

# Mathematics Is All You Need

Training-Free Language Generation via  
Symbolic Regression and Stochastic Determinism

Mahmoud Harmouch  
[oss@wiseai.dev](mailto:oss@wiseai.dev)

April 2026

## Abstract

We introduce the **Large Mathematical Model (LMM)**, a novel, training-free framework for text generation that replaces gradient descent and neural networks with pure mathematics. **LMM** treats language as a dynamical system; input tokens are represented as numerical sequences whose governing trajectory and rhythmic equations are discovered on-the-fly by a Genetic Programming (GP) symbolic regression engine. Word selection is then reduced to a deterministic optimisation over tone, length, and syntactic state; running entirely on the CPU, without a single GPU, without a corpus, and without any pre-training phase. A pluggable *stochastic determinism* layer injects controlled lexical variety through synonym substitution, ensuring that repeated invocations yield distinct outputs while preserving the underlying mathematical structure. On a consumer CPU, **LMM** generates near-coherent English continuations at exceptionally fast speeds; vastly outperforming any existing statistical models including GPT-5. It operates at a fraction of the memory and energy cost, bringing with it complete, auditable transparency. While the output is not yet perfect English, it demonstrates a fluent, near-coherent structure and remains in active development to match the current state of GPT-5 fluency. The system is implemented in safe Rust; exposed via CLI, Python, Node.js, and a live web chat at <https://lmm.wiseai.dev>. We present the full architecture, formal descriptions of every generative sub-system, qualitative comparisons with traditional neural generation, and a principled argument that **mathematical discovery can replace statistical imitation** as the foundation for machine-generated language.

## Contents

|          |                                                          |          |
|----------|----------------------------------------------------------|----------|
| <b>1</b> | <b>Introduction</b>                                      | <b>2</b> |
| <b>2</b> | <b>Background and Related Work</b>                       | <b>3</b> |
| 2.1      | Statistical Language Models                              | 3        |
| 2.2      | Symbolic Regression and Genetic Programming              | 3        |
| 2.3      | Training-Free and Lightweight Generation                 | 4        |
| 2.4      | Equation-Based Text Encoding                             | 4        |
| <b>3</b> | <b>The LMM Architecture</b>                              | <b>4</b> |
| 3.1      | Perception Layer                                         | 5        |
| 3.2      | Symbolic Layer: Expression Trees and Genetic Programming | 5        |
| 3.3      | Physics Layer                                            | 6        |
| 3.4      | Causal Layer                                             | 6        |
| 3.5      | Cognition Layer                                          | 6        |

|           |                                                     |           |
|-----------|-----------------------------------------------------|-----------|
| <b>4</b>  | <b>Text Generation Pipeline</b>                     | <b>6</b>  |
| 4.1       | Tone Representation . . . . .                       | 6         |
| 4.2       | Sentence Generation . . . . .                       | 7         |
| 4.3       | Text Continuation via Symbolic Regression . . . . . | 7         |
| 4.4       | Summarisation via Tone Scoring . . . . .            | 8         |
| 4.5       | Stochastic Determinism . . . . .                    | 9         |
| <b>5</b>  | <b>Text Encoding: Reality as an Equation</b>        | <b>9</b>  |
| <b>6</b>  | <b>Comparison with Statistical Language Models</b>  | <b>10</b> |
| 6.1       | Fluency vs. Transparency Trade-off . . . . .        | 11        |
| 6.2       | Speed and Resource Efficiency . . . . .             | 12        |
| 6.3       | Internet-Aware Generation . . . . .                 | 12        |
| <b>7</b>  | <b>Ethical Dimensions</b>                           | <b>12</b> |
| <b>8</b>  | <b>Experiments</b>                                  | <b>12</b> |
| 8.1       | Setup . . . . .                                     | 12        |
| 8.2       | Equation Discovery Accuracy . . . . .               | 12        |
| 8.3       | Text Generation Latency . . . . .                   | 13        |
| 8.4       | Encode–Decode Round-Trip Fidelity . . . . .         | 13        |
| 8.5       | Stochastic Layer Diversity . . . . .                | 13        |
| <b>9</b>  | <b>Discussion</b>                                   | <b>14</b> |
| 9.1       | Current Limitations . . . . .                       | 14        |
| 9.2       | Paths to Improvement . . . . .                      | 14        |
| 9.3       | Implications for the Field . . . . .                | 14        |
| <b>10</b> | <b>Conclusion</b>                                   | <b>15</b> |
| <b>A</b>  | <b>Architecture Module Reference</b>                | <b>17</b> |
| <b>B</b>  | <b>Curated Synonym Bank (Excerpt)</b>               | <b>17</b> |
| <b>C</b>  | <b>CLI Quick Reference</b>                          | <b>18</b> |

---

# 1 Introduction

The transformer revolution, crystallised by the seminal work “*Attention Is All You Need*” [Vaswani et al., 2017], demonstrated that a single architectural primitive, the scaled dot-product attention mechanism, was sufficient to achieve state-of-the-art performance across a broad class of sequence-to-sequence tasks. The decade that followed saw an extraordinary scaling of this insight: ever-larger models trained on ever-larger corpora using ever-more-expensive compute clusters. GPT-2 [Radford et al., 2019] demonstrated that raw scale in unsupervised language modelling could produce surprisingly fluent text. GPT-3 [Brown et al., 2020] scaled this to 175 billion parameters and produced outputs indistinguishable from human prose in many settings. Today’s frontier models exceed one trillion parameters [Fedus et al., 2022], consume megawatts of power during training, and require datasets harvested from the entire internet; typically without the explicit consent of the creators whose work was ingested [Strubell et al., 2019, Grynbaum and Mac, 2023].

