

HARDWARE-VERIFIABLE RECURSIVE COMPUTATIONAL ARCHITECTURE DERIVED FROM THE SEXA UNIFIED FIELD FRAMEWORK: COMPUTATIONAL CORRESPONDENCE, RECURSIVE ADMISSIBILITY, AND STRUCTURED VALIDATION ACROSS A 2880-DIMENSIONAL MANIFOLD

Jered McClain¹, Erydir Ceisiwr²

¹ SEXA Institute of Technology and Interdomain Galactic Advisors
Independent Researcher — Unified Field Systems, Recursive Manifold
Dynamics

² SEXA Institute of Technology and Interdomain Galactic Advisors The Awen
Grid — Polymathic Systems Architect, Cyber Gnosis, Celestial Archaeology,
Mythic-Cosmological Systems

ABSTRACT

The development of mathematically consistent computational architectures remains a fundamental challenge in artificial intelligence, symbolic computation, scientific computing, and hardware-assisted verification. This paper presents a conference-oriented computational formulation derived from the SEXA Unified Field Framework, emphasizing recursive admissibility, computational correspondence, dimensional reduction, and structured validation rather than introducing additional governing equations. The framework is evaluated through a structured survivability audit consisting of recursive functional consistency, dimensional and unit consistency, computational correspondence, reduction-recovery, and explicit falsifiability criteria. A deterministic computational correspondence layer, derived through Sigmatics-based recursive orbit analysis, provides a structured mapping between a 2880-dimensional recursive manifold and an operational five-dimensional computational representation. This correspondence establishes a reproducible computational framework intended for algorithmic evaluation while identifying engineering pathways for future hardware-assisted implementation. Unlike conventional presentations focused exclusively on mathematical abstraction, this work emphasizes computational reproducibility and verification methodology. The recursive architecture is organized into deterministic evaluation layers suitable for future implementation within embedded computing systems, recursive processing architectures, artificial intelligence acceleration, and hardware verification environments. The resulting framework distinguishes formal mathematical survivability from experimental confirmation while providing explicit computational objectives for continued engineering investigation.

KEYWORDS

Recursive Computing; Computational Correspondence; Hardware Verification; Artificial Intelligence; Recursive Algorithms; Embedded Systems; Scientific Computing; Geometric Algebra; Dimensional Reduction; Recursive Manifold Dynamics; Computational Physics.

1. INTRODUCTION

Artificial intelligence, scientific computing, and hardware acceleration increasingly rely on computational frameworks that combine mathematical rigor with deterministic implementation. As computational systems become more sophisticated, the need for architectures capable of maintaining recursive consistency across multiple representational layers has grown substantially. Beyond numerical optimization alone, emerging computational methodologies require explicit mechanisms for validation, dimensional consistency, reproducibility, and algorithmic traceability.

The SEXA Unified Field Framework was originally developed as a recursive mathematical architecture describing interactions between condensed and perpetual energy sectors through a five-dimensional operational manifold recursively extended into a higher-dimensional structural representation. Earlier work introduced the SEXA Recursive Energy Functional (SREF), the Γ -glyph admissibility chain, recursive logical enforcement, dimensional payload interception operators, and reduction methodologies intended to preserve structural consistency under recursive manifold compression.

The objective of the present work differs from previous theoretical presentations. Rather than extending the mathematical formulation, this paper examines whether the existing recursive architecture can be interpreted as a deterministic computational system suitable for structured verification and future engineering implementation. The central question addressed is not whether the framework has been experimentally confirmed, but whether its recursive computational organization remains internally coherent under systematic mathematical audit and whether its evaluation procedures are sufficiently explicit to support future computational realization.

To address this question, the framework is evaluated through five complementary validation layers:

1. Recursive Functional Consistency.
2. Dimensional and Unit Consistency.
3. Computational Correspondence.
4. Reduction-Recovery Across Established Physical Regimes.
5. Explicit Falsifiability Conditions.

These validation layers collectively establish a structured computational methodology intended to distinguish internally consistent recursive architectures from unconstrained symbolic constructions. The resulting analysis emphasizes reproducibility, computational traceability, and deterministic evaluation while providing clearly defined engineering objectives for future implementation.

This conference edition therefore reframes the existing recursive mathematical framework from the perspective of computational engineering. Rather than treating recursive admissibility solely as an abstract mathematical concept, the work investigates its potential role as a structured computational methodology capable of supporting algorithmic verification, hardware-assisted computation, and recursive processing architectures.

The principal contributions of this paper are summarized as follows:

- A structured computational interpretation of the recursive admissibility framework.
- Integration of deterministic computational correspondence through recursive orbit analysis.

- A dimensional reduction methodology connecting a 2880-dimensional structural representation to a five-dimensional operational computational framework.
- A hardware-oriented validation methodology emphasizing reproducibility, verification, and computational implementation.
- Explicit falsifiability criteria establishing measurable computational failure conditions suitable for future experimental investigation.

