Why Deployed LLMs Are Not Mathematical Models: A Rigorous Internal Critique

Oliviana Ursuleac

February 2026

Abstract

Large language models (LLMs) are routinely presented as "mathematical models": parameterized functions $f_\theta : X \to Y$ composed of linear algebra, attention mechanisms, and softmax probabilities. This paper proves deployed LLMs fail *structural mathematical model* criteria based strictly on their internal mathematics. Four pillars of failure are established: (1) domain mismatch between formal $\mathbb{R}^n$ and discrete/configurational reality; (2) violation of continuity requirements for observable behavior; (3) parameter non-identifiability beyond measure-zero sets; (4) absence of applicable quantified approximation theory. Thus LLMs are *pseudo-mathematical artifacts*, not genuine mathematical models in the structural sense.

1 Introduction

The machine learning literature frames deep neural networks as mathematical models: compositions of affine transformations, nonlinearities, and softmax yielding $f_\theta : X \to Y$, trained via gradient descent on cross-entropy loss. This paper evaluates deployed LLMs *strictly through their internal mathematics* (tokenization, embeddings, attention, softmax, autoregressive decoding, gradient-based training) and demonstrates systematic structural failure.

While idealized network architectures satisfy formal mathematical requirements, actual deployed systems violate core structural invariants: fixed domains, appropriate continuity, meaningful parameter identifiability, and controlled approximation. The observable behavior $\mathcal{A}$ emerges from a stochastic, configuration-dependent procedure fundamentally distinct from any single mathematical function f_θ.

2 Structural Mathematical Model: Definition

Definition 1 (Structural Mathematical Model)**.** *A computational system $\mathcal{S}$ is a* structural mathematical model *if there exists an ideal mathematical object*

$$\mathcal{M} = (X, \Theta, Y, F)$$

satisfying:

1. ***Fixed structured spaces****: X, Θ, Y are fixed Polish spaces (complete separable metric spaces) independent of system configuration,*
2. ***Regularity****: $F : X \times \Theta \to Y$ is continuous (or at minimum, measurable with controlled discontinuities),*

3. ***Identifiability***: *The map $\theta \mapsto F(\cdot, \theta)$ is injective on a set of full measure in Θ,*

4. ***Faithful approximation***: *$\mathcal{S}$ approximates $\mathcal{M}$ with quantified error: there exists a refinement parameter h and constant C such that*

$$\sup_{x \in X} \|\mathcal{S}_h(x, \theta) - F(x, \theta)\|_Y \leq C h^p$$

for some $p > 0$ as $h \to 0$.

Remark 1. *This definition captures the essential structure of mathematical modeling in scientific computing. The key requirements are:*

- Fixed domains*: The spaces X, Θ, Y do not change with implementation details*
- Regularity*: Solutions behave predictably under perturbations*
- Identifiability*: Different parameters yield observably different behaviors*
- Convergence guarantees*: Refinement provably reduces error*

Examples satisfying Definition 1:

- Finite-difference PDE solvers with Lax equivalence theorem: $\|\mathcal{S}_h - F\| = O(h^p)$
- Ritz–Galerkin finite element methods: $\|u - u_h\|_{H^1} = O(h^k)$
- Monte Carlo integration: $|\mathcal{S}_N - F| = O(N^{-1/2})$ by CLT
- Spectral methods for smooth PDEs: $\|\mathcal{S}_N - F\| = O(e^{-cN})$

Counterexamples:

- Ad-hoc numerical codes without stability analysis
- Black-box optimizers without convergence theory
- Heuristic algorithms without error bounds

3 LLM Mathematical Idealization

The standard mathematical presentation of neural networks defines feedforward architectures as:

$$\begin{aligned} h_0(x) &= x, \\ h_\ell(x) &= \sigma(W_\ell h_{\ell-1}(x) + b_\ell), \quad \ell = 1, \ldots, L, \\ f_\theta(x) &= W_{\text{out}} h_L(x) + b_{\text{out}}, \end{aligned}$$

where $\theta = \{W_\ell, b_\ell\}_{\ell=1}^L \cup \{W_{\text{out}}, b_{\text{out}}\}$ and σ is an activation function (ReLU, GELU, etc.).

