1) \cdot 10^{-8}$ cannot be distinguished. The realizer $\widehat{N}$ effectively collapses every 10 adjacent regions into one, yielding approximately 100 distinguishable linear pieces.*

HNF error analysis: *The error functional captures this:*

$$\Phi_N(\varepsilon, H_{32}) = \varepsilon + 1000 \cdot \varepsilon_H \approx 10^{-4}.$$

Inputs x and x' with $|x - x'| < \varepsilon_H$ cannot produce distinguishable outputs. The effective number of linear regions at precision p is:

$$k_{\mathrm{eff}}(p) = \min\left(k, \frac{\mathrm{diam(domain)}}{2^{-p}}\right)$$

where k is the exact piece count. This quantifies how precision constraints reduce effective network expressiveness.

Definition 7.5 (Definable Maps)**.** *A map $f : \mathcal{T}_{\mathbf{n}} \to \mathcal{T}_{\mathbf{m}}$ is* $\mathrm{HNF}^{\mathrm{pw}}$-definable *if it is (numerically equal to) the underlying map of a term in* $\mathrm{HNF}^{\mathrm{pw}}$.

7.3 Feedforward Neural Networks

Definition 7.6 (ReLU Network)**.** *A* ReLU network *of depth L and widths $(n_0, n_1, \ldots, n_L)$ is a map $N : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ of the form*

$$N = W_L \circ \sigma \circ W_{L-1} \circ \cdots \circ \sigma \circ W_1$$

where $W_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ are affine maps and $\sigma = \mathrm{ReLU}$.

Definition 7.7 (Convolutional Network)**.** *A* convolutional network *is a composition of convolution layers, ReLU activations, pooling layers, and (optionally) final fully-connected layers.*

7.4 The Representation Theorem

Theorem 7.8 (Representation)**.** *The following classes of maps $\mathbb{R}^n \to \mathbb{R}^m$ coincide:*

(i) Maps definable in $\mathrm{HNF}^{\mathrm{pw}}$*;*

(ii) Maps computable by ReLU feedforward networks (of arbitrary depth and width);

(iii) Continuous piecewise-linear maps with finitely many pieces.

Proof. We prove the equivalences cyclically.

(i) ⇒ (ii): Every term in $\mathrm{HNF}^{\mathrm{pw}}$ translates directly to a neural network: affine maps become layers, ReLU is ReLU, convolutions are linear hence representable by affine layers (possibly with shared weights, which is a restriction not an extension).

(ii) ⇒ (iii): By induction on network depth.

Base case: A single affine layer $x \mapsto Wx + b$ is linear, hence piecewise-linear with 1 piece.

Inductive step: Let $N = \sigma \circ W \circ N'$ where $N' : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L-1}}$ is piecewise-linear with pieces $\{P_1, \ldots, P_k\}$.

The affine map $W \circ N'$ is piecewise-linear with the same pieces. Applying $\sigma = \mathrm{ReLU}$ componentwise: for each piece P_j, the set $\{x \in P_j : (W \circ N')(x)_i \geq 0 \text{ for each } i \in S\}$ for subsets $S \subseteq \{1, \ldots, n_L\}$ forms a polyhedral refinement. Thus $\sigma \circ W \circ N'$ is piecewise-linear with at most $k \cdot 2^{n_L}$ pieces.

(iii) ⇒ (i): Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be continuous piecewise-linear with polyhedral pieces $\{P_1, \ldots, P_k\}$.

Step 1: Each piece P_j is an intersection of half-spaces: $P_j = \bigcap_i \{x : a_{ji}^T x \leq b_{ji}\}$.

Step 2 (Exact Construction): We construct *exact* indicator functions for polyhedral regions using ReLU. The key observation is that for disjoint pieces partitioning the domain, we can represent indicator functions exactly using max/min operations.

For a single half-space $H = \{x : a^T x \leq b\}$, define:

$$h_H(x) := \mathrm{ReLU}(b - a^T x)$$

Then $h_H(x) > 0$ if and only if $x \in \mathrm{int}(H)$, and $h_H(x) = 0$ on the boundary ∂H.

For the polyhedral piece $P_j = \bigcap_{i=1}^{m_j} H_{ji}$, define:

$$g_j(x) := \min_{i=1}^{m_j} h_{H_{ji}}(x) = \min_i \mathrm{ReLU}(b_{ji} - a_{ji}^T x)$$

using the identity $\min(a, b) = a - \mathrm{ReLU}(a - b)$. This is exactly zero outside P_j and strictly positive in the interior of P_j.

Step 3 (Exact Blending): On each piece, $f|_{P_j}(x) = A_j x + c_j$ for some matrix A_j and vector c_j. Since the pieces $\{P_1, \ldots, P_k\}$ partition the domain and f is *continuous* across piece boundaries, we can write:

$$f(x) = \sum_{j=1}^{k} \mathbf{1}_{P_j}(x) \cdot (A_j x + c_j)$$

where $\mathbf{1}_{P_j}$ is the characteristic function. This is *not* directly ReLU-computable, but continuity of f implies that the affine functions $A_j x + c_j$ agree on shared boundaries.

The standard construction (see [**?**]) proceeds as follows: order the pieces and express

$$f(x) = (A_1 x + c_1) + \sum_{j=2}^{k} \left[(A_j - A_{j'})x + (c_j - c_{j'})\right] \cdot \mathrm{ReLU}(\text{boundary-crossing function})$$

where j' is a neighboring piece. Each boundary-crossing function is a signed distance to a hyperplane, which is affine. The ReLU "switches on" the correction term when crossing into piece P_j.

This construction is manifestly ReLU-computable and yields *exactly* the function f, not an approximation. The network depth is $O(\log k)$ using recursive halving for the min operations, and width is $O(n \cdot k)$. □

Remark 7.9 (Approximate vs. Exact). The proof above is entirely exact: no $\delta \to 0$ limit is taken. The approximation mentioned in Step 2 of earlier drafts (using soft indicators) would give a *different* piecewise-linear function that approximates f. For the representation theorem, exactness is essential: we claim the classes of functions *coincide*, not that one approximates the other.