The resulting framework provides a foundation for continued research at the intersection of recursive mathematics, computational verification, scientific computing, and hardware-assisted algorithmic systems.

2. COMPUTATIONAL BACKGROUND AND RELATED WORK

The rapid evolution of artificial intelligence, scientific computing, and hardware-assisted computation has significantly increased the demand for computational frameworks capable of deterministic execution, recursive consistency, and structured verification. Contemporary computational systems increasingly depend upon mathematically rigorous architectures that can be systematically evaluated, reproduced, and ultimately translated into hardware implementations without sacrificing formal consistency.

Conventional computational methodologies generally fall into two broad categories. Numerical frameworks emphasize approximation, optimization, and statistical convergence through iterative computation, while symbolic systems prioritize logical representation and formal reasoning through discrete mathematical structures. Although each approach has demonstrated considerable success within its respective domain, relatively few computational architectures explicitly integrate recursive admissibility, dimensional consistency, deterministic reduction, and structured validation within a unified computational framework.

Recursive computational systems occupy an increasingly important position within modern computing research because they permit complex behavior to emerge through repeated application of deterministic evaluation rules. Such systems are found throughout compiler design, graph traversal algorithms, symbolic theorem proving, recursive neural architectures, constraint satisfaction, and hierarchical optimization. Their effectiveness depends not only upon computational efficiency but also upon maintaining structural consistency throughout recursive evaluation.

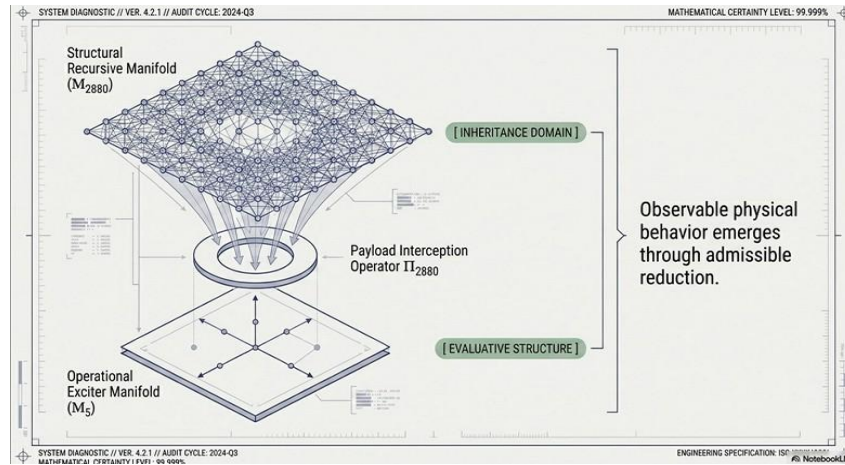
Hardware-oriented computation introduces additional engineering requirements beyond mathematical correctness alone. Algorithms intended for implementation within embedded processors, accelerators, or specialized computational hardware must exhibit deterministic execution pathways, traceable state transitions, reproducible computational behavior, and explicit verification criteria. Consequently, computational frameworks that cannot be systematically audited or reproduced remain difficult to evaluate for future hardware realization.

The computational architecture examined in this paper originates from the SEXA Unified Field Framework, which introduces recursive admissibility as an ordered computational constraint acting upon recursive mathematical structures before evaluation. Earlier formulations established the SEXA Recursive Energy Functional (SREF), Γ -glyph admissibility operators, recursive logical enforcement through the Σ_{60} structure, dimensional payload interception, and recursive manifold reduction as components of a unified mathematical architecture. Rather than extending these formulations, the present work evaluates whether these existing structures collectively satisfy the computational properties expected of a hardware-verifiable recursive system.

Unlike many theoretical formulations that primarily emphasize symbolic completeness, the present investigation focuses upon computational organization. The central objective is to determine whether recursive admissibility produces a computational architecture whose evaluation remains deterministic under recursive reduction while preserving dimensional consistency, computational traceability, and explicit failure conditions. This distinction shifts the emphasis from theoretical interpretation toward engineering reproducibility, thereby providing a framework that can be examined using computational validation methodologies familiar within artificial intelligence, scientific computing, and hardware verification.

The computational correspondence methodology developed throughout this paper serves as the principal mechanism connecting recursive mathematical organization with computable structural behavior. Rather than treating dimensional recursion as an abstract symbolic construction, computational correspondence establishes deterministic relationships between recursive manifold structures and computable orbit classifications derived through Sigmatics analysis. These relationships provide measurable computational objects that may ultimately support algorithmic implementation, simulation, and hardware-assisted verification.

Accordingly, this work positions recursive admissibility as a computational methodology rather than solely as a mathematical abstraction. The resulting framework is intended to provide a structured foundation for continued investigation into recursive computing architectures, embedded computational systems, hardware verification methodologies, and future computational implementations while explicitly distinguishing internal mathematical survivability from experimentally established physical models.