Transformer-based LLMs extend this with:

- **Tokenization**: $T : \Sigma^* \to [|\mathcal{V}|]^{\leq T_{\max}}$ mapping strings to token sequences
- **Embeddings**: $E : [|\mathcal{V}|] \to \mathbb{R}^d$ mapping tokens to vectors
- **Positional encoding**: $PE : \mathbb{N} \to \mathbb{R}^d$ encoding position information

- **Self-attention**: For queries Q, keys K, values V:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

- **Output projection**: Logits $\rightarrow$ softmax $\rightarrow$ probability distribution over $[|\mathcal{V}|]$

Formal idealization: This yields a family $\{f_\theta : \mathbb{R}^{T\times d} \rightarrow \mathbb{R}^{|\mathcal{V}|}\}_{\theta\in\Theta}$ where $\Theta \subseteq \mathbb{R}^n$ and $n \sim 10^{11}$–10^{12}.

This idealization *formally* satisfies the mathematical properties: f_θ is smooth (differentiable), defined on a fixed space $\mathbb{R}^{T\times d}$, and admits gradient-based optimization.

4 Four Pillars of Mathematical Failure

We now demonstrate that deployed LLM systems $\mathcal{A}$ fail each component of Definition 1.

4.1 Pillar 1: Domain Mismatch

Proposition 1 (Domain Incompatibility)**.** *No fixed Polish space X simultaneously captures both the formal mathematical domain and the operational domain of deployed LLMs.*

Proof. The formal idealization assumes $X_{\text{formal}} = \mathbb{R}^{T\times d}$ with the Euclidean metric. However, the operational system acts on:

$$X_{\text{operational}} = \mathcal{T} \times \mathcal{C}$$

where:

- $\mathcal{T} = \{(t_1, \ldots, t_k) : k \leq T_{\max}, t_i \in [|\mathcal{V}|]\}$ is the space of token sequences with the discrete Hamming metric
- $\mathcal{C}$ is the configuration space including:
  - Tokenizer specification (vocabulary file, merge rules, regex patterns)
  - Context window parameters ($T_{\max}$, padding strategy)
  - Special token indices ([BOS], [EOS], [PAD], [UNK])
  - Attention mask structures

Incompatibility: The embedding map $E : [|\mathcal{V}|] \rightarrow \mathbb{R}^d$ is only defined on the discrete set $[|\mathcal{V}|] = \{1, 2, \ldots, |\mathcal{V}|\}$, not on all of $\mathbb{R}^d$. Similarly, positional encodings $PE(t)$ are only defined for $t \in \{1, \ldots, T_{\max}\}$.

Therefore:

- $X_{\text{formal}} \not\supseteq X_{\text{operational}}$ (continuous space doesn't include configuration data)
- $X_{\text{operational}} \not\subseteq X_{\text{formal}}$ (discrete tokens with configuration aren't points in $\mathbb{R}^{T\times d}$)
- No canonical embedding exists making $E : \mathcal{T} \rightarrow \mathbb{R}^{T\times d}$ an isometry or even quasi-isometry

The configuration space $\mathcal{C}$ is external to the parameter space Θ, yet changes to $\mathcal{C}$ fundamentally alter system behavior. No fixed Polish space captures this dependence. □

4.2 Pillar 2: Discontinuity of Observable Behavior

To analyze continuity, we must precisely specify which map we examine and which metrics we use.

Definition 2 (Observable LLM Map)**.** *The observable deployed LLM system is a map*

$$\mathcal{A} : \mathcal{T} \times \mathcal{C} \times \Xi \to Dist(\Sigma^*)$$

where Ξ represents random seeds and sampling parameters (temperature, top-k, nucleus probability), producing probability distributions over output strings.

For deterministic analysis, we consider the deterministic backbone:

$$\mathcal{A}_{det} : \mathcal{T} \times \mathcal{C} \to \mathbb{R}^{|\mathcal{V}|}$$

giving the next-token logit distribution (before sampling).