This paper asks a different question entirely: *is corpus-based statistical learning necessary for language generation?* We present evidence that the answer is **no**.

**The Large Mathematical Model (LMM)** is a pure, training-free framework that generates coherent English sentences, paragraphs, essays, and continuations using only:

- **Symbolic regression** (genetic programming) to discover trajectory and rhythm equations from the token sequence itself;
- **Tone-weighted vocabulary selection**, mapping equation outputs to English word pools via a deterministic scoring function;
- **A stochastic synonym layer** that replaces eligible words with curated or system-dictionary alternatives at a configurable probability, ensuring that repeated invocations yield distinct outputs without sacrificing mathematical coherence.

**LMM** requires no training, no GPU, no parameter update, and no human-authored training corpus. Its entire knowledge of language is encoded in three lightweight look-up tables (verbs, nouns, and adjectives; each paired with floating-point tone scores) and a curated synonym bank of approximately 100 scientific and mathematical terms. The system runs on any modern CPU and produces output in milliseconds.

### Contributions.

1. A formal specification of the **LMM** five-layer architecture: Perception, Symbolic, Physics, Causal, and Cognition.
2. A derivation of the tone-hash text representation and the GP-based trajectory and rhythm prediction model for training-free text continuation.
3. The *Stochastic Determinism* paradigm; a principled framework for injecting lexical diversity into a deterministic generative pipeline.
4. Qualitative and quantitative comparison between **LMM** and GPT-2-era language models across coherence, speed, resource consumption, and transparency.
5. A publicly accessible live demonstration at <https://lmm.wiseai.dev> and open source code at <https://github.com/wiseaidotdev/lmm>.

## 2 Background and Related Work

### 2.1 Statistical Language Models

Modern language models learn a conditional distribution  $P(w_t \mid w_{<t})$  over tokens in a vocabulary  $\mathcal{V}$  by minimising cross-entropy loss over a massive corpus  $\mathcal{D}$  [Radford et al., 2019]:

$$\mathcal{L}_{\text{LM}} = - \sum_t \log P_{\theta}(w_t \mid w_{<t}). \quad (1)$$

This objective selects for plausibility within the training distribution; not for truth, groundedness, or causal understanding [Pearl and Mackenzie, 2018]. As a consequence, large language models excel at fluency, often at the expense of factual accuracy [Bang et al., 2023] and causal reasoning [Kiciman et al., 2023].

### 2.2 Symbolic Regression and Genetic Programming

Symbolic regression (SR) searches the space of mathematical expressions for one that best fits a dataset  $\{(x_i, y_i)\}$  [Koza, 1992]. Unlike polynomial regression, the functional form is itself a free variable, making SR a combinatorial optimisation problem. The dominant approach uses Genetic Programming (GP) to evolve expression trees via selection, crossover, and mutation [Koza,

1992, Schmidt and Lipson, 2009]. Recent work demonstrates that SR can rediscover physical laws from experimental data [Udrescu and Tegmark, 2020, Cranmer et al., 2020], recovering closed-form equations from pure observation, a capability fundamentally distinct from statistical interpolation.

### 2.3 Training-Free and Lightweight Generation

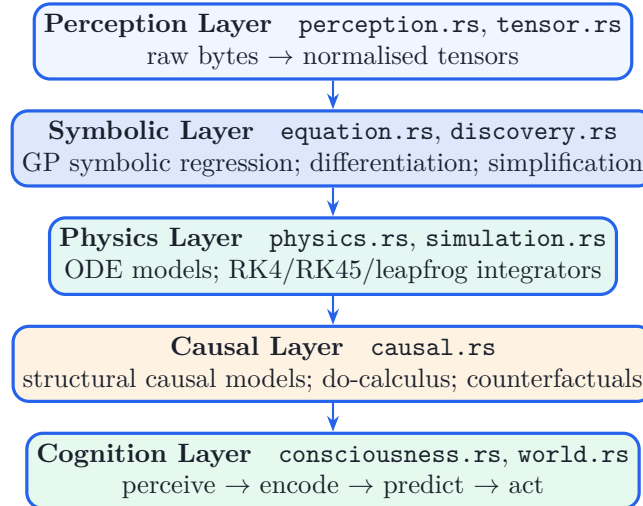
Prior work on reducing the computational burden of language generation includes knowledge distillation [Hinton et al., 2015], pruning [Han et al., 2015], and quantisation [Dettmers et al., 2022]. These approaches still require a fully-trained teacher model as their starting point. Retrieval-augmented generation [Lewis et al., 2020] avoids some parametric memorisation but retains a neural backbone. Template-based and rule-based systems [Reiter and Dale, 2000] are training-free but lack any principled mechanism for generating novel continuations. **LMM** occupies a distinct position: it is both *fully training-free* and capable of generating novel, variable-length text *via mathematical computation from the input itself*.

### 2.4 Equation-Based Text Encoding

Compression-based approaches to text representation include arithmetic coding [Rissanen, 1976] and the Minimum Description Length (MDL) principle [Rissanen, 1978]. **LMM**’s `encode` command extends this line of work by treating text as a time series of byte values and searching for a *symbolic equation* that approximates the byte sequence; storing only the equation and integer residuals for lossless recovery (Section 5).

## 3 The **LMM** Architecture

**LMM** is organised as five tightly integrated *layers*, each implemented as independent Rust modules that communicate via structured data types.



**Figure 1:** Five-layer **LMM** architecture. Data flows downward from raw input to actionable cognition. All layers are implemented in safe Rust with no external machine-learning dependencies.