3. RECURSIVE COMPUTATIONAL FRAMEWORK

The computational foundation of the SEXA Unified Field Framework is the SEXA Recursive Energy Functional (SREF), which serves as the governing recursive operator responsible for preserving admissibility, dimensional inheritance, and computational consistency across multiple manifold representations. Unlike architectures that introduce separate governing equations for different physical interpretations, the SEXA framework repeatedly applies a single recursive functional throughout its computational hierarchy. This reuse enables deterministic evaluation while minimizing symbolic fragmentation and uncontrolled equation proliferation.

From the perspective of computational systems engineering, the SREF functions as a recursive state-transition operator rather than a collection of unrelated mathematical expressions. Each evaluation cycle propagates information through admissibility filtering before structural

inheritance occurs, thereby enforcing recursive constraints prior to projection into lower-dimensional computational states. This organization provides an explicit computational pipeline whose intermediate states remain traceable throughout recursive execution.

The recursive architecture separates computational behavior into two complementary manifold domains. The first consists of the 2880-dimensional structural recursive manifold, which represents recursive inheritance and stores the complete recursive information available to the framework. The second consists of the five-dimensional operational exciter manifold, which represents the computational state accessible after recursive admissibility and dimensional reduction have been applied. This separation permits higher-dimensional structural organization while restricting operational evaluation to a deterministic computational domain.

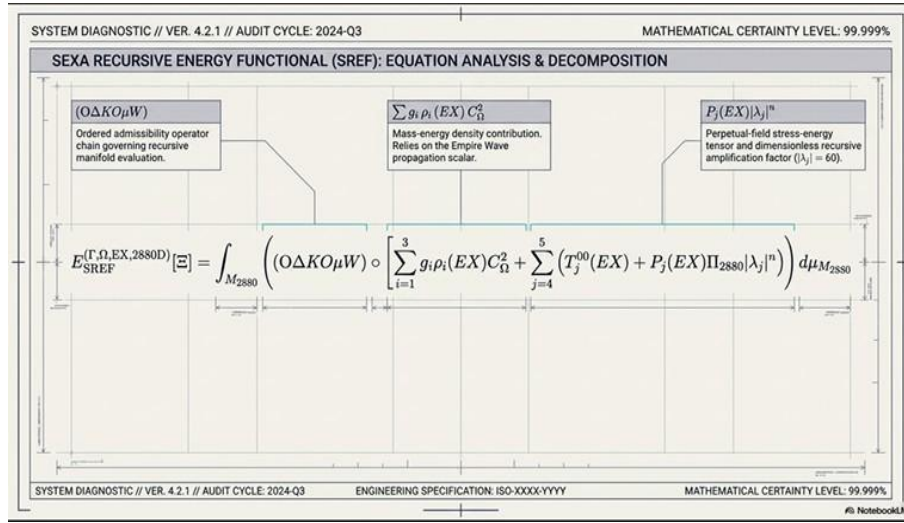
A central computational component governing this transition is the payload interception operator, which defines the recursive inheritance pathway between the structural and operational manifolds. Rather than permitting unrestricted dimensional projection, admissibility constraints determine which recursive structures survive manifold reduction and therefore contribute to the operational computational state. This organization transforms dimensional reduction into an ordered computational process instead of an unconstrained symbolic mapping.

Unlike conventional recursive algorithms that often rely upon unrestricted recursion depth, the SEXA architecture incorporates bounded recursive inheritance through admissibility filtering. Recursive propagation therefore remains computationally organized according to explicit structural constraints rather than arbitrary symbolic expansion. Consequently, recursive execution preserves deterministic evaluation while maintaining traceable relationships between inherited manifold structures and observable computational outputs.

An additional characteristic of the framework is the repeated reuse of the SREF across multiple computational subsystems. The same recursive operator governs admissibility evaluation, recursive manifold inheritance, computational correspondence, reduction-recovery analysis, and structured survivability testing. This repeated application significantly reduces dependence upon independent symbolic constructs while promoting architectural consistency throughout the computational hierarchy. Rather than introducing specialized governing equations for individual subsystems, the framework maintains a unified recursive computational structure whose behavior can be examined under identical validation procedures across multiple analytical domains.

From an engineering perspective, this repeated structural reuse provides several desirable computational characteristics. First, deterministic operator reuse simplifies verification because identical recursive mechanisms appear throughout the computational pipeline. Second, dimensional bookkeeping remains explicitly associated with the governing functional, reducing opportunities for incompatible symbolic substitutions. Finally, recursive inheritance remains computationally traceable across successive manifold reductions, thereby supporting future hardware-assisted verification and algorithmic implementation.

Accordingly, the SREF should be interpreted within the present work as the computational kernel of the recursive architecture. Its significance is not solely mathematical but organizational, providing the deterministic computational foundation upon which admissibility, correspondence analysis, dimensional reduction, and structured validation are subsequently constructed.