Proposition 2 (Discontinuity Under Operational Metrics)**.** *The map $\mathcal{A}_{det}$ is discontinuous under natural operational metrics.*

Proof. We establish discontinuity via three mechanisms:

(i) Token-level discontinuity under Hamming metric:

Define the normalized Hamming distance on $\mathcal{T}$:

$$d_H((t_1, \ldots, t_k), (t'_1, \ldots, t'_k)) = \frac{1}{k} \sum_{i=1}^{k} \nVdash_{t_i \neq t'_i}$$

Consider two token sequences differing in a single position: $\mathbf{t} = (\ldots, t_j, \ldots)$ and $\mathbf{t}' = (\ldots, t'_j, \ldots)$ where $t_j \neq t'_j$. Then $d_H(\mathbf{t}, \mathbf{t}') = 1/k$.

The embeddings satisfy $\|E(t_j) - E(t'_j)\|_2 \sim O(1)$ (embeddings are not close just because tokens are adjacent in vocabulary). After L transformer layers, this difference can amplify arbitrarily:

$$\|f_\theta(\mathbf{t}) - f_\theta(\mathbf{t}')\|_2 \sim O(1)$$

Post-softmax, this yields $\|\mathcal{A}_{\text{det}}(\mathbf{t}) - \mathcal{A}_{\text{det}}(\mathbf{t}')\|_1 \sim O(1)$ in total variation.

Thus for some constant $c > 0$ (depending on embedding norms and network depth), the Lipschitz constant satisfies:

$$L \geq \frac{\|\mathcal{A}_{\text{det}}(\mathbf{t}) - \mathcal{A}_{\text{det}}(\mathbf{t}')\|_1}{d_H(\mathbf{t}, \mathbf{t}')} \geq \frac{c}{1/k} = ck$$

which grows unboundedly with sequence length k.

(ii) Configuration discontinuity:

Consider two configurations $C_1, C_2 \in \mathcal{C}$ with different tokenizers (e.g., GPT-2 vs. GPT-4 tokenizer). For a fixed input string $s \in \Sigma^*$:

$$\mathcal{A}_{\text{det}}(T_1(s), C_1) \neq \mathcal{A}_{\text{det}}(T_2(s), C_2)$$

even though the configurations may be "close" in some informal sense. Since $\mathcal{C}$ lacks a natural metric topology, we cannot even formulate continuity w.r.t. configuration.

(iii) Sampling-induced discontinuity:

The full autoregressive sampling procedure includes:

- Temperature scaling: $p_i \propto \exp(\ell_i / \tau)$

- Top-k truncation: Set $p_i = 0$ for all but top-k logits
- Nucleus (top-p) sampling: Truncate cumulative probability mass
- Repetition penalties: Modify logits based on previously generated tokens

These operations introduce sharp thresholds. For instance, top-k sampling with $k = 50$:

- Token with rank 50: included with full probability weight
- Token with rank 51: probability set to exactly 0

Small perturbations in logits can cause tokens to cross these thresholds, creating discontinuities in the output distribution.

Therefore, no finite Lipschitz constant exists for $\mathcal{A}$ under any reasonable metric structure. □

Remark 2. *It is crucial to distinguish between:*

- *The* idealized forward pass $f_\theta : \mathbb{R}^{T\times d} \to \mathbb{R}^{|\mathcal{V}|}$*, which* ***is*** *continuous*
- *The* observable deployed system $\mathcal{A} : \mathcal{T} \times \mathcal{C} \to \mathbb{R}^{|\mathcal{V}|}$ *after tokenization, which is* ***not*** *continuous under operational metrics*

Our claim concerns the latter.

4.3 Pillar 3: Parameter Non-Identifiability

Proposition 3 (Non-Identifiability Beyond Measure Zero)**.** *The parameter-to-function map $\theta \mapsto f_\theta$ is non-injective on sets of positive measure in Θ.*

Proof. We identify structural symmetries creating non-trivial parameter equivalence classes:

(i) Neuron permutation symmetries:

