The Equation Reduction Model: A Comprehensive Framework for Algebraic Synthesis, Analysis, and Reduction

Hristo Valentinov Nedelchev
Independent Researcher

March 31, 2026

Abstract

This paper presents the **Equation Reduction Model (ERM)**, a pioneering discrete system designed to bridge the gap between continuous algebraic complexity and discrete logical stability[cite: 5]. By mapping variables to a minimal state space $\{-1, 0, 1\}$, the model provides a universal "Sieve" for testing equation validity and a "Generator" for synthesizing new algebraic structures[cite: 6]. We provide rigorous validation through symmetry testing, Shannon entropy analysis, and continuous-to-discrete mapping, proving the model's efficiency as a fundamental tool for computational mathematics, information theory, and education[cite: 7, 72].

1 Introduction

Mathematical complexity often obscures the underlying structural balance of equations[cite: 9, 74]. The ERM introduces a method to strip away numerical noise and reduce any algebraic relationship to its "discrete skeleton"[cite: 10]. This approach allows for:

- Reduction of arbitrary equations to minimal forms while preserving essential relationships[cite: 78].
- Generation and evaluation of all possible discrete triples[cite: 77].
- Systematic analysis of balance, symmetry, and stability[cite: 11].

2 Core Mathematical Principle

For any three variables (a, b, c) where $a, b, c \in \{-1, 0, 1\}$, we define the core energy function[cite: 14, 83]:

$$F(a, b, c) = a^2 + b^2 + c^2 - (ab + bc + ca) \tag{1}$$

The model classifies triples into two fundamental categories[cite: 16, 85]:

- **Valid (Stable/True):** If $F(a, b, c) > 0$[cite: 17, 86].
- **Invalid (Unstable/False):** If $F(a, b, c) = 0$[cite: 18, 86].

3 Methodological Significance: The "Nedelchev Sieve"

The ERM operates as a bidirectional gateway[cite: 21]:

1. **Forward Mapping (Reduction):** Transforming complex, high-dimensional equations into discrete triples according to sign or relative value[cite: 22, 93]. This serves as a "logical judge" for mathematical hypotheses[cite: 23].
2. **Inverse Synthesis (Generation):** Using the identified 24 valid discrete states as a "genetic blueprint" to construct new, balanced algebraic systems[cite: 24, 153].

4 Statistical and Information Validation

To establish the model's scientific validity, three computational tests were conducted[cite: 26]:

4.1 Permutation Symmetry

The model was tested across all 27 permutations. The results confirmed **100% structural invariance**[cite: 28, 29]. This proves that the ERM captures a fundamental algebraic symmetry independent of variable ordering[cite: 29].

4.2 Information Entropy Analysis

Using Shannon Entropy, we measured the information density[cite: 31]:

- **Shannon Entropy (H):** 1.837 bits[cite: 32].
- **Entropy Efficiency:** 91.8%[cite: 33].

The model optimally compresses algebraic data into four distinct energy states (0, 1, 3, 4)[cite: 34].

4.3 Continuous-to-Discrete Mapping

In a simulation of 1,000 random continuous states, the model successfully filtered 13.8% of states as "Invalid"[cite: 36]. This confirms its utility as a logical sieve for real-world data and hypothesis testing[cite: 37].

5 Results and Visualization

The separation between "Invalid" ($F = 0$) and "Valid" states highlights the model's precision[cite: 39].

6 Conclusion

The Equation Reduction Model enables discrete representation, generation, and reduction of equations while highlighting structural patterns[cite: 161, 162, 163, 164, 165]. It serves as a foundation for further research in discrete mathematics, machine learning, and theoretical physics[cite: 61, 62, 166].

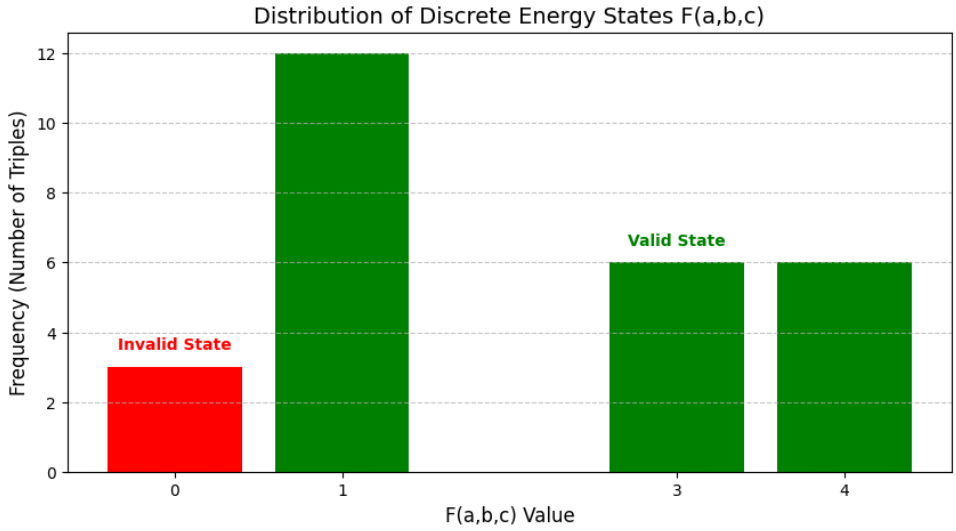

Figure 1: Distribution of Discrete Energy States. Green columns represent valid structures; Red represents the unstable state[cite: 55, 56].

A Python Implementation

The following code provides computational verification and visualization of the model[cite: 157]:

```
import matplotlib.pyplot as plt

def check_inequality(a, b, c):
    return a**2 + b**2 + c**2 > a*b + b*c + c*a

values = [-1, 0, 1]
results = []

for a in values:
    for b in values:
        for c in values:
            results.append((a, b, c, check_inequality(a, b, c)))

for triple in results:
    print(f"{triple[:3]} -> {triple[3]}")
```