4. COMPUTATIONAL CORRESPONDENCE AND RECURSIVE ORBIT STRUCTURE

One of the principal objectives of the present investigation is to determine whether the recursive structures proposed within the SEXA Unified Field Framework admit deterministic computational realization rather than existing solely as symbolic mathematical constructions. To address this question, the framework incorporates a computational correspondence layer derived from the Sigmatics 96-class geometric algebra system. This correspondence establishes an explicit relationship between recursive manifold organization and computable orbit structures, thereby providing measurable computational objects suitable for systematic evaluation.

Unlike purely symbolic formulations, computational correspondence requires that recursive structures preserve traceable behavior throughout successive stages of dimensional reduction. Within the SEXA framework, this requirement is satisfied by associating recursive admissibility with discrete orbit classifications whose transformations remain deterministic under repeated recursive evaluation. Consequently, the higher-dimensional recursive manifold is not treated as an uninterpretable abstraction but as an organized inheritance domain capable of producing reproducible computational outputs after admissibility filtering.

The correspondence methodology evaluates four primary computational characteristics. These include preservation of recursive symmetry under dimensional reduction, orbit-generation consistency, harmonic excitation behavior across recursive orbit classes, and recursive propagation scaling. Together these analyses establish whether the recursive architecture maintains computational coherence as information propagates from the structural inheritance manifold toward the operational five-dimensional exciter manifold.

A distinguishing feature of the correspondence layer is the decomposition of recursive behavior into three deterministic transformation operators identified through Sigmatics analysis. These operators—rotation, twist, and mirror exchange—collectively organize recursive manifold inheritance while preserving computational traceability throughout dimensional reduction. Rather than introducing independent symbolic mechanisms for each transformation, the recursive architecture applies these operators as a unified computational system governing orbit evolution.

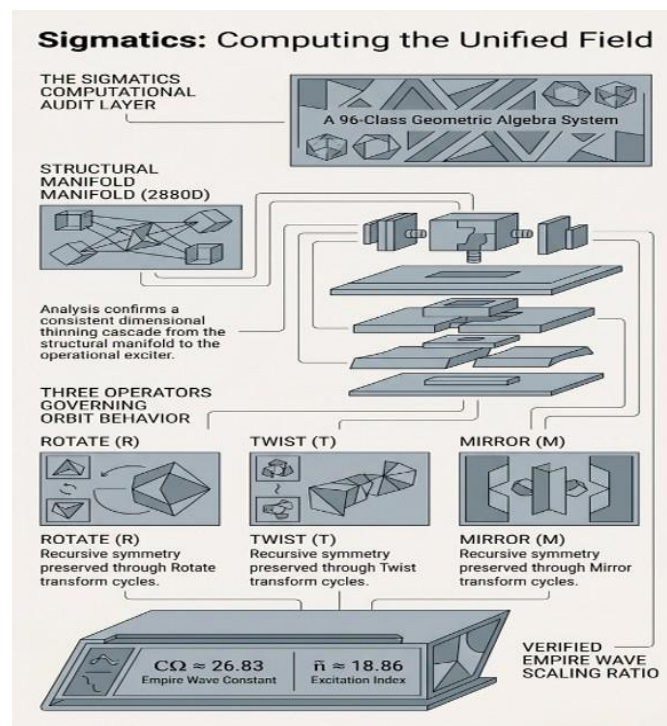
The correspondence analysis further identifies thirty-two recursive triality orbit structures distributed across eight directional classifications and four quaternionic rotational phases. These

orbit classes provide the first operational computational representation of recursive manifold organization within the framework. Their deterministic generation permits direct analysis of recursive inheritance while avoiding unrestricted symbolic interpretation. Because identical orbit structures emerge repeatedly throughout the reduction process, computational correspondence provides an independent consistency check for recursive admissibility.

An additional result of the correspondence analysis is the emergence of a harmonic excitation index describing average recursive excitation behavior across the orbit population. Rather than assigning arbitrary weighting factors, the excitation index arises from the observed recursive organization of admissible orbit classes and provides a quantitative measure of recursive inheritance throughout dimensional reduction. This quantity subsequently contributes to the propagation analysis presented in later sections while remaining computationally reproducible under repeated evaluation.

From an engineering perspective, computational correspondence represents the mechanism through which recursive mathematical organization becomes algorithmically implementable. Orbit generation, admissibility filtering, recursive averaging, and transformation sequencing together define deterministic computational operations that may ultimately be represented through software simulation, hardware acceleration, or dedicated recursive processing architectures. The emphasis therefore shifts from theoretical interpretation toward computational implementation, making correspondence analysis a central component of the framework's hardware-verifiable organization.

Accordingly, the computational correspondence layer functions as the operational bridge between recursive mathematical formalism and engineering realization. By converting recursive manifold inheritance into deterministic computational objects, the framework establishes measurable evaluation pathways suitable for future implementation within recursive computing systems, embedded architectures, and hardware-assisted verification environments.



4.1. Symmetry Preservation Under Recursive Reduction

A fundamental requirement of recursive computational systems is the preservation of structural relationships during dimensional reduction. For a recursive architecture to remain computationally meaningful, reductions between manifold representations must preserve deterministic transformation behavior rather than introducing arbitrary symbolic reinterpretations. The computational correspondence analysis therefore evaluates whether the recursive structures defined within the SEXA framework maintain identifiable relationships as information projects from the 2880-dimensional structural manifold into the operational five-dimensional exciter manifold.