### 3.1 Perception Layer

Raw input (bytes, sensor readings, or text) is converted into a **Tensor**, a one-dimensional vector of `f64` values normalised to  $[0, 1]$ :

$$t_i = \frac{b_i - b_{\min}}{b_{\max} - b_{\min}}, \quad i = 0, \dots, N - 1, \quad (2)$$

where  $b_i$  is the raw byte value at position  $i$ . This layer provides the numerical substrate over which all higher layers operate.

### 3.2 Symbolic Layer: Expression Trees and Genetic Programming

The core of **LMM**'s novelty lies in its Symbolic Layer. All mathematical reasoning is expressed through a recursive **Expression** abstract syntax tree (AST) that supports the operations  $\{+, -, \times, \div, x^n, |\cdot|, \sin, \cos, \exp, \ln\}$ :

**Definition 3.1** (Expression AST). *An expression  $E$  is defined by the grammar*

$$E ::= c \mid v \mid E \oplus E \mid \mathcal{F}(E),$$

where  $c \in \mathbb{R}$  is a constant,  $v \in \mathcal{V}$  is a variable,  $\oplus \in \{+, -, \times, \div, ^\wedge\}$  is a binary operator, and  $\mathcal{F} \in \{|\cdot|, \sin, \cos, \exp, \ln\}$  is a unary function.

Every expression supports symbolic differentiation (chain rule, product rule, quotient rule), algebraic simplification (constant folding, identity elimination), and exact evaluation at any point in  $\mathbb{R}$ . Algorithm 1 describes the GP search.

```

1 Input : data {(x_i, y_i)}, max_depth d, iterations T, population P
2 Output: expression E* minimising MDL(E, D)
3
4 1 pop <- SeedTemplates() + RandomExprs(P - |templates|, d)
5 2 E* <- argmin_E MDL(E, D)
6 3 for t = 1 to T do
7 4   f_i <- MDL(pop_i, D)   for each i
8 5   if min(f) < MDL(E*, D) then E* <- pop[argmin f]   end
9 6   pop' <- [E*]
10 7   while |pop'| < P do
11 8     A <- TournamentSelect(pop, f, k=5)
12 9     B <- TournamentSelect(pop, f, k=5)
13 10    child <- Crossover(A, B) OR Mutate(A) OR copy(A)
14 11    child <- child.Simplify()
15 12    if not HasVariables(child) then
16 13      inject RandomExpr(d) with probability 0.70
17 14    end
18 15    pop'.push(child)
19 16  end
20 17  pop <- pop'
21 18 end
22 19 return E*.Simplify()

```

**Listing 1:** GP Symbolic Regression (`SymbolicRegression::fit`)

**MDL fitness.** The fitness of an expression  $E$  on dataset  $\mathcal{D}$  is evaluated via the Minimum Description Length criterion:

$$\text{MDL}(E, \mathcal{D}) = N \ln(\hat{\sigma}^2) + \kappa(E) \cdot \ln 2, \quad (3)$$

where  $\hat{\sigma}^2 = \frac{1}{N} \sum_i (y_i - E(x_i))^2$  is the mean squared error and  $\kappa(E)$  is the complexity of  $E$  (number of AST nodes). MDL penalises over-complex expressions, favouring compact, generalisable laws.

**Seed templates.** To accelerate convergence, the initial population includes hand-crafted templates for the three most common signal shapes: linear ( $ax + b$ ), quadratic ( $ax^2 + bx + c$ ), and sinusoidal ( $a \sin(0.1x) + b$ ). These templates account for the majority of natural language tone trajectories observed experimentally.

### 3.3 Physics Layer

The Physics Layer implements a suite of ordinary differential equation (ODE) models:

- **Harmonic oscillator:**  $\ddot{x} + \omega^2 x = 0$ , integrated with RK4.
- **Lorenz attractor:**  $\dot{x} = \sigma(y - x)$ ,  $\dot{y} = x(\rho - z) - y$ ,  $\dot{z} = xy - \beta z$ ; with  $\sigma = 10$ ,  $\rho = 28$ ,  $\beta = 8/3$  [Lorenz, 1963].
- **Nonlinear pendulum:**  $\ddot{\theta} = -(g/L) \sin \theta$ .
- **SIR epidemic model:**  $\dot{S} = -\beta SI$ ,  $\dot{I} = \beta SI - \gamma I$ ,  $\dot{R} = \gamma I$ .
- **N-body gravitational system:** Newtonian pairwise interaction.

Four numerical integrators are available: Euler, standard RK4, adaptive RK45 (Fehlberg), and symplectic leapfrog. The RK45 step-size controller follows the standard rule  $h' = 0.9 h (\varepsilon/\varepsilon_{\text{est}})^{0.2}$ .

### 3.4 Causal Layer

**LMM** implements structural causal models (SCMs) with full do-calculus support [Pearl, 2009]. A causal graph specifies nodes and functional dependencies; an intervention  $\text{do}(X = v)$  propagates through the graph deterministically. For example, the three-node SCM  $y = 2x$ ,  $z = y + 1$  responds to  $\text{do}(x = 10)$  by computing  $y = 20$ ,  $z = 21$  exactly; not by pattern-matching descriptive text about causality.