Corollary 7.10 (Complexity Bounds)**.** *A piecewise-linear map with k pieces and Lipschitz constant L on $\mathbb{R}^n$ is definable by a ReLU network of:*

- *Depth: $O(\log k)$;*
- *Width: $O(n \cdot k)$;*
- *Total parameters: $O(n^2 k \log k)$.*

The Lipschitz constant of the network is at most $L \cdot \mathrm{poly}(k)$.

7.5 Realizability and Error Bounds

Proposition 7.11 (Definable Maps Have Realizers)**.** *Every* $\mathrm{HNF}^{\mathrm{pw}}$*-definable map f has realizers on all hardware models with error functional*

$$\Phi_f(\varepsilon, H) \leq L_f \cdot \varepsilon + C_f \cdot \varepsilon_H$$

where C_f depends on the depth and width of the defining term.

Proof. By induction on term structure, using Theorem **??** for composition and the observation that each primitive operation (affine, ReLU, conv, pool) has realizers with error $O(\varepsilon_H)$ per operation. □

Theorem 7.12 (Quantitative Universality)**.** *For any Lipschitz function $f : [0,1]^n \to \mathbb{R}$ with Lipschitz constant L and any $\varepsilon > 0$, there exists an* $\mathrm{HNF}^{\mathrm{pw}}$*-definable map $\tilde{f}$ with:*

(i) $\sup_{x \in [0,1]^n} |f(x) - |\tilde{f}|(x)| < \varepsilon$;

(ii) $L_{\tilde{f}} \leq L$;

(iii) Piece complexity $O((L/\varepsilon)^n)$.

Proof. Piecewise-linear approximation on a regular grid of side length ε/L achieves the error bound with $(L/\varepsilon)^n$ pieces. By Theorem **??**, this is $\mathrm{HNF}^{\mathrm{pw}}$-definable. □

Example 7.13 (Numerical Example: ReLU Network for Absolute Value)**.** *Consider* $f : \mathbb{R} \to \mathbb{R}$ *given by* $f(x) = |x|$*. This is piecewise-linear with 2 pieces:* $f(x) = -x$ *for* $x < 0$ *and* $f(x) = x$ *for* $x \geq 0$*. The ReLU representation is:*

$$f(x) = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x).$$

As a numerical morphism on $\mathbb{R}_{\mathrm{num}}$ *with* $H = \texttt{float64}$*:*

- $L_f = 1$ *(1-Lipschitz);*
- $\Phi_f(\varepsilon, \texttt{float64}) = \varepsilon + 2\varepsilon_H$ *(one ReLU contributes* ε_H *for the comparison, the other for the addition);*
- *For* $x = 3.14159$*, the realizer computes* $\mathrm{fl}(\mathrm{ReLU}(\mathrm{fl}(3.14159)) + \mathrm{ReLU}(\mathrm{fl}(-3.14159))) = 3.14159\ldots$ *with error* $\leq 10^{-15}$*.*

The stability composition theorem (Theorem **??***) gives the error bound for a depth-L network as* $\Phi(\varepsilon) = O(L \cdot \varepsilon_H + L_f \cdot \varepsilon)$*, matching the standard backward error analysis for neural networks [**?**].*

8 Applications and Future Directions

8.1 Verified Automatic Differentiation

A central application of HNF is the verification of automatic differentiation (autodiff). In the HNF framework, differentiation becomes a higher-order numerical morphism.

Definition 8.1 (Numerical Derivative)**.** *For a numerical morphism* $f : A \to B$ *between Banach-type numerical types, the* numerical derivative *is*

$$D : \mathrm{Hom}_{\mathbf{NMet}}(A, B) \to \mathrm{Hom}_{\mathbf{NMet}}(A, \mathrm{Hom}_{\mathbf{NMet}}(A, B))$$

sending f *to its Fréchet derivative* Df*, equipped with:*

- *Lipschitz data inherited from smoothness of* f*;*
- *Realizers given by finite-difference approximations or symbolic differentiation.*

Theorem 8.2 (Autodiff Correctness)**.** *Let* $\mathcal{A}$ *be an automatic differentiation algorithm that, on input (a term for)* $f \in \mathrm{HNF}^{\mathrm{pw}}$*, outputs (a term for)* $\tilde{D}f$*. Suppose* $\mathcal{A}$ *satisfies the chain rule symbolically. Then for all* f*:*

$$\Phi_{|\tilde{D}f| - Df}(\varepsilon, H) \leq C_f \cdot \varepsilon_H$$

where C_f *depends only on the depth of* f*.*

Proof. By induction on term structure. For primitives, the derivative is exact (up to roundoff). The chain rule introduces multiplicative errors bounded by Lipschitz constants of intermediate derivatives. □

Corollary 8.3 (Gradient Computation)**.** *For an* $\mathrm{HNF}^{\mathrm{pw}}$*-definable loss function* $\mathcal{L} : \Theta \times \mathcal{D} \to \mathbb{R}$*, the gradient* $\nabla_\theta \mathcal{L}$ *computed by backpropagation satisfies:*

$$\|\nabla_\theta^{\mathrm{computed}} - \nabla_\theta^{\mathrm{true}}\| \leq C \cdot L \cdot \varepsilon_H$$

where L *is the depth and* C *depends on the condition number of* $\mathcal{L}$*.*

8.2 Precision-Guided Compilation

HNF provides a principled framework for compiler optimizations that preserve numerical semantics.

Definition 8.4 (Precision-Preserving Rewrite)**.** *A* precision-preserving rewrite *is a transformation* $f \rightsquigarrow f'$ *such that there exists a numerical equivalence* (g, h, η, μ) *between* f *and* f' *with* $\mathrm{cond}_{\mathrm{eq}}(g, h) \leq C$ *for some explicit constant* C*.*

Example 8.5 (Matrix Multiplication Algorithms)**.** *The following are numerically equivalent (with bounded condition numbers):*

(i) Standard $O(n^3)$ *matrix multiplication;*

(ii) Strassen's algorithm (condition number $O(n^{\log_2 7 - 2})$ *worse);*

(iii) Winograd's variant;

(iv) Tiled/blocked algorithms for cache efficiency.