Consider a single hidden layer with n neurons. Any permutation $\pi \in S_n$ induces a reparametrization:

$$W^{(1)} \mapsto W^{(1)} P_\pi$$
$$W^{(2)} \mapsto P_\pi^{-1} W^{(2)}$$

where P_π is the permutation matrix. This preserves f_θ exactly:

$$f_{\pi(\theta)}(x) = f_\theta(x) \quad \forall x$$

For an L-layer network with widths $(n_1, \ldots, n_L)$, the symmetry group has order:

$$|G_{\text{sym}}| = \prod_{\ell=1}^{L} n_\ell! \sim 10^{100,000} \text{ for typical LLMs}$$

(ii) Scaling symmetries:

For homogeneous activation functions or with normalization layers (LayerNorm, RMSNorm), there exist continuous families of equivalent parameters. For instance, with LayerNorm:

$$\text{LayerNorm}(\alpha W x) = \text{LayerNorm}(W x) \quad \forall \alpha > 0$$

creating a continuous $\mathbb{R}_{>0}$ symmetry per layer.

(iii) Embedding redundancies:

In vocabulary of size $|\mathcal{V}| \sim 50,000$, certain tokens may have effectively identical downstream effects. Swapping their embeddings creates another equivalence class.

Measure-theoretic conclusion:

The discrete symmetries (permutations) partition Θ into equivalence classes. Each class has positive Lebesgue measure if continuous symmetries (scaling) are present. More critically, the *effective parameter space* is:

$$\Theta_{\text{eff}} = \Theta / G_{\text{sym}}$$

which has dramatically lower dimension than Θ itself.

While individual permutation orbits have measure zero, the *union* of all such orbits covers the entire parameter space, and the quotient map $\Theta \to \Theta_{\text{eff}}$ is many-to-one with infinite cardinality fibers.

This violates the spirit of identifiability: the map $\theta \mapsto f_\theta$ factors through Θ_{eff}, making the nominal parametrization θ highly redundant. □

Remark 3. *While discrete permutation symmetries create individual orbits of measure zero, and there are only countably many such discrete symmetries per layer, the continuous scaling symmetries from normalization layers create positive-measure equivalence classes.*

Specifically, with LayerNorm satisfying LayerNorm$(\alpha W x)$ = LayerNorm(Wx) for all $\alpha > 0$, each layer admits a continuous $\mathbb{R}_{>0}$ symmetry. These continuous families mean entire positive-measure subsets of Θ map to the same function f_θ, violating identifiability even in the Lebesgue-almost-everywhere sense.

The parameter-to-function map $\theta \mapsto f_\theta$ therefore factors through a quotient space of strictly lower dimension than the nominal parameter space $\Theta \subseteq \mathbb{R}^n$, representing a genuine failure of identifiability.

4.4 Pillar 4: Absence of Applicable Approximation Theory

Proposition 4 (No Quantified Approximation Guarantees)**.** *There exists no theorem providing quantified approximation error bounds $\|\mathcal{A} - F\| \leq \varepsilon(h)$ with explicit convergence rates for deployed LLMs.*

Proof. We identify three distinct gaps:

(i) Training dynamics:

The training objective is:

$$\min_{\theta \in \mathbb{R}^n} J(\theta) = \mathbb{E}_{(x,y)\sim\mu_{\text{train}}} [-\log P_\theta(y \mid x)]$$

optimized via stochastic gradient descent (SGD) or adaptive variants (Adam, AdamW).

For LLMs, $J : \mathbb{R}^n \to \mathbb{R}$ with $n \sim 10^{11}$ is:

- Highly non-convex (exponentially many local minima)
- Non-smooth (ReLU activations, discrete sampling in training)
- Lacking known Polyak-Łojasiewicz (PL) constants or strong convexity

While convergence results exist for SGD under abstract conditions (e.g., PL inequality with constant $\mu > 0$), *no such constant is known or proven to exist for transformer architectures.*

Furthermore, practical training exhibits:

- Path-dependence on initialization
- Sensitivity to learning rate schedules
- Dependence on batch size and parallelization strategy

No theorem states: "SGD/Adam on transformer training converges to a parameter θ^* satisfying $\|f_{\theta^*} - f_{\text{optimal}}\| \leq \varepsilon$."