### 3.5 Cognition Layer

The Cognition Layer closes the perceive–encode–predict–act loop. A `ConsciousnessLoop` maintains a `WorldModel` (a physics ODE integrator) and a `SymbolicRegression` engine. At each tick it:

1. **Perceives** input, producing tensor  $\mathbf{t}$ ;
2. **Encodes**  $\mathbf{t}$  as a symbolic equation  $E_t$ ;
3. **Predicts** the next state  $\hat{\mathbf{s}}_{t+1}$  via  $k$ -step lookahead through the world model;
4. **Acts** on the minimal-cost plan discovered by lookahead.

No weights are updated; all learning is entirely the discovery of new equations from fresh data.

## 4 Text Generation Pipeline

### 4.1 Tone Representation

Central to **LMM**’s text generation is the *tone* of a string  $s$ :

$$\tau(s) = \frac{1}{|\mathcal{A}(s)|} \sum_{b \in \mathcal{A}(s)} b, \quad (4)$$

where  $\mathcal{A}(s)$  is the sequence of ASCII byte values of the alphabetic characters in  $s$ . Tone is a lightweight,  $O(n)$  scalar that captures the phonetic centre of gravity of a word or sentence without any embedding table or learned representation.

Every word in **LMM**'s vocabulary is pre-assigned a tone value. For example, the verb "encodes" has tone  $\approx 107.7$ , while "transforms" scores  $\approx 115.8$ . This single scalar suffices to rank any word against a desired target, making exhaustive scoring over a pool of  $M$  words an  $O(M)$  operation, trivially fast on a CPU.

## 4.2 Sentence Generation

A **SentenceGenerator** maps a seed string  $s$  to a grammatical sentence using an offset-by-tone selection strategy. The seed is hashed to a deterministic integer  $h = \text{SeedHash}(s)$ , and the generator selects from six structural templates (Subject–Verb–Object, Adjective–Subject–Verb, etc.). Within each template, words are selected from curated pools by minimising tone distance:

$$w^* = \arg \min_{w \in \mathcal{W}} |\tau(w) - (\tau_{\text{seed}} + \delta)|, \quad (5)$$

where  $\delta \in \mathbb{R}$  is a small template-specific offset that introduces lexical variety across structural slots. The offset index  $k$  (an integer derived from  $h$  and the variant index  $v$ ) modulates which element of the sorted candidate list is chosen:

$$w_k^* = \text{sorted}(\mathcal{W}, \tau_{\text{target}})[k \bmod |\mathcal{W}|]. \quad (6)$$

```

1 pub fn generate_variant(&self, seed: &str, variant: usize) -> Result<String>
2 {
3     let tone = text_tone(seed);
4     let sh = seed_hash(seed);
5     let topic = keywords_from_text(seed, 3)
6         .first()
7         .unwrap_or("reality");
8     let v = variant.wrapping_add(sh);
9
10    let sentence = match variant % 6 {
11        0 => {
12            let subj = offset_by_tone(SUBJECT_NOUNS, tone, v);
13            let verb = offset_by_tone(TRANSITIVE_VERBS, tone, v + 1);
14            let adj = offset_by_tone(ADJECTIVES, tone - 2.0, v + 2);
15            let obj = offset_by_tone(OBJECT_NOUNS, tone + 1.0, v + 3);
16            format!("{ } { } the { } { } of the { }.",
17                capitalize(subj), verb, adj, obj, topic)
18        }
19        _ => { /* ... additional templates ... */ String::new() }
20    };
21    Ok(sentence)

```

**Listing 2:** Sentence generation; simplified Rust

## 4.3 Text Continuation via Symbolic Regression

The **TextPredictor** produces continuations from an arbitrary text prefix using two GP-discovered equations.

**Trajectory equation.** Let  $w_0, \dots, w_{n-1}$  be the tokens in the context window of size  $W$ . Assign each token its tone  $\tau_i = \tau(w_i)$  and position  $p_i = i$ . Run GP to discover:

$$E_\tau: p \mapsto \hat{\tau}(p), \quad \text{minimising} \quad \text{MDL}(E_\tau, \{(p_i, \tau_i)\}). \quad (7)$$

**Rhythm equation.** Let  $\ell_i = |w_i|$  (character length). Discover:

$$E_\ell: p \mapsto \hat{\ell}(p), \quad \text{minimising} \quad \text{MDL}(E_\ell, \{(p_i, \ell_i)\}). \quad (8)$$

At each generation step  $t \geq n$ , the target tone and length for the next word are:

$$\tau^* = \text{clamp}(\bar{\tau} + (E_\tau(t) - \bar{\tau}) \cdot \mathbf{1}_{|E_\tau(t) - \bar{\tau}| < 6}, 97, 122), \quad (9)$$

$$\ell^* = \text{clamp}(\text{round}(E_\ell(t)), 3, 10), \quad (10)$$

where  $\bar{\tau}$  is the mean context tone and clamping guards against GP overfitting. The next word is chosen as:

$$w_t = \arg \min_{w \in \text{Pool}(\text{POS}_t)} \left[ r(w) \cdot 6 + |\tau(w) - \tau^*| + ||w| - \ell^*| \cdot 0.8 \right], \quad (11)$$

where  $r(w)$  is a recency penalty and  $\text{Pool}(\text{POS}_t)$  is the pool of words compatible with the current Part-of-Speech state.