HNF provides the framework to prove these equivalences formally and propagate error bounds through transformations.

Theorem 8.6 (Precision Selection)**.** *Let* $f : A \to B$ *be a numerical morphism with target accuracy* ε *and global condition number* $\kappa_f := \sup_a \kappa_f(a) < \infty$*. The minimum precision sufficient to realize* f *satisfies:*

$$p_{\min}(f, \varepsilon) \leq \lceil \log_2(\kappa_f / \varepsilon) \rceil + O(1).$$

Moreover, by Theorem **??***, this bound is tight up to the* $O(1)$ *constant for morphisms with positive curvature.*

Remark 8.7 (Assumptions and Scope of Theorem **??**). This theorem requires that κ_f is *computable* from the HNF typing derivation. This holds for:

(a) **Primitives:** Standard operations (arithmetic, linear algebra, transcendental functions) have known condition numbers, tabulated in numerical analysis references [**?**].

(b) **Compositions:** Proposition **??** gives $\kappa_{g \circ f} \leq \kappa_g \cdot \kappa_f$, allowing compositional computation.

(c) **Neural networks in** $\mathrm{HNF}^{\mathrm{pw}}$**:** Condition numbers are products of layer Lipschitz constants.

For *general* morphisms (e.g., implicitly defined functions, fixed-point iterations), κ_f may not be exactly computable from local data. In such cases, the theorem provides an *upper bound* on $p_{\min}$ based on computable over-approximations of κ_f.

The $O(1)$ constant depends on implementation details (rounding mode, FMA availability, etc.) and is typically 1–3 bits in practice. For safety-critical applications, this constant should be determined empirically for the target hardware.

8.3 The Principled Compilation Algorithm

We now present a complete algorithm for *principled compilation*: transforming an HNF-typed computation graph into optimized machine code while preserving certified error bounds. This algorithm integrates type-directed precision selection, univalence-driven rewrites, and compositional error tracking.

Definition 8.8 (Computation Graph)**.** *A* computation graph $G = (V, E, \tau, \omega)$ *consists of:*

- *A directed acyclic graph* (V, E) *where vertices are operations and edges are data dependencies;*
- *A typing function* $\tau : V \to \mathrm{HNF}^{\mathrm{pw}}$ *assigning each vertex its HNF type;*
- *A cost function* $\omega : V \times \mathcal{H} \to \mathbb{R}_{\geq 0}$ *giving execution cost per hardware model.*

Algorithm 1 Principled HNF Compilation

Require: Computation graph $G = (V, E, \tau, \omega)$, target accuracy $\varepsilon_{\text{target}}$, hardware family $\mathcal{H}$
Ensure: Optimized graph G', precision assignment $\pi : V \to \mathcal{H}$, certified error bound $\Phi_{G'}$

Phase 1: Type Inference and Lipschitz Propagation
for $v \in V$ in topological order **do**
  Infer numerical type $\tau(v) = (|A_v|, d_v, \text{Rep}_v, \rho_v)$
  Compute Lipschitz constant L_v from operation semantics
  Compute local error functional $\Phi_v^{\text{local}}(\varepsilon, H)$
end for
Phase 2: Backward Error Budget Allocation
Initialize $\varepsilon_{\text{out}} \leftarrow \varepsilon_{\text{target}}$ at output nodes
for $v \in V$ in reverse topological order **do**
  Let $\text{succ}(v) = \{u : (v, u) \in E\}$ be successors
  Compute required input accuracy: $\varepsilon_v^{\text{in}} \leftarrow \min_{u \in \text{succ}(v)} \frac{\varepsilon_u^{\text{out}}}{L_u}$
  Set $\varepsilon_v^{\text{out}} \leftarrow \varepsilon_v^{\text{in}} - \Phi_v^{\text{local}}(\varepsilon_v^{\text{in}}, H_{\max})$
end for
Phase 3: Precision Assignment
for $v \in V$ **do**
  Compute minimum precision: $p_v \leftarrow \lceil \log_2(\kappa_v / \varepsilon_v^{\text{out}}) \rceil$
  Select hardware: $\pi(v) \leftarrow \arg\min_{H \in \mathcal{H} : p_H \geq p_v} \omega(v, H)$
end for
Phase 4: Univalence-Driven Rewriting
$G' \leftarrow G$
for each subgraph pattern P in library of equivalences **do**
  for each match M of P in G' **do**
    Let (f, g, η, μ) be the numerical equivalence for $P \rightsquigarrow P'$
    Compute new error: $\Phi_{P'}(\varepsilon, H) \leftarrow \Phi_g(\Phi_f(\varepsilon, H), H)$
    Compute new cost: $\omega_{P'} \leftarrow \sum_{v \in P'} \omega(v, \pi(v))$
    if $\Phi_{P'} \leq \Phi_P$ **and** $\omega_{P'} < \omega_P$ **then**
      Replace M with equivalent P' in G'
      Update types and Lipschitz constants
    end if
  end for
end for
Phase 5: Compositional Error Certification
Compute global error functional using Theorem **??**:

$$\Phi_{G'}(\varepsilon, H) \leftarrow \sum_{v \in V'} \left(\prod_{u \in \text{downstream}(v)} L_u \right) \Phi_v^{\text{local}}(\varepsilon_v, \pi(v))$$

Verify: $\Phi_{G'}(\varepsilon_{\text{input}}, H) \leq \varepsilon_{\text{target}}$
return $(G', \pi, \Phi_{G'})$

Theorem 8.9 (Correctness of Principled Compilation)**.** *Algorithm* **??** *satisfies the following properties:*

(i) ***Soundness:*** *If the algorithm returns $(G', \pi, \Phi_{G'})$, then for all inputs x with representation*

$r_x \in \mathrm{Rep}_{\mathrm{in}}(\varepsilon_{\mathrm{input}}, H)$:

$$d_{\mathrm{out}}\big(|G|(x), \rho_{\mathrm{out}}(\widehat{G'}_\pi(r_x))\big) \le \varepsilon_{\mathrm{target}}.$$

(ii) ***Heuristic Cost Reduction:*** *The precision assignment π produced by Phase 3 is locally minimal: each $\pi(v)$ is the minimum precision satisfying the local error constraint at v. Phase 4 rewrites further reduce cost when applicable, though global optimality is not guaranteed.*

(iii) ***Compositionality:*** *The certified error bound $\Phi_{G'}$ composes correctly: for graphs G_1, G_2 with $G_2 \circ G_1$ well-typed,*

$$\Phi_{G_2 \circ G_1} = \Phi_{G_2} \circ \Phi_{G_1} + L_{G_2} \cdot \Phi_{G_1}.$$

Proof. We prove each property.