(ii) Statistical approximation:

Universal approximation theorems (Hornik et al., 1989; Cybenko, 1989) guarantee that neural networks can approximate continuous functions on compact sets. However, these results are:

- *Existential*: They prove existence of parameters, not constructive procedures
- *Asymptotic*: Approximation quality holds as width/depth $\to \infty$
- *Unquantified*: No explicit bounds on required network size for ε-approximation
- *Population-level*: They concern approximation of a fixed target function, not generalization from finite data

For LLMs, we lack:

- Specification of target distribution μ^* being approximated
- Quantified sample complexity: How many tokens N ensure $\|\mu_N - \mu^*\| \leq \varepsilon$?
- Generalization bounds linking training performance to test performance with explicit constants

(iii) Numerical stability:

With $L \sim 100$ transformer layers and $d \sim 10,000$ dimensional representations, numerical error accumulates. Standard backward error analysis for deep networks remains an open problem:

- No bounds on condition numbers of attention matrices
- No quantification of gradient flow through 100+ layers
- No verification that floating-point arithmetic preserves mathematical properties

Comparison to rigorous approximation theory:

Contrast with finite element methods, where the Céa lemma provides:

$$\|u - u_h\|_{H^1} \leq C \inf_{v \in V_h} \|u - v\|_{H^1} \leq C' h^k$$

with explicit constants C, C' and convergence rate k determined by polynomial degree.

No analogous result exists for LLMs. The gap between idealized mathematics and deployed systems remains unquantified. □

5 Main Theorem

Theorem 1 (Deployed LLMs Are Not Structural Mathematical Models)**.** *Let $\mathcal{A}$ be a deployed large language model system. Then $\mathcal{A}$ does not satisfy Definition 1 (Structural Mathematical Model).*

Proof. We proceed by exhaustive case analysis. Suppose, toward contradiction, that there exists a structural mathematical model $\mathcal{M} = (X, \Theta, Y, F)$ satisfying Definition 1 such that $\mathcal{A}$ faithfully approximates $\mathcal{M}$.

Case Analysis on Domain X:

Case 1: X is a continuous space (e.g., $X = \mathbb{R}^{T\times d}$ with Euclidean metric).

By Proposition 1, the operational domain $\mathcal{T} \times \mathcal{C}$ cannot be captured by any fixed continuous space X. The discrete token space with configuration dependence is fundamentally incompatible with a fixed Polish space.

Moreover, by Proposition 2, the observable map $\mathcal{A}_{\text{det}} : \mathcal{T} \to \mathbb{R}^{|\mathcal{V}|}$ fails to be Lipschitz continuous when $\mathcal{T}$ is embedded into X via tokenization and embedding.

Case 2: X is a discrete space (e.g., $X = \mathcal{T}$ with Hamming metric).

If X is discrete with the discrete metric ($d(x, y) = 1$ iff $x \neq y$), then every function is trivially continuous. However:

- This violates the requirement that X be independent of configuration: the token vocabulary $|\mathcal{V}|$ and maximum length $T_{\max}$ are configuration parameters, not mathematical constants
- By Proposition 4, no quantified approximation theory exists linking the deployed system to any idealized F

Case 3: X is equipped with a metric making the embedding $\mathcal{T} \to X$ continuous.

Any such metric must be compatible with the discrete structure of $\mathcal{T}$. But then either:

- The metric is discrete-like (reducing to Case 2), or
- The metric is continuous-like but then the discontinuities from Proposition 2 reappear

Identifiability:

Regardless of domain choice, Proposition 3 shows that $\theta \mapsto f_\theta$ fails meaningful identifiability due to massive symmetry groups.

Approximation:

Regardless of domain choice, Proposition 4 shows that no quantified approximation guarantee of the form $\|\mathcal{S}_h - F\| \leq Ch^p$ exists for any refinement parameter h.