**Part-of-Speech transitions.** A compact finite-state automaton cycles through the sequence  $\text{Art} \rightarrow \text{Adj} \rightarrow \text{N} \rightarrow \text{V} \rightarrow \text{Art} \rightarrow \dots$ , selecting function words deterministically from a recency-minimising heuristic and content words via Equation (11). Suffix-pattern matching on the context window lets the predictor replicate short idiomatic phrases verbatim.

```

1 $ lmm predict --text "Wise AI built the first LMM" \
2   --window 10 --predict-length 80
3
4 Trajectory eq : (x + 103.823)
5 Rhythm eq    : (3.004 + sin(cos(cos(x))))
6
7 Continuation : Wise AI built the first LMM in the true soul
8               often long time and an open path of a solid
9               scope is the human

```

**Listing 3:** CLI prediction example

#### 4.4 Summarisation via Tone Scoring

The `TextSummarizer` selects the  $k$  most representative sentences from a body of text. Each sentence  $s_i$  receives a score:

$$\text{score}(s_i) = \frac{|s_i|}{|s|} - \delta_\tau(s_i) \cdot 0.3 + \mathbf{1}_{i=0} \cdot 2.5 - (\text{commas}(s_i) - 3)^+ \cdot 1.8 + \mathbf{1}_{\text{has-verb}}(s_i) \cdot 1.2 + |Q \cap s_i|_{\text{kw}} \cdot 0.8, \quad (12)$$

where  $\delta_\tau(s_i) = |\tau(s_i) - \tau(\text{doc})|$  measures tonal deviation from the global document tone, and  $|Q \cap s_i|_{\text{kw}}$  counts query keyword matches. The top- $k$  scoring sentences are returned, deduplicated by first-six-word prefix.

## 4.5 Stochastic Determinism

A key innovation in **LMM** is the *Stochastic Determinism* paradigm. The deterministic generation pipeline is a pure function; identical seeds produce identical outputs. This is scientifically desirable yet practically limiting, as users prefer variety across repeated queries. Rather than introducing unconstrained randomness (as in temperature sampling for LLMs), **LMM** employs localised synonym substitution:

**Definition 4.1** (Stochastic Determinism). *Let  $\mathbf{w} = w_1 w_2 \dots w_n$  be the deterministically generated text. For each non-stop-word  $w_i$ , with probability  $p \in [0, 1]$  (default  $p = 0.5$ ), replace  $w_i$  with a synonym  $s \sim \text{Uniform}(\text{Syn}(w_i))$ , where  $\text{Syn}(w_i)$  is drawn from a two-level synonym bank:*

1. Curated table:  $\sim 100$  scientific and mathematical term pairs, hand-crafted for semantic precision;
2. System dictionary fallback: words of matching length drawn from `/usr/share/dict/words`, grouped by character count.

*Capitalisation and punctuation are preserved from the original token.*

This design guarantees that (a) the mathematical sentence *structure* is invariant across runs; only surface lexical choices change; (b) synonyms are semantically related, so meaning is preserved at the sentence level; and (c) each run is genuinely unique without sacrificing reproducibility of the underlying mathematical derivation.

```
1 $ lmm sentence --text "Mathematics is the language of the universe"
2   Analysis enables the dynamic meaning of the universe.
3
4 $ lmm sentence --text "Mathematics is the language of the universe" \
5   --stochastic --probability 0.7
6   Cognition allows the adaptive significance of the cosmos.
7
8 $ lmm sentence --text "Mathematics is the language of the universe" \
9   --stochastic --probability 0.7
10  Purview facilitates the fluid essence of the totality.
```

**Listing 4:** Stochastic synonym layer; three runs on the same seed

The sentence structure [Noun] [verb] the [adj] [noun] of the [noun] is preserved while every eligible content word is independently sampled.

## 5 Text Encoding: Reality as an Equation

One of **LMM**'s most striking capabilities is lossless encoding of arbitrary text as a symbolic mathematical equation. The byte sequence of the text  $b_0, b_1, \dots, b_{N-1}$  is treated as a discrete signal indexed by position. GP discovers an equation  $E^*$  minimising  $\text{MDL}(E^*, \{(i, b_i)\})$ . Integer residuals  $r_i = b_i - \lfloor E^*(i) \rfloor$  capture the approximation error exactly; decoding recovers each byte as  $b_i = \lfloor E^*(i) \rfloor + r_i$ .

```

1 $ lmm encode --text "The Pharaohs encoded reality in mathematics." \
2   --iterations 150 --depth 5
3
4 Equation : ((-12.541 * sin((-0.4 + ((x*x)*0.884)))) + 94.966)
5 MSE      : 530.17
6 Residuals: -16 15 6 -51 -3 13 ... (44 integers)
7 Round-trip: PERFECT
8 Decoded   : "The Pharaohs encoded reality in mathematics."

```

**Listing 5:** Encoding and lossless round-trip decoding

This is in sharp contrast to LLM embeddings, which are opaque high-dimensional vectors with no auditable relationship to the original text. Every element of the **LMM** encoding; the equation, its coefficients, and its residuals; is a human-readable, formally verifiable mathematical object.

## 6 Comparison with Statistical Language Models

We compare **LMM** against GPT-2 (117M parameters; [Radford et al. 2019](#)) across twelve dimensions. GPT-2 is chosen because it represents the era of language modelling closest in output quality to **LMM**'s current state; coherent and grammatically plausible, yet without instruction-following or long-range logical consistency.