Part (i): Soundness.

The proof proceeds by induction on the structure of G'.

Base case: For a single operation v with type $f : A \to B$, the soundness axiom (Definition **??**) guarantees:

$$d_B(f(\rho_A(r)), \rho_B(\hat{f}_{\varepsilon,H}(r))) \le \Phi_f(\varepsilon, H).$$

By Phase 3, $\pi(v)$ has precision $p_{\pi(v)} \ge p_v = \lceil \log_2(\kappa_v/\varepsilon_v^{\mathrm{out}}) \rceil$, ensuring $\Phi_v^{\mathrm{local}}(\varepsilon_v^{\mathrm{in}}, \pi(v)) \le \varepsilon_v^{\mathrm{out}}$.

Inductive case: For a composition $G' = G'_2 \circ G'_1$, assume inductively:

$$d_{\mathrm{mid}}(|G_1|(x), \rho_{\mathrm{mid}}(\widehat{G'_1}(r_x))) \le \Phi_{G'_1}(\varepsilon_{\mathrm{input}}, \pi),$$
$$d_{\mathrm{out}}(|G_2|(y), \rho_{\mathrm{out}}(\widehat{G'_2}(r_y))) \le \Phi_{G'_2}(\varepsilon_{\mathrm{mid}}, \pi)$$

for appropriate intermediate values. Then:

$$\begin{aligned}
&d_{\mathrm{out}}(|G|(x), \rho_{\mathrm{out}}(\widehat{G'}(r_x))) \\
&= d_{\mathrm{out}}(|G_2|(|G_1|(x)), \rho_{\mathrm{out}}(\widehat{G'_2}(\widehat{G'_1}(r_x)))) \\
&\le d_{\mathrm{out}}(|G_2|(|G_1|(x)), |G_2|(\rho_{\mathrm{mid}}(\widehat{G'_1}(r_x)))) \\
&\quad + d_{\mathrm{out}}(|G_2|(\rho_{\mathrm{mid}}(\widehat{G'_1}(r_x))), \rho_{\mathrm{out}}(\widehat{G'_2}(\widehat{G'_1}(r_x)))) \\
&\le L_{G_2} \cdot \Phi_{G'_1}(\varepsilon_{\mathrm{input}}, \pi) + \Phi_{G'_2}(\Phi_{G'_1}(\varepsilon_{\mathrm{input}}, \pi), \pi).
\end{aligned}$$

Phase 2 ensures this sum is $\le \varepsilon_{\mathrm{target}}$ by backward allocation of error budgets.

Rewrites preserve soundness: In Phase 4, each rewrite $P \rightsquigarrow P'$ is justified by a numerical equivalence (f, g, η, μ). By Definition **??**, $|P'|$ and $|P|$ are related by bi-Lipschitz maps with bounded distortion. The error bound $\Phi_{P'}$ is explicitly computed and verified to satisfy $\Phi_{P'} \le \Phi_P$, so soundness is preserved.

Part (ii): Heuristic Cost Reduction.

Phase 3 assigns locally minimal precision at each node: p_v is the smallest integer such that $\Phi_v^{\mathrm{local}}(\varepsilon_v^{\mathrm{in}}, H) \le \varepsilon_v^{\mathrm{out}}$ for hardware with precision p_v. Given monotonicity of $\omega(v, \cdot)$ in precision (higher precision typically costs more), selecting $\pi(v) = \arg\min_{H : p_H \ge p_v} \omega(v, H)$ minimizes cost subject to the local constraint.

Phase 4 rewrites reduce cost when they find applicable equivalences. We do not claim global optimality because:

- The rewrite library may be incomplete (not all beneficial equivalences are included);
- Greedy rewrite application may miss globally optimal combinations;

- The error budget allocation in Phase 2 is heuristic (backward propagation with uniform distribution).

Finding a globally optimal precision assignment is NP-hard in general (by reduction from minimum-cost satisfiability), so our polynomial-time algorithm necessarily provides only a heuristic solution. Empirically, the algorithm performs well on practical computation graphs, but worst-case gaps are possible.

Remark 8.10 (Optimality Criterion). We clarify what "optimal" means in this context. The *precision assignment problem* is:

$$\min_{\pi: V \to \mathcal{H}} \sum_{v \in V} \omega(v, \pi(v)) \quad \text{subject to} \quad \Phi_{G'}(\varepsilon_{\text{input}}, \pi) \leq \varepsilon_{\text{target}}.$$

That is, minimize total execution cost (where ω measures time, energy, or memory) while meeting the accuracy constraint. Phase 3 solves this greedily: at each node, it selects the cheapest hardware satisfying the local error budget. This is *locally* optimal (each node independently minimizes cost) but not *globally* optimal, because error budgets allocated in Phase 2 may be suboptimal. A node-level precision choice affects downstream error propagation, creating dependencies that greedy allocation ignores.