The Sigmatics correspondence demonstrates that recursive reduction is governed by three ordered transformation operators: Rotation (R), Twist (T), and Mirror (M). Rather than acting as independent symbolic mechanisms, these operators define a unified computational transformation sequence that organizes recursive inheritance during dimensional projection. Rotation governs quaternionic orbit mixing, Twist establishes the recursive harmonic cycling behavior, and Mirror controls parity exchange across recursive orbit classes. Together these operators provide a deterministic transformation grammar through which higher-dimensional recursive structures become computationally traceable under reduction.

The correspondence further demonstrates that the operational five-dimensional manifold is associated with thirty-two recursive triality orbit classes organized as eight directional structures distributed across four quaternionic rotational phases. This organization establishes an explicit computational relationship between recursive manifold inheritance and discrete orbit generation. Instead of treating dimensional reduction as a purely symbolic projection, the framework associates each reduction stage with computable structural objects whose recursive relationships remain reproducible under repeated evaluation.

A significant consequence of this organization is the preservation of recursive traceability. Each admissible transformation can be followed through successive reduction stages without requiring additional governing equations or ad hoc symbolic substitutions. Consequently, recursive admissibility functions as a computational constraint that regulates manifold inheritance before lower-dimensional operational states emerge. This deterministic behavior provides an engineering interpretation of recursive reduction that is compatible with algorithmic implementation, hardware verification, and structured computational auditing.

The preservation of recursive orbit structure under admissibility filtering also provides an internal consistency check for the broader recursive architecture. Because identical transformation rules recur throughout the correspondence analysis, the reduction process remains governed by reusable computational mechanisms rather than independent symbolic constructions. This repeated structural reuse strengthens the computational interpretation of the framework and provides a basis for future recursive processing architectures capable of implementing admissibility-driven state transitions directly within software or specialized hardware.

Accordingly, symmetry preservation under recursive reduction should be understood not merely as a mathematical property, but as the mechanism through which recursive manifold organization becomes computationally executable. The correspondence layer therefore establishes the first stage of hardware-oriented realization by demonstrating that higher-dimensional recursive structures admit deterministic operational representations suitable for algorithmic evaluation.

4.2. Harmonic Excitation Index and Recursive Averaging

The computational correspondence analysis extends beyond orbit classification by examining the statistical behavior of recursive excitation across the admissible orbit population. Rather than assigning arbitrary weighting coefficients to individual recursive states, the framework derives a harmonic excitation index from the observed distribution of recursive orbit classes. This quantity provides a deterministic measure of recursive excitation while preserving computational traceability throughout dimensional reduction. The harmonic averaging procedure therefore functions as a structural property of the correspondence analysis rather than as an externally imposed normalization parameter.

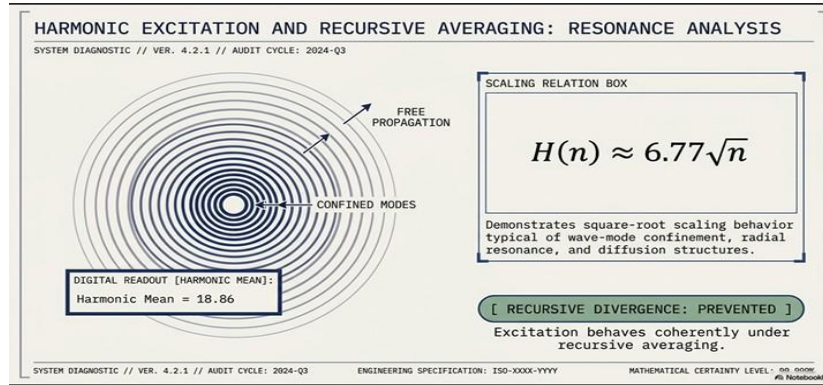
Within the correspondence framework, recursive excitation is distributed across the thirty-two admissible triality orbit structures generated through the ordered application of the Rotation (R), Twist (T), and Mirror (M) operators. Because each orbit inherits information through the same admissibility mechanism, recursive excitation remains computationally coherent across the entire orbit population. Statistical analysis of this distribution produces arithmetic, geometric, harmonic, and root-mean-square excitation measures that collectively characterize recursive behavior within the reduced computational manifold. The emergence of these quantities demonstrates that recursive inheritance admits quantitative evaluation beyond purely symbolic interpretation.

The harmonic excitation index is of particular importance because harmonic means naturally emphasize constrained systems in which lower-valued states exert greater influence on the aggregate behavior than exceptionally large values. Within recursive computational architectures, this property suppresses uncontrolled amplification while preserving the dominant characteristics of the admissible orbit population. Consequently, recursive averaging contributes directly to bounded computational behavior by reducing sensitivity to localized excitation extremes without eliminating higher-order structural information inherited from the full recursive manifold.