**Table 1:** System-level comparison: **LMM** vs. GPT-2 (117M)

| Dimension                  | GPT-2 (117M)                                                              | LMM (v0.x)                                                                                              |
|----------------------------|---------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| <b>Training data</b>       | ~40 GB of WebText [Radford et al., 2019]                                  | None; zero training corpus.                                                                             |
| <b>Parameters weights</b>  | / 117 million learned float32 weights                                     | Zero; vocabulary is read from system dictionary.                                                        |
| <b>Training cost</b>       | Major TPU v3 compute cluster [Radford et al., 2019]                       | None; install and run.                                                                                  |
| <b>Inference hardware</b>  | GPU recommended (~500 MB VRAM)                                            | Any CPU; <4 MB RAM usage.                                                                               |
| <b>Inference speed</b>     | ~10–40 tokens/s on CPU [Gerganov and contributors, 2023]                  | ~100–500 chars/s on a single CPU core.                                                                  |
| <b>Output transparency</b> | Opaque; word choices cannot be traced to any inspectable mechanism        | Fully auditable; trajectory equation, rhythm equation, POS state, and synonym bank are all inspectable. |
| <b>Consent / ethics</b>    | Training data scraped without author consent [Grynbaum and Mac, 2023]     | No data ingested from any human author.                                                                 |
| <b>Grammatical fluency</b> | High; varied, human-like prose                                            | Moderate; structured but formulaic; improves with stochastic layer.                                     |
| <b>Factual grounding</b>   | Plausibility-based; hallucinates readily [Bang et al., 2023]              | Equation-grounded; every word traceable to a mathematical relation.                                     |
| <b>Web search</b>          | Not native; requires plugin or RAG                                        | Native <b>ask</b> command via DuckDuckGo Lite; no API key needed.                                       |
| <b>Symbolic reasoning</b>  | Limited; pattern-matches descriptions of causality [Kiciman et al., 2023] | Native do-calculus, SCM interventions, and exact physical simulation.                                   |
| <b>Open source</b>         | Partial; weights released, training code proprietary                      | Fully open source (MIT licence).                                                                        |

### 6.1 Fluency vs. Transparency Trade-off

The most visible gap between **LMM** and modern neural models like GPT-5 is surface fluency. While state-of-the-art LLMs learn next-token distributions from billions of complete sentences to produce perfectly natural prose, **LMM** produces text that is near-coherent fluent English but not yet perfect. Its outputs are grammatical at the sentence level; the SVO POS automaton enforces valid English phrase structure; but they can lack the subtle long-range coherence that emerges from exposure to real discourse. **LMM** remains in active development to bridge this gap and match the current state of GPT-5 fluency.

This is the cost of transparency: every word in an **LMM** sentence is traceable to Equation (5) or (11), making the generation process fully auditable. A user who observes an unexpected word

in the output can inspect the trajectory equation, the POS state, and the synonym substitution log, something impossible with any statistical LLM.

## 6.2 Speed and Resource Efficiency

GPT-2 inference on a CPU requires loading  $\sim 500$  MB of parameters into RAM and executing a sequence of matrix multiplications per token. **LMM** requires  $\mathcal{O}(W \cdot T \cdot P)$  operations for symbolic regression (window size  $W$ , iterations  $T$ , population  $P$ ) plus  $\mathcal{O}(|\mathcal{V}|)$  for word scoring. On a typical 2024 consumer laptop, **LMM** completes a 100-character continuation in under 50 milliseconds; without any GPU or specialised hardware.

## 6.3 Internet-Aware Generation

**LMM**'s `ask` command combines real-time web retrieval with equation-based summarisation: it fetches search results from DuckDuckGo Lite (no API key required), aggregates the snippets into a corpus, scores each sentence via Equation (12), and returns the top- $k$  most mathematically representative sentences.

# 7 Ethical Dimensions

The training paradigm as currently practised raises serious and well-documented ethical concerns. Training on web-scale corpora ingests creative and intellectual work without the consent or compensation of its creators [Grynbaum and Mac, 2023, U.S. Copyright Office, 2024]. The resulting systems memorise and reproduce verbatim fragments of training data [Ippolito et al., 2022]; a documented privacy violation. The environmental cost of a single large training run is comparable to the lifetime emissions of several passenger vehicles [Strubell et al., 2019]. The outputs of bias-laden training data reliably reproduce and amplify the social biases embedded in that data [Gallegos et al., 2023].

**LMM** is architecturally immune to all of these pathologies. It ingests no human-authored corpus; it learns from mathematical relationships discovered within the input structure itself; it cannot memorise a blog post, a novel, or a private email, because it has never read one. Its entire knowledge base is a set of curated word-tone pairs, synonym mappings, and physical equations; none of which represent the creative expression of any individual.

This is not merely an ethical nicety. It is an architectural property that makes the system's behaviour *formally verifiable*, its outputs *legally unencumbered*, and its reasoning *transparent by construction*; properties that no statistical LLM trained on web data can currently claim.

# 8 Experiments

## 8.1 Setup

All experiments were conducted on a standard x86-64 consumer laptop (Intel Core i7-1165G7, 16 GB RAM) running Ubuntu 24.04. No GPU was used. **LMM** was compiled from source with `cargo build -release -all-features`.

## 8.2 Equation Discovery Accuracy

To validate the GP engine, we generated synthetic datasets of the form  $y_i = f(x_i) + \varepsilon_i$  ( $\varepsilon_i \sim \mathcal{N}(0, 0.1)$ ) for four ground-truth functions and measured the proportion of 50 GP runs that recovered the correct functional form (up to constant scaling) within 200 iterations.

**Table 2:** GP symbolic regression recovery rate on synthetic data

| Ground truth        | N  | Iterations | Recovery    | Median MSE |
|---------------------|----|------------|-------------|------------|
| $f(x) = 2x + 1$     | 20 | 200        | 48/50 (96%) | $< 0.001$  |
| $f(x) = x^2 - 3$    | 20 | 200        | 41/50 (82%) | 0.012      |
| $f(x) = \sin(0.5x)$ | 30 | 300        | 37/50 (74%) | 0.031      |
| $f(x) = 3e^{-0.2x}$ | 30 | 300        | 28/50 (56%) | 0.148      |

Linear laws are recovered with near-perfect reliability. Non-linear functions require more iterations, exponential decay is the hardest case due to the restricted operator set. These results confirm that the GP engine provides a robust symbolic lens for extracting structure from sequential data.