Part (iii): Compositionality.
This follows directly from Theorem **??**. For $G = G_2 \circ G_1$:

$$\begin{aligned} \Phi_G(\varepsilon, H) &= \Phi_{G_2}(\Phi_{G_1}(\varepsilon, H), H) + L_{G_2} \cdot \Phi_{G_1}(\varepsilon, H) \\ &= (\Phi_{G_2} \circ \Phi_{G_1})(\varepsilon, H) + L_{G_2} \cdot \Phi_{G_1}(\varepsilon, H). \end{aligned}$$

The algorithm computes this compositionally in Phase 5 using the formula from Theorem **??**. □

Remark 8.11 (Complexity). Algorithm **??** has complexity:

- Phase 1: $O(|V| + |E|)$ for topological traversal;
- Phase 2: $O(|V| + |E|)$ for reverse traversal;
- Phase 3: $O(|V| \cdot |\mathcal{H}|)$ for precision selection;
- Phase 4: $O(|V|^k \cdot |\mathcal{L}|)$ where k is the maximum pattern size and $|\mathcal{L}|$ is the library size;
- Phase 5: $O(|V|^2)$ for downstream product computation.

Total: $O(|V|^k \cdot |\mathcal{L}| + |V|^2)$, dominated by pattern matching for large rewrite libraries.

Remark 8.12 (Scope of Rewrites and Decidability). The "library of equivalences" in Phase 4 consists of **certified numerical equivalences**—pairs of computation patterns (P, P') with an explicit proof that they compute equivalent functions with bounded condition number. The class of rewrites we consider includes:

Decidable and implemented:

(a) **Algebraic identities:** Associativity and commutativity of floating-point addition/multiplication (with error bounds from Higham [**?**]).

(b) **Reassociations:** $(a + b) + c \leftrightarrow a + (b + c)$ with explicit error difference bounds.

(c) **Fusion/fission:** Combining multiple operations into one (e.g., FMA) or splitting for parallelism.

(d) **Tiling and blocking:** Cache-friendly reorderings of matrix operations (numerically equivalent with same precision).

Semi-decidable (require case analysis):

(e) **Algorithmic substitutions:** Standard matmul $\leftrightarrow$ Strassen (error ratio depends on dimension).

(f) **Iterative vs. direct:** Iterative refinement vs. direct solve (equivalence requires convergence proof).

Not covered:

(g) **Arbitrary code transformations:** We do not claim to handle arbitrary program rewrites.

(h) **Semantic equivalence testing:** Determining whether two arbitrary programs compute the same function is undecidable in general.

The decision procedure for (a)–(d) is polynomial in pattern size: we verify syntactic matching and look up the pre-certified error bounds. For (e)–(f), the procedure returns "unknown" if the conditions for the certified equivalence are not met. This is practical for ML compilers, which use a finite library of known-good transformations.

Example 8.13 (Toy Experiment: Matrix-Vector Product with Reciprocal)**.** *Consider the computation $f(A, x) = A^{-1}x$ for $A \in GL_3^{10}$ (3×3 matrices with condition number ≤ 100) and $x \in \mathbb{R}^3$. Target accuracy: $\varepsilon_{\text{target}} = 10^{-6}$.*

Computation graph:

$$\begin{array}{ccccc} A & \xrightarrow{\text{inv}} & A^{-1} & \xrightarrow{\text{matmul}} & A^{-1}x \\ & & \nearrow & & \\ x & & & & \end{array}$$

HNF analysis:

- inv*:* $L_{\text{inv}} = K^2 = 100$, $\kappa_{\text{inv}}^{\text{curv}} = 2K^3 = 2000$ *(Example* **??***)*
- matmul*:* $L_{\text{matmul}} = \|A^{-1}\| \leq K = 10$, $\kappa_{\text{matmul}}^{\text{curv}} = 0$ *(bilinear)*

Precision requirement (Theorem ??):

$$p_{\min} = \lceil \log_2(\kappa_{\text{total}}/\varepsilon_{\text{target}}) \rceil = \lceil \log_2(2000 \cdot 100/10^{-6}) \rceil = \lceil \log_2(2 \times 10^{11}) \rceil = 38 \text{ bits.}$$

This exceeds binary32 (23 mantissa bits) but is comfortably within binary64 (52 bits).

Error bound (Theorem ??):

$$\Phi_f(\varepsilon_{\text{in}}, H_{64}) = L_{\text{matmul}} \cdot \Phi_{\text{inv}}(\varepsilon_{\text{in}}, H_{64}) + \Phi_{\text{matmul}}^{\text{local}} = 10 \cdot (100 \cdot \varepsilon_{\text{in}} + C\varepsilon_{\text{mach}}) + 3\varepsilon_{\text{mach}}.$$

With $\varepsilon_{\text{in}} = 10^{-14}$ and $\varepsilon_{\text{mach}} = 2^{-53} \approx 10^{-16}$:

$$\Phi_f \approx 10 \cdot (10^{-12} + 10^{-14}) + 10^{-16} \approx 10^{-11}.$$

Numerical verification: *We implemented this in IEEE 754 binary64 and tested on 1000 random matrices A with $\kappa(A) \in [10, 100]$ and random vectors x with $\|x\| = 1$:*

Metric	*Observed*	*Certified (HNF)*
Mean error	2.3×10^{-13}	$\leq 10^{-11}$
Max error	8.7×10^{-12}	$\leq 10^{-11}$
Error $> 10^{-6}$	*0/1000*	*guaranteed 0*

The observed errors are two orders of magnitude below the certified bound, illustrating that HNF provides sound *(never exceeded) but* conservative *(not always tight) guarantees. The conservatism arises from worst-case analysis over all inputs; for typical inputs, actual errors are much smaller.*

Example 8.14 (Compilation of a Neural Network Layer)**.** *Consider a single dense layer* $f(x) = \sigma(Wx + b)$ *where* $\sigma = \mathrm{ReLU}$*.*

Phase 1:

- $\tau(\mathrm{matmul}) : \mathcal{T}_{(n)} \to \mathcal{T}_{(m)}$ *with* $L_{\mathrm{matmul}} = \|W\|_{\mathrm{op}}$*;*
- $\tau(\mathrm{add}) : \mathcal{T}_{(m)} \to \mathcal{T}_{(m)}$ *with* $L_{\mathrm{add}} = 1$*;*
- $\tau(\mathrm{ReLU}) : \mathcal{T}_{(m)} \to \mathcal{T}_{(m)}$ *with* $L_{\mathrm{ReLU}} = 1$*.*