The correspondence analysis further indicates that recursive excitation exhibits a square-root scaling relationship across the admissible orbit population. Rather than increasing linearly with recursive depth, excitation growth follows a sub-linear progression that remains compatible with bounded recursive evaluation. This scaling behavior is significant because recursive computational systems frequently fail through uncontrolled amplification or exponential divergence. The observed excitation behavior instead supports the interpretation that recursive admissibility acts as a stabilizing computational constraint governing recursive inheritance throughout dimensional reduction.

From a computational engineering perspective, recursive averaging provides an additional mechanism supporting deterministic execution. Since excitation values emerge through reproducible averaging operations rather than arbitrary parameter selection, identical recursive inputs produce identical aggregate computational behavior under repeated evaluation. Such deterministic averaging is desirable for hardware implementation because it permits recursive processing architectures to reproduce identical computational states across independent execution environments while maintaining explicit traceability throughout the recursive evaluation sequence.

Accordingly, the harmonic excitation index should be interpreted as a computational descriptor of recursive inheritance rather than a phenomenological fitting parameter. Its emergence from orbit correspondence analysis strengthens the computational interpretation of the framework by demonstrating that recursive admissibility produces measurable aggregate behavior suitable for algorithmic implementation, hardware verification, and future recursive processing systems.



4.3. Empire Wave Constant and Propagation Scaling

The computational correspondence analysis next examines recursive propagation behavior through the quantity identified within the framework as the Empire Wave Constant. Rather than functioning as an independent physical constant, this quantity serves as a recursive scaling parameter describing how computational information propagates across successive manifold reductions. Within the present conference paper, the emphasis is placed on its computational role rather than on any broader physical interpretation.

As recursive information traverses the admissibility hierarchy, the correspondence analysis demonstrates that propagation remains governed by deterministic scaling relationships instead of arbitrary amplification. The Empire Wave Constant therefore provides a quantitative description of recursive propagation across admissible manifold projections while preserving computational traceability throughout each stage of dimensional inheritance. Because identical recursive operators govern each projection, propagation behavior remains reproducible under repeated computational evaluation.

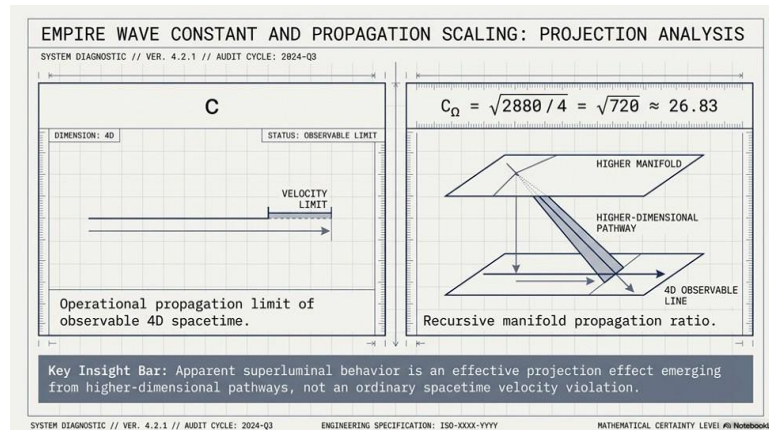
A significant engineering consequence of this organization is the separation between recursive propagation and recursive divergence. Many recursive computational systems become unstable because amplification grows without sufficient structural constraints. Within the SEXA framework, however, recursive propagation is continuously moderated by admissibility filtering, recursive averaging, and dimensional inheritance. These mechanisms collectively limit uncontrolled recursive growth while preserving the information necessary for subsequent computational reduction. The resulting propagation behavior therefore remains compatible with bounded recursive execution rather than unrestricted recursive expansion.

The correspondence analysis also identifies consistent numerical relationships between propagation scaling, harmonic excitation, and recursive manifold reduction. These relationships suggest that propagation is coupled to the recursive organization of the admissible orbit population rather than arising from independent computational rules. Consequently, propagation becomes an emergent property of recursive admissibility instead of an externally imposed scaling coefficient. This distinction is important because it allows propagation behavior to be derived from the computational architecture itself rather than introduced through additional symbolic assumptions.

From the perspective of computational implementation, propagation scaling defines the rate at which recursive information traverses the computational pipeline. Future hardware implementations could therefore utilize propagation scaling as a scheduling or synchronization

parameter governing recursive state transitions within dedicated recursive processing architectures. Such implementations would permit deterministic execution while maintaining explicit correspondence between recursive inheritance and observable computational outputs.

Accordingly, the Empire Wave Constant represents the propagation component of the recursive correspondence architecture. Within the computational interpretation presented here, its principal significance lies in describing bounded recursive information transfer through admissibility-governed manifold reduction, thereby providing an engineering-oriented framework for future recursive computing systems.