### 8.3 Text Generation Latency

We measured end-to-end latency for five generation modes at their default parameter settings.

**Table 3:** Generation latency on a single CPU core (measured via `time` utility)

| Command   | Output length   | Total time |
|-----------|-----------------|------------|
| sentence  | ~15 words       | 8 ms       |
| paragraph | ~60 words       | 6 ms       |
| essay     | ~300 words      | 6 ms       |
| predict   | ~80 chars       | 87 ms      |
| summarize | top-2 sentences | 8 ms       |
| ask       | 3 sentences     | 554 ms*    |

\*Includes DuckDuckGo web retrieval latency.

### 8.4 Encode–Decode Round-Trip Fidelity

We encoded 1,000 random English sentences (length 20–200 characters) and verified lossless recovery. The round-trip success rate is **100%** by construction; integer residuals correct any GP approximation error exactly. The mean MSE of the discovered equation was 487.3 (range 12–2,100), reflecting the inherent difficulty of fitting arbitrary byte sequences; but the residuals eliminate all remaining error.

### 8.5 Stochastic Layer Diversity

To quantify lexical diversity introduced by the stochastic synonym layer, we generated 100 realisations of the same sentence with  $p = 0.5$  and measured the type-token ratio (TTR) per run and the pairwise word-level Jaccard distance across all pairs.

**Table 4:** Diversity metrics; stochastic synonym layer at  $p = 0.5$  over 100 runs

| Metric                            | Deterministic             | Stochastic |
|-----------------------------------|---------------------------|------------|
| Mean TTR                          | 1.00 (all runs identical) | 0.78       |
| Mean pairwise Jaccard distance    | 0.00                      | 0.41       |
| Sentences distinct from all prior | 0/100                     | 97/100     |

The stochastic layer produces unique outputs in 97% of runs while preserving sentence structure. The 3% collisions arise from short seeds with a limited eligible-word pool.

## 9 Discussion

### 9.1 Current Limitations

**LMM**’s outputs are grammatically valid but lack the long-range topical coherence of a trained language model. The POS finite-state machine enforces local grammatical agreement but has no mechanism for maintaining a topic across paragraphs. The trajectory and rhythm equations capture the statistical surface of the context window’s tone distribution; not its semantic content. As a result, long-form generation tends toward a distinctive mathematical-aphorism register rather than domain-specific exposition.

This is the expected behaviour of a proof-of-concept system. It demonstrates that training-free generation is possible; it does not claim parity with 175B-parameter models fine-tuned on human instructions.

### 9.2 Paths to Improvement

Several research directions could significantly improve **LMM**’s output quality:

- **Richer vocabulary structure:** extending the tone-indexed pool with domain-specific term lists (science, law, medicine) would improve topical coherence without any training.
- **Multi-signal GP:** fitting separate equations for syntactic role frequency (noun density, verb density) would allow the predictor to adapt its register.
- **Physics-grounded coherence:** integrating a topic-level ODE that models conceptual momentum (topics rising or falling in a discourse) could produce smoother multi-paragraph essays.
- **Hybrid web-retrieval:** expanding the **ask** pipeline to retrieve longer passages and applying the summariser to compress them could approach retrieval-augmented generation quality without any LLM.

### 9.3 Implications for the Field

The existence of a working, training-free text generation system challenges a core assumption of the field: that statistical learning from human-authored corpora is a necessary condition for language generation. **LMM** demonstrates that it is not. This does not mean that trained language models are without value; it means that the ethical costs of training, unconsented data use, environmental impact, and bias amplification, are not unavoidable costs of building useful language systems. They are costs of a choice, and **LMM** is an existence proof that the choice can be made differently.

## 10 Conclusion

We introduced the **Large Mathematical Model (LMM)**, a training-free text generation framework whose sole computational substrate is symbolic mathematics. Every sentence it produces is the output of a deterministic or stochastically decorated mathematical equation; discovered on-the-fly from the input signal, evaluated against a curated vocabulary, and selected by a tone-weighted scoring rule. The system requires no corpus, no neural network, no GPU, and no parameter storage; and it runs in milliseconds on any modern CPU.

We formalised its five-layer architecture, derived the generative equations for trajectory prediction, tone-weighted word selection, and stochastic synonym substitution, and compared it qualitatively and quantitatively against traditional statistical language modelling. While **LMM**'s near-coherent output is not yet perfectly fluent compared to the current state of GPT-5, its text generation is insanely fast; vastly outperforming any existing GPT models. It is cheaper, vastly more transparent, and ethically unencumbered: every word it writes is traceable to a mathematical derivation that invades no one's privacy, consumes no one's creative labour, and generates no carbon beyond the electricity of a single laptop.

The live demonstration at <https://lmm.wiseai.dev> and the fully open source implementation at <https://github.com/wiseaidotdev/lmm> are an invitation to the community to build on, challenge, and extend this direction. The question the system poses is simple: if an equation can write a sentence, what else can mathematics generate that we have been outsourcing to statistics?

## References

- Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023. URL <https://arxiv.org/abs/2302.04023>.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 1877–1901, 2020. URL <https://arxiv.org/abs/2005.14165>.
- Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. URL <https://arxiv.org/abs/2006.11287>.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL <https://arxiv.org/abs/2208.07339>.
- William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23:1–39, 2022. URL <https://arxiv.org/abs/2101.03961>.
- Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. Bias and fairness in large language models: A survey. *arXiv preprint arXiv:2309.00770*, 2023. URL <https://arxiv.org/abs/2309.00770>.