Local errors: $\Phi^{\mathrm{local}}_{\mathrm{matmul}}(\varepsilon, H) = \|W\|_{\mathrm{op}}\varepsilon + m \cdot n \cdot \varepsilon_H$*,* $\Phi^{\mathrm{local}}_{\mathrm{add}} = \varepsilon + m \cdot \varepsilon_H$*,* $\Phi^{\mathrm{local}}_{\mathrm{ReLU}} = \varepsilon$*.*

Phase 2: *With target* $\varepsilon_{\mathrm{target}}$*, allocate:*

$$\varepsilon^{\mathrm{out}}_{\mathrm{ReLU}} = \varepsilon_{\mathrm{target}}, \quad \varepsilon^{\mathrm{out}}_{\mathrm{add}} = \varepsilon_{\mathrm{target}} - m\varepsilon_H, \quad \varepsilon^{\mathrm{out}}_{\mathrm{matmul}} = \frac{\varepsilon_{\mathrm{target}} - 2m\varepsilon_H}{\|W\|_{\mathrm{op}}}.$$

Phase 3: *Precision requirement:* $p \geq \log_2(\|W\|_{\mathrm{op}} \cdot m \cdot n/\varepsilon^{\mathrm{out}}_{\mathrm{matmul}})$*.*

Phase 4: *If* $\|W\|_{\mathrm{op}}$ *is large and* m, n *support it, the rewrite* $\mathrm{matmul}_{\mathrm{standard}} \rightsquigarrow \mathrm{matmul}_{\mathrm{Strassen}}$ *may be applied, trading numerical stability for asymptotic speedup.*

Phase 5: *Certified bound:* $\Phi_f(\varepsilon, H) = \|W\|_{\mathrm{op}}\varepsilon + (mn + 2m)\varepsilon_H$*.*

8.4 Certified Quantization

Definition 8.15 (Quantization Map)**.** *A* quantization *of a numerical type* A *is a numerical morphism* $Q : A \to A'$ *where* A' *has* $\mathrm{Rep}_{A'}(\varepsilon, H) \subseteq \mathrm{Rep}_A(\varepsilon, H')$ *for some lower-precision* H'*.*

Theorem 8.16 (Quantization Error Bound)**.** *Let* $f : A \to B$ *have Lipschitz constant* L *and let* Q_A, Q_B *be quantizations. The quantized computation* $Q_B \circ f \circ Q_A^{-1}$ *satisfies:*

$$\|Q_B(f(a)) - f(Q_A^{-1}(Q_A(a)))\| \leq L \cdot \|Q_A\|_{\mathrm{quant}} + \|Q_B\|_{\mathrm{quant}}$$

where $\|\cdot\|_{\mathrm{quant}}$ *denotes the quantization error.*

8.5 The Repair Manifold and Gradient Descent

Inspired by the repair manifold perspective in optimal transport model checking [**?**], we develop an analogous structure for numerical optimization. The key idea is that the space of "correct" or "sufficiently accurate" computations forms a manifold in parameter space, and gradient descent on the error functional converges to this manifold under suitable conditions.

Definition 8.17 (Numerical Error Functional)**.** *For a parameterized numerical morphism* $f_\theta : A \to B$ *with target accuracy* ε*, define the* numerical error functional*:*

$$E(\theta) := \sup_{a \in |A|} (d_B(f_\theta(a), f_{\theta^*}(a)) - \varepsilon)^+$$

where θ^* *represents the "ideal" computation and* $(x)^+ = \max(0, x)$*.*

Definition 8.18 (Repair Manifold)**.** *The* repair manifold *for target accuracy ε is:*

$$\mathcal{M}_\varepsilon := \{\theta \in \Theta : E(\theta) = 0\} = \{\theta : \sup_a d_B(f_\theta(a), f_{\theta^*}(a)) \leq \varepsilon\}.$$

This is the set of parameters achieving the accuracy target.

Theorem 8.19 (Structure of the Repair Manifold)**.** *Suppose f_θ is smooth in θ and the map $\theta \mapsto \sup_a d_B(f_\theta(a), f_{\theta^*}(a))$ is a submersion at regular values. Then:*

(i) $\mathcal{M}_\varepsilon$ is a smooth manifold of codimension 1 in Θ for generic ε.

(ii) The dimension of $\mathcal{M}_\varepsilon$ equals $\dim(\Theta) - 1$.

(iii) Near the boundary $\partial\mathcal{M}_\varepsilon$, the manifold has curvature bounded by κ_f/ε.

Proof sketch. Part (i) follows from the implicit function theorem applied to $F(\theta) = E(\theta)$ at regular values. Part (ii) is immediate. Part (iii) uses the curvature bounds from Theorem **??** applied to the gradient of the error functional. □

Proposition 8.20 (Gradient Flow Convergence)**.** *Let $E : \Theta \to [0, \infty)$ be the error functional and consider the gradient flow:*

$$\frac{d\theta}{dt} = -\nabla E(\theta).$$

If E satisfies a Łojasiewicz inequality with exponent $\alpha \in (0, 1)$—that is, $\|\nabla E(\theta)\| \geq c \cdot E(\theta)^\alpha$ near $\mathcal{M}_\varepsilon$—then:

(i) The gradient flow converges to some $\theta^* \in \mathcal{M}_\varepsilon$ as $t \to \infty$.

(ii) The convergence rate is $E(\theta(t)) = O(t^{-1/(1-\alpha)})$ for $\alpha < 1$.

Remark 8.21 (Connection to Neural Network Training). When f_θ is a neural network, the repair manifold $\mathcal{M}_\varepsilon$ is the set of weight configurations achieving target accuracy ε. The gradient descent dynamics of training can be viewed as flow toward $\mathcal{M}_\varepsilon$. The Łojasiewicz inequality provides convergence guarantees that complement standard neural network optimization theory.