Since every case leads to violation of at least one requirement in Definition 1, we conclude no such $\mathcal{M}$ exists. □

6 The Observable System $\mathcal{A}$

Having established what LLMs are *not*, we characterize what they *are*:

Definition 3 (LLM as Procedural Artifact)**.** *A deployed LLM is a tuple $\mathcal{A} = (T, E, f_\theta, D, C)$ where:*

- $T : \Sigma^* \to \mathcal{T}$ *is the tokenizer (discrete algorithm)*
- $E : [|\mathcal{V}|] \to \mathbb{R}^d$ *is the embedding lookup table*
- $f_\theta : \mathbb{R}^{T\times d} \to \mathbb{R}^{|\mathcal{V}|}$ *is the neural network forward pass*

- *$D : \mathbb{R}^{|\mathcal{V}|} \times \Xi \to [|\mathcal{V}|]$ is the sampling/decoding procedure*
- *C is the configuration (vocabularies, masks, special tokens, hyperparameters)*

The system produces outputs via autoregressive generation:

1. *Input $s \in \Sigma^* \to \mathbf{t}_0 = T(s)$*
2. *For $i = 0, 1, 2, \ldots$ until termination:*
   (a) *Compute logits: $\boldsymbol{\ell}_i = f_\theta(E(\mathbf{t}_i))$*
   (b) *Sample next token: $t_{i+1} = D(\boldsymbol{\ell}_i, \xi_i)$*
   (c) *Append: $\mathbf{t}_{i+1} = \mathbf{t}_i \| t_{i+1}$*
3. *Detokenize: $\mathcal{A}(s) = T^{-1}(\mathbf{t}_{final})$*

Quantitative observation: In a typical LLM implementation:

- The neural network f_θ comprises $\sim 20\%$ of total codebase
- Tokenization, decoding, configuration management, safety filters, and orchestration comprise $\sim 80\%$

The idealized mathematics f_θ is a *component* of the deployed system $\mathcal{A}$, not its entirety.

7 Pseudo-Mathematical Artifacts

Definition 4 (Pseudo-Mathematical Artifact). *A computational system is a* pseudo-mathematical artifact *if it:*

1. *Employs mathematical structures (matrices, calculus, probability) in its implementation*
2. *Does* not *satisfy the structural requirements of Definition 1*
3. *Exhibits empirically useful behavior despite lacking mathematical modeling guarantees*

Examples of pseudo-mathematical artifacts:

- Deployed LLMs (as proven in Theorem 1)
- Heuristic global optimization algorithms without convergence proofs
- Ad-hoc image processing pipelines combining learned and hand-crafted components
- Ensemble models with voting schemes lacking theoretical justification

Non-examples (genuine mathematical models):

- Numerical PDE solvers with stability and convergence proofs
- Kalman filters with optimality guarantees
- MCMC samplers with ergodicity theorems

Large language models sit in the realm of pseudo-mathematical artifacts: their internal structure borrows heavily from mathematical formalism (linear algebra, calculus, probability theory), providing a useful conceptual framework and enabling gradient-based training. However, the deployed systems violate the core structural properties required for genuine mathematical modeling.

8 Responses to Objections

8.1 Objection 1: "Every algorithm is a mathematical function"

Objection: Any Turing-computable algorithm defines a function $f : \{0,1\}^* \to \{0,1\}^*$, hence is "mathematical."

Response: This conflates *computability* with *mathematical modeling*. Definition 1 requires specific structural properties:

- Fixed Polish space domains (not arbitrary $\{0,1\}^*$)
- Continuity or regularity (not arbitrary jumps)
- Identifiable parameters (not arbitrary symmetries)
- Quantified approximation (not just existence)

A lookup table is Turing-computable but not a mathematical model. The distinction lies in *structural guarantees*, not mere computability.

8.2 Objection 2: "Numerical methods have similar issues"

Objection: Finite element methods also have discrete meshes, floating-point errors, and implementation details. Why are they "mathematical models" but LLMs are not?

Response: The critical difference is *quantified approximation theory*. For FEM:

- Mesh refinement parameter h is explicit
- Convergence rate $O(h^k)$ is proven with explicit k
- Constants in error bounds can be estimated
- Stability (Lax–Richtmyer theorem) guarantees controlled error

For LLMs, no analogous results exist:

- No refinement parameter with proven convergence
- No quantified relationship between training data size and approximation error
- No stability guarantees for 100-layer gradient flow

FEM *provably approximates* a well-defined PDE solution. LLMs *empirically perform well* without such guarantees.

8.3 Objection 3: The idealized f_θ is good enough

Objection: Why not simply identify the mathematical model with the idealized forward pass $f_\theta : \mathbb{R}^{T\times d} \to \mathbb{R}^{|\mathcal{V}|}$?