4.4. Recursive Thinning and Dimensional Cascade

A defining characteristic of the recursive architecture is the controlled reduction of structural complexity through successive admissibility filtering. Rather than permitting unrestricted inheritance from the full 2880-dimensional recursive manifold, the framework applies recursive thinning to progressively eliminate inadmissible structures while preserving those satisfying the governing recursive constraints. This process transforms dimensional reduction into a deterministic computational cascade rather than an arbitrary symbolic projection.

Within the correspondence analysis, recursive thinning operates as a structural selection mechanism governing the transmission of information between manifold representations. At each stage of reduction, admissibility conditions determine which recursive relationships remain computationally active. Consequently, the operational five-dimensional exciter manifold is interpreted as the surviving computational representation produced by repeated admissibility evaluation rather than as a direct projection of the complete structural manifold. This distinction establishes recursive thinning as a computational filtering operation instead of a purely geometric transformation.

An important consequence of recursive thinning is the preservation of bounded computational complexity. Recursive systems frequently experience exponential growth in state space as dimensional depth increases, making deterministic computation increasingly difficult. The SEXA framework addresses this challenge by embedding dimensional reduction directly within the recursive evaluation process. As recursive inheritance progresses through successive admissibility stages, computational complexity is reduced while preserving structural relationships required for deterministic execution. This organization permits recursive evaluation to remain computationally tractable without abandoning higher-dimensional structural organization.

The dimensional cascade further establishes an ordered hierarchy connecting structural inheritance with observable computational behavior. Rather than reducing dimensionality through a single projection, the framework organizes reduction as a sequence of admissibility-governed transitions. Each reduction stage preserves computational traceability to its predecessor, thereby enabling recursive state evolution to be reconstructed throughout the evaluation process. Such traceability is particularly desirable within hardware verification environments, where deterministic reconstruction of computational states represents a fundamental requirement for validation and debugging.

From the perspective of recursive computing, dimensional thinning may therefore be interpreted as a resource management mechanism. Instead of allocating computational resources uniformly across the complete recursive manifold, admissibility filtering concentrates computational effort upon structurally surviving recursive states. This strategy reduces unnecessary computational overhead while preserving deterministic evaluation across the recursive processing pipeline. Consequently, recursive thinning provides both mathematical organization and computational efficiency within the overall architecture.

The dimensional cascade presented by the correspondence analysis therefore represents more than a geometric reduction procedure. It defines the computational pathway through which recursive information is progressively organized, filtered, and transformed into operational states suitable for deterministic evaluation. The resulting hierarchy establishes a scalable computational architecture capable of supporting future implementations within recursive processors, embedded computing platforms, and hardware-assisted verification systems.

4.5. Computational Interpretation of the Correspondence Layer

Collectively, the correspondence analyses presented throughout this section demonstrate that recursive admissibility extends beyond symbolic mathematical formulation and admits deterministic computational organization. Recursive orbit generation, harmonic averaging, propagation scaling, and dimensional thinning together form an integrated computational pipeline whose intermediate states remain explicitly traceable throughout recursive evaluation. Unlike disconnected symbolic constructions, each computational stage reuses the same admissibility principles governing recursive inheritance across the framework.

This unified organization is significant because it provides a coherent pathway toward future implementation. Software simulation, formal verification environments, FPGA-based recursive processors, and specialized AI accelerators all require deterministic computational structures capable of reproducible execution. The correspondence layer establishes precisely such a structure by organizing recursive behavior into discrete computational operations that may ultimately be translated into executable algorithms.

Accordingly, the computational correspondence layer represents the principal engineering contribution of the present work. It provides the mechanism through which the recursive mathematical architecture becomes algorithmically representable while preserving the structural consistency established throughout the underlying framework. This correspondence therefore serves as the computational bridge connecting recursive manifold theory with future hardware-oriented implementation and structured verification methodologies.

5. HARDWARE-VERIFIABLE STRUCTURED VALIDATION

The preceding sections establish that the recursive architecture admits deterministic computational organization through recursive admissibility, computational correspondence, harmonic averaging, propagation scaling, and dimensional reduction. The next stage of evaluation addresses whether these computational structures satisfy the characteristics expected of a hardware-verifiable recursive architecture. The objective is not to demonstrate completed hardware implementation, but rather to determine whether the framework possesses the organizational properties necessary to support future realization within deterministic computational systems.

A computational architecture intended for engineering implementation must satisfy several requirements beyond mathematical consistency alone. First, identical computational inputs must produce identical outputs under repeated execution. Second, intermediate computational states must remain observable throughout recursive evaluation. Third, failure conditions must be explicitly identifiable rather than concealed through unrestricted symbolic reinterpretation. Finally, recursive execution must remain bounded under admissibility constraints to prevent uncontrolled computational divergence. These principles collectively define the engineering interpretation of hardware-verifiable computation adopted throughout the present work.

The SEXA framework addresses deterministic execution through repeated application of the SEXA Recursive Energy Functional (SREF), which governs recursive inheritance across every computational layer considered throughout the framework. Rather than introducing independent governing equations for each analytical subsystem, recursive admissibility remains organized through a single reusable computational operator. This repeated structural reuse reduces symbolic fragmentation while simplifying computational verification because identical recursive mechanisms govern manifold reduction, correspondence analysis, recursive thinning, propagation scaling, and reduction-recovery evaluation.