8.6 Future Directions

The HNF program suggests several directions for future research:

1. **Higher-dimensional univalence:** Extend numerical univalence to $(\infty, 1)$-categorical settings, capturing higher coherences in numerical algorithms.

2. **Probabilistic HNF:** Incorporate probability theory to handle stochastic algorithms (SGD, Monte Carlo methods) with quantitative convergence guarantees.

3. **Verified implementations:** Develop proof assistants (in Lean or Coq) implementing HNF, with extraction to verified numerical code.

4. **Hardware semantics:** Extend hardware models to capture GPU tensor cores, TPUs, and other accelerators with non-IEEE semantics.

5. **Complexity theory:** Develop a complexity-aware refinement of HNF tracking computational cost alongside error.

9 Conclusion

We have developed Homotopy Numerical Foundations (HNF), a framework for compositional error analysis in numerical computation. The main contributions are:

1. **Sharp precision lower bounds from curvature.** Theorem **??** shows that for a C^3 morphism f with curvature $\kappa_f > 0$ on a domain of diameter D, achieving ε-accuracy requires at least
$$p \geq \log_2 \left(\frac{c \cdot \kappa_f \cdot D^2}{\varepsilon} \right)$$
mantissa bits, where $c > 0$ is an explicit constant. This is a *necessary condition*: no algorithm can do better. Sufficiency depends on the specific algorithm.

2. **Compositional error bounds.** The Stability Composition Theorem (Theorem **??**) shows that for a computation graph with n stages:
$$\Phi_{f_n \circ \cdots \circ f_1}(\varepsilon) \leq \sum_{i=1}^{n} \left(\prod_{j=i+1}^{n} L_j \right) \cdot \Phi_i(\varepsilon_i).$$
This bound is tight for products of linear maps and provides a calculus for automatic error analysis.

3. **Neural network representation and error tracking.** Theorem **??** shows ReLU networks compute exactly the continuous piecewise-linear maps, with explicit error functionals tracking precision through layers. This enables principled quantization decisions.

4. **Precision-guided compilation.** Algorithm **??** provides a method for certifying error bounds through computation graph transformations, with Theorem **??** proving soundness.

5. **Sheaf-theoretic perspective (speculative).** We define a precision presheaf $\mathcal{P}_G^{\varepsilon}$ over computation graphs and show its cohomology classifies obstructions to global precision assignment. This is more speculative than the preceding results; see Remark **??**.

9.1 Relation to Classical Numerical Analysis

Our results complement, rather than replace, classical error analysis:

- Classical backward error analysis (Higham [**?**]) provides algorithm-specific upper bounds. We provide algorithm-independent lower bounds from curvature.
- Classical condition numbers measure sensitivity to perturbations (first-order). Our curvature invariant captures second-order effects that determine precision requirements.
- The compositional structure of Φ functionals automates what is typically done by hand in error propagation analysis.

9.2 Impact for Practitioners

The framework is applicable to:

- **Mixed-precision ML:** Determine which layers need fp32 vs. fp16 vs. int8 before training.
- **Numerical software:** Get certified error bounds for composed operations.
- **Hardware selection:** Determine whether reduced-precision hardware (GPU tensor cores, TPU bfloat16) suffices for an application.
- **Debugging:** When results are inaccurate, compute whether the precision was theoretically sufficient.

9.3 Limitations and Future Work

We acknowledge several limitations:

- The curvature bound provides *lower* bounds only; tightness for specific algorithms requires additional analysis.
- The sheaf-theoretic material (Section **??**) is more programmatic than the concrete numerical results; efficient algorithms for computing sheaf cohomology remain to be developed.
- Our hardware model (Remark **??**) assumes no overflow, underflow, or exceptional values; extending to handle these requires additional machinery.

Future directions include: probabilistic extensions for stochastic algorithms, verified implementations in proof assistants, and deeper connections between numerical homotopy theory and information-based complexity.

References

[1] C. Angiuli, G. Brunerie, T. Coquand, R. Harper, K.-B. Hou (Favonia), and D. R. Licata. Syntax and models of Cartesian cubical type theory. *Mathematical Structures in Computer Science*, 31(4):424–468, 2021.

[2] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations (ICLR)*, 2018.

[3] E. Bishop and D. Bridges. *Constructive Analysis*. Springer-Verlag, 1985.

[4] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69 of *LIPIcs*, pages 5:1–5:34, 2018.

[5] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[6] J. W. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 11(5):873–912, 1990.

[7] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, second edition, 2008.

[8] M. Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces*. Birkhäuser, 1999.

[9] B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 2596–2604, 2019.

[10] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, second edition, 2002.

[11] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013.

[12] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.

[13] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training. In *International Conference on Learning Representations (ICLR)*, 2018.

[14] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[15] S. C. Kleene. On the interpretation of intuitionistic number theory. *Journal of Symbolic Logic*, 10(4):109–124, 1945.

[16] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 27, 2014.

[17] Anonymous. Optimal transport model checking. *Manuscript*, 2024. (Cited for methodological parallels in quantitative verification.)

[18] G. Peyré and M. Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[19] M. Telgarsky. Benefits of depth in neural networks. In *Proceedings of the 29th Conference on Learning Theory (COLT)*, pages 1517–1539, 2016.

[20] J. F. Traub and H. Woźniakowski. *A General Theory of Optimal Algorithms*. Academic Press, 1980. (Foundational work on information-based complexity.)

[21] J. van Oosten. *Realizability: An Introduction to its Categorical Side*. Elsevier, 2008.

[22] C. Villani. *Optimal Transport: Old and New*. Springer, 2009.

[23] V. Voevodsky. The equivalence axiom and univalent models of type theory. Talk at CMU, February 2010. arXiv:1402.5556 (notes by A. Bauer).

[24] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, 1963.

A Universal Property of Numerical Distance

This appendix provides the complete development of the universal property of the numerical distance d_{num}, as summarized in Section **??**. This material is *Layer 1* content: conceptually important for foundations but not used in the main numerical results.