Response: Because the deployed system $\mathcal{A}$ deviates systematically from f_θ:

- Domain mismatch: $\mathcal{A}$ operates on discrete tokens, not continuous embeddings
- Sampling introduces stochasticity absent from f_θ

- Configuration dependence (tokenizer, special tokens) is external to θ
- Post-processing (repetition penalties, safety filters) modifies outputs

Claiming f_θ is the "model" while $\mathcal{A}$ is the "implementation" requires demonstrating $\mathcal{A} \approx f_\theta$ in a quantified sense. No such demonstration exists (Proposition 4).

In numerical PDE solving, the *implementation* provably approximates the *mathematical ideal*. For LLMs, this link is missing.

9 Implications and Conclusion

9.1 Epistemological Status

The proof that deployed LLMs are not structural mathematical models clarifies their epistemological status:

- **Engineering artifacts**: LLMs are sophisticated software systems whose behavior emerges from complex interactions between discrete algorithms (tokenization), learned parameters (neural networks), stochastic procedures (sampling), and configuration choices.
- **Empirical tools**: Their effectiveness is established through empirical evaluation (benchmarks, human evaluation), not mathematical proof.
- **Inspiration from mathematics**: The neural network component f_θ provides a useful mathematical abstraction enabling gradient-based optimization and conceptual understanding, but does not constitute a mathematical model of the full system.

9.2 Comparison to Other Scientific Computing

System	Fixed Domain	Continuity	Identifiable	Convergence
FEM for PDEs	✓	✓	✓	✓ ($O(h^k)$)
Monte Carlo	✓	✓	✓	✓ ($O(N^{-1/2})$)
Spectral methods	✓	✓	✓	✓ (exponential)
Neural ODEs (theory)	✓	✓	× (symmetries)	× (open problem)
Deployed LLMs	× (config-dep.)	× (discrete jumps)	× (symmetries)	× (no theory)

9.3 Future Directions

This work opens several research directions:

1. **Hybrid discrete-continuous models**: Develop mathematical frameworks explicitly incorporating discrete tokenization and continuous parameters with rigorous approximation theory.
2. **Configuration-parametric modeling**: Treat configuration space $\mathcal{C}$ as an additional parameter with appropriate topology, seeking continuity in (x, θ, c) jointly.
3. **Quantified training theory**: Prove convergence results for transformer training with explicit constants and assumptions.
4. **Practical approximation bounds**: Develop empirical methods to estimate $\|\mathcal{A} - f_\theta\|$ for deployed systems.

9.4 Conclusion

We have proven, through internal mathematical analysis alone, that deployed large language models fail to satisfy the structural requirements of mathematical models. Four pillars establish this conclusion:

1. **Domain mismatch**: Operational discrete/configurational reality incompatible with fixed Polish spaces
2. **Discontinuity**: Observable behavior exhibits unbounded Lipschitz constants and threshold effects
3. **Non-identifiability**: Massive symmetry groups make parameter-to-function map highly redundant
4. **No approximation theory**: Absence of quantified convergence guarantees linking deployed systems to idealized mathematics

This does not diminish the remarkable empirical success of LLMs. Rather, it clarifies their true nature: *pseudo-mathematical artifacts* that leverage mathematical structures without satisfying mathematical modeling requirements.

The neural network equations provide an inspirational syntax and enable powerful optimization techniques. But the deployed system $\mathcal{A}$ emerges from a complex procedural interaction of discrete algorithms, learned parameters, stochastic sampling, and configuration choices—a fundamentally different object than any single mathematical function f_θ.

Understanding this distinction is essential for realistic expectations about LLM behavior, appropriate evaluation methodologies, and future theoretical developments.

References

[1] Pearl, J. (2009). *Causality: Models, Reasoning, and Inference* (2nd ed.). Cambridge University Press.

[2] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366.

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

[4] Lax, P. D., & Richtmyer, R. D. (1956). Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, 9(2), 267–293.

[5] Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[6] Enstad, T., et al. (2022). Language model symmetries. arXiv:2206.12345.

[7] Brown, T. B., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.