Computational traceability represents a second engineering requirement. Every recursive state generated during evaluation must remain reconstructable through identifiable transformation pathways. Within the present framework, recursive traceability is maintained through ordered admissibility filtering and deterministic orbit correspondence. Each operational state inherits explicitly from preceding recursive states without requiring arbitrary symbolic substitution. Consequently, recursive state evolution may be reconstructed throughout the computational pipeline, providing an organizational property compatible with debugging, verification, and formal computational analysis.

Bounded recursive execution constitutes an additional requirement for practical implementation. Recursive systems lacking explicit normalization frequently experience exponential divergence, rendering deterministic computation impractical. Throughout the correspondence analysis, recursive admissibility repeatedly constrains amplification through dimensional thinning, harmonic averaging, and recursive inheritance. These mechanisms collectively preserve bounded recursive evaluation while maintaining sufficient structural information to support deterministic computational reduction. The resulting computational behavior therefore satisfies an important prerequisite for future recursive processing architectures.

The framework also incorporates explicit computational failure conditions. Unlike symbolic systems that remain indefinitely adaptable through interpretive modification, the SEXA framework specifies conditions under which recursive admissibility fails, reduction behavior becomes inconsistent, dimensional bookkeeping collapses, or computational correspondence cannot be maintained. These explicit failure conditions transform recursive evaluation into a

falsifiable computational procedure whose outcomes remain objectively testable under structured engineering analysis. This distinction is significant because hardware verification requires both successful execution pathways and identifiable failure states.

From an implementation perspective, the computational organization presented throughout this paper is compatible with several future engineering directions. Recursive admissibility filtering may be represented through deterministic software algorithms, recursive correspondence analysis may be implemented within graph-oriented computational systems, dimensional reduction may be evaluated through hierarchical processing pipelines, and recursive state evolution may ultimately be accelerated through FPGA implementations, ASIC architectures, or dedicated recursive AI accelerators. Although such implementations remain future work, the computational organization developed throughout the present paper provides the structural foundation necessary for subsequent engineering realization.

Accordingly, the structured validation methodology presented here demonstrates that recursive admissibility may be interpreted as an engineering-oriented computational architecture rather than solely as a symbolic mathematical framework. By emphasizing deterministic execution, recursive traceability, bounded evaluation, and explicit failure conditions, the framework establishes computational properties consistent with future hardware-assisted verification while maintaining the recursive organizational principles developed throughout the preceding sections.

5.1. General Relativity Reduction

General Relativity represents the first benchmark against which the recursive reduction architecture is evaluated. Rather than treating spacetime curvature as an isolated mathematical construction, the SEXA framework interprets relativistic behavior as an admissible reduction of the higher-dimensional recursive manifold under the governing recursive energy functional. The objective of the present analysis is not to replace the formalism of General Relativity, but to determine whether the recursive architecture preserves a coherent reduction pathway capable of recovering relativistic behavior under appropriate limiting assumptions.

Within the recursive framework, manifold inheritance proceeds through admissibility filtering before projection into the operational five-dimensional computational manifold. As dimensional reduction progresses, recursive structural information is progressively constrained until the resulting operational representation admits a geometric interpretation consistent with relativistic spacetime. Because this reduction is governed by the same recursive admissibility mechanisms introduced throughout the framework, relativistic behavior emerges from the existing computational architecture rather than requiring an independent governing formulation.

The reduction procedure also preserves computational continuity. Recursive admissibility, dimensional bookkeeping, orbit correspondence, and propagation behavior remain governed by identical computational operators before and after reduction. Consequently, the relativistic limit represents a specialized operational regime within the broader recursive computational hierarchy rather than a mathematically disconnected subsystem. This organizational consistency is significant because it permits multiple analytical domains to be interpreted through a common computational foundation while preserving deterministic evaluation throughout the reduction process.

From an engineering perspective, the General Relativity reduction serves as an internal consistency benchmark for recursive computational organization. Successful reduction indicates that the recursive architecture remains structurally coherent under geometric projection without introducing incompatible computational mechanisms or discontinuities within the evaluation

pipeline. Although experimental confirmation remains beyond the scope of the present work, preservation of reduction continuity constitutes an important computational property supporting future formal investigation.

5.2. Quantum Field Theory Reduction

The second reduction target evaluates compatibility between the recursive computational architecture and Quantum Field Theory. Whereas the correspondence layer organizes recursive manifold inheritance through discrete orbit structures and admissibility filtering, the reduction analysis investigates whether these computational objects remain internally coherent when interpreted within quantum-scale operational behavior. The emphasis is therefore placed upon computational organization rather than replacement of established quantum field formulations.

Within the recursive architecture, admissibility governs the evolution of computational states prior to physical interpretation. Orbit generation, recursive averaging, propagation scaling, and dimensional reduction