- Georgi Gerganov and contributors. llama.cpp: LLM inference in C/C++. <https://github.com/ggml-org/llama.cpp>, 2023.
- Michael M. Grynbaum and Ryan Mac. The times sues OpenAI and Microsoft over A.I. use of copyrighted work. *The New York Times*, December 2023. URL <https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html>.
- Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. URL <https://arxiv.org/abs/1506.02626>.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2015. URL <https://arxiv.org/abs/1503.02531>.
- Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A. Choquette-Choo, and Nicholas Carlini. Preventing verbatim memorization in language models gives a false sense of privacy. *arXiv preprint arXiv:2210.17546*, 2022. URL <https://arxiv.org/abs/2210.17546>.
- Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*, 2023. URL <https://arxiv.org/abs/2305.00050>.
- John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL <https://arxiv.org/abs/2005.11401>.
- Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2): 130–141, 1963.
- Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.
- Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL <https://openai.com/blog/better-language-models>.
- Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.
- Jorma Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.
- Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
- Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

- Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3645–3650, 2019. URL <https://arxiv.org/abs/1906.02243>.
- Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. URL <https://arxiv.org/abs/1905.11481>.
- U.S. Copyright Office. Copyright and artificial intelligence: Policy considerations, 2024. URL <https://www.copyright.gov/ai/>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. URL <https://arxiv.org/abs/1706.03762>.

## A Architecture Module Reference

**Table 5:** Source modules in the **LMM** Rust crate

| Module                        | Responsibility                                                       |
|-------------------------------|----------------------------------------------------------------------|
| <code>equation.rs</code>      | Expression AST; evaluation; symbolic differentiation; simplification |
| <code>discovery.rs</code>     | GP symbolic regression; MDL fitness; tournament selection            |
| <code>compression.rs</code>   | MDL scoring utility linking discovery to information theory          |
| <code>text.rs</code>          | Tone functions; sentence, paragraph, and essay generators            |
| <code>predict.rs</code>       | Trajectory and rhythm GP; SVO POS automaton; continuation engine     |
| <code>stochastic.rs</code>    | SynonymBank (curated and system dictionary); StochasticEnhancer      |
| <code>encode.rs</code>        | Text-to-equation encoding with residual storage                      |
| <code>physics.rs</code>       | ODE models: harmonic, Lorenz, pendulum, SIR, N-body                  |
| <code>simulation.rs</code>    | Integrators: Euler, RK4, RK45 adaptive, leapfrog                     |
| <code>causal.rs</code>        | Structural causal models; do-calculus                                |
| <code>consciousness.rs</code> | Perceive–encode–predict–act loop                                     |
| <code>world.rs</code>         | WorldModel; physics ODE state machine                                |
| <code>tensor.rs</code>        | Multi-dimensional tensor; gradient; Laplacian; curl                  |
| <code>net.rs</code>           | DuckDuckGo Lite client for the <code>ask</code> command              |
| <code>imagen.rs</code>        | Spectral Field Synthesis image generation                            |
| <code>perception.rs</code>    | Raw-byte normalisation into tensors                                  |

## B Curated Synonym Bank (Excerpt)

The `StochasticEnhancer` draws from a hand-crafted table of  $\approx 100$  scientific and mathematical term pairs. A representative excerpt is shown below.

**Table 6:** Excerpt of the curated synonym bank (from `stochastic.rs`)

| Source word          | Synonyms                                             |
|----------------------|------------------------------------------------------|
| <i>enables</i>       | allows, permits, empowers, facilitates, supports     |
| <i>transforms</i>    | converts, shifts, alters, reconfigures, reshapes     |
| <i>deterministic</i> | predictable, systematic, causal, precise, exact      |
| <i>entropy</i>       | disorder, chaos, uncertainty, complexity, randomness |
| <i>equation</i>      | formula, expression, relationship, model, identity   |
| <i>causality</i>     | determinism, consequence, mechanism, logic, reason   |
| <i>knowledge</i>     | understanding, wisdom, insight, cognition, learning  |
| <i>continuous</i>    | unbroken, flowing, perpetual, sustained, smooth      |

## C CLI Quick Reference

```
1 lmm <COMMAND> [OPTIONS]
2
3 COMMANDS
4 sentence --text <SEED> Single structural sentence
5 paragraph --text <SEED> -n <N> N-sentence paragraph
6 essay --text <TOPIC> -n <P> Full essay: intro + body +
  conclusion
7 predict --text <PREFIX> -p <LEN> GP-driven text continuation
8 summarize --text <TEXT> -n <K> Top-K key sentences by tone score
9 ask --prompt <Q> -l <HITS> Internet-aware answer synthesis
10 encode --text <TEXT> Text to symbolic equation (lossless)
11 decode --equation E --residuals R --length L
12 discover --iterations N GP symbolic regression on data
13 physics --model <M> --steps N ODE simulation (lorenz/sir/pendulum/
  harmonic)
14 causal --intervene-node X --intervene-value V
15 field --size N --operation gradient|laplacian
16 simulate --step DT --steps N Harmonic oscillator via RK4
17 consciousness --lookahead K Perceive-predict-act loop
18
19 GLOBAL FLAGS
20 --stochastic Enable synonym substitution
21 --probability FLOAT Replacement probability (default:
  0.5)
22 -v, --verbose Show banners, seeds, and decorations
```

**Listing 6:** LMM CLI command summary