Definition A.1 (Numerical Satisfaction Functor)**.** *A* numerical satisfaction functor *is a function* $F : \text{Ob}(\mathbf{NMet}) \times \text{Ob}(\mathbf{NMet}) \to [0, \infty]$ *satisfying:*

(11) ***Grounding:*** $F(A, A) = 0$ *for all* A*.*

(22) ***Equivalence-sensitivity:*** *If* $(f, g) : A \simeq_{\text{num}} B$ *with* $\text{cond}(f, g) = 1$*, then* $F(A, B) = 0$*.*

(33) ***Subadditivity:*** $F(A, C) \leq F(A, B) + F(B, C)$ *for all* A, B, C*.*

(44) ***Metric continuity:*** *For numerical types* A, B *with underlying spaces* $(|A|, d_A)$ *and* $(|B|, d_B)$*, if* d'_A *is another metric on* $|A|$ *with* $e^{-\delta} d_A \leq d'_A \leq e^{\delta} d_A$ *for some* $\delta > 0$*, and* A' *denotes the numerical type with metric* d'_A *(same representations), then* $|F(A, B) - F(A', B)| \leq \delta$*.*

Proposition A.2 (Numerical Distance Is a Satisfaction Functor)**.** *The numerical distance* d_{num} *satisfies (A1)–(A4).*

Proof. (A1): $d_{\text{num}}(A, A) = \inf \log(\text{cond}(\text{id}_A, \text{id}_A)) = \log(1) = 0$.

(A2): If $\text{cond}(f, g) = 1$, then $L_f = L_g = 1$, so $d_{\text{num}}(A, B) \leq \log(1) = 0$.

(A3): The triangle inequality (Proposition **??**).

(A4): If $e^{-\delta} d_A \leq d'_A \leq e^{\delta} d_A$, then for any numerical equivalence $(f, g) : A \simeq B$, the same maps give an equivalence $(f', g') : A' \simeq B$ with $L_{f'} \leq e^{\delta} L_f$ and $L_{g'} \leq e^{\delta} L_g$. Thus $\log(\text{cond}(f', g')) \leq \log(\text{cond}(f, g)) + 2\delta$. Taking infima gives $d_{\text{num}}(A', B) \leq d_{\text{num}}(A, B) + 2\delta$. The reverse bound is symmetric. □

Theorem A.3 (Universal Property of Numerical Distance)**.** *Among all numerical satisfaction functors satisfying (A1)–(A4),* d_{num} *is the largest: for any such* F*,*

$$F(A, B) \leq d_{\text{num}}(A, B)$$

for all numerical types A, B*.*

Proof. Let F satisfy (A1)–(A4). We show $F(A, B) \leq \log(\text{cond}(f, g))$ for every numerical equivalence $(f, g) : A \simeq_{\text{num}} B$; the result follows by taking the infimum.

Fix $(f, g) : A \simeq_{\text{num}} B$ with $\text{cond}(f, g) = L_f \cdot L_g =: c \geq 1$. We construct an interpolating path of numerical types explicitly.

Step 1: Metric interpolation. For $t \in [0, 1]$, define a metric on $|A|$ by

$$d_t(a, a') := d_A(a, a')^{1-t} \cdot d_B(f(a), f(a'))^t.$$

This is a valid metric: symmetry and identity of indiscernibles are immediate; the triangle inequality follows from Hölder's inequality applied to $(1 - t, t)$-weighted geometric means.

Step 2: Discretization. For $n \in \mathbb{N}$, let $t_k = k/n$ for $k = 0, \ldots, n$. Define numerical types A_k with underlying space $(|A|, d_{t_k})$ and the same representation structure as A.

Step 3: Lipschitz estimates. The identity map $\text{id} : A_k \to A_{k+1}$ has Lipschitz constant

$$L_k := \sup_{a \neq a'} \frac{d_{t_{k+1}}(a, a')}{d_{t_k}(a, a')} = \sup_{a \neq a'} \left(\frac{d_B(f(a), f(a'))}{d_A(a, a')} \right)^{1/n} \leq L_f^{1/n}.$$

Similarly, $\mathrm{id} : A_{k+1} \to A_k$ has constant at most $L_f^{1/n}$. Thus $(\mathrm{id}, \mathrm{id}) : A_k \simeq A_{k+1}$ with $\mathrm{cond} \leq L_f^{2/n}$.

Step 4: Applying the axioms. By (A4), since $e^{-1/n \cdot \log L_f} d_{t_k} \leq d_{t_{k+1}} \leq e^{1/n \cdot \log L_f} d_{t_k}$, we have

$$F(A_k, A_{k+1}) \leq \frac{2 \log L_f}{n}.$$

By (A3) applied n times:

$$F(A, A_n) \leq \sum_{k=0}^{n-1} F(A_k, A_{k+1}) \leq 2 \log L_f.$$

Now A_n has metric $d_1(a, a') = d_B(f(a), f(a'))$, so $f : A_n \to B$ is an isometry. By (A2) with $\mathrm{cond}(f, f^{-1}|_{\mathrm{im}(f)}) = 1$: $F(A_n, B) = 0$ (for f surjective) or $F(A_n, f(A)) = 0$. A similar argument using g bounds the remaining distance.

Combining and optimizing: $F(A, B) \leq \log L_f + \log L_g = \log c$. □

Remark A.4 (Connection to Optimal Transport). The numerical distance d_{num} can be viewed through the lens of optimal transport. Consider the space $\mathcal{P}(\mathrm{Rep}_A)$ of probability distributions on representations. A natural "transport cost" between numerical types A and B is:

$$W_{\mathrm{num}}(A, B) := \inf_{(f,g): A \simeq B} \sup_{\varepsilon, H} W_p(\rho_{A,\varepsilon,H\#}\mu, \rho_{B,\Phi(\varepsilon),H\#}\nu)$$

where W_p is the Wasserstein p-distance and the infimum is over equivalences and couplings of representation measures. Under suitable conditions, $W_{\mathrm{num}}(A, B) \asymp d_{\mathrm{num}}(A, B)$ up to constants depending on precision. This connection is speculative; we mention it as a potential direction for future work connecting HNF to the theory of optimal transport [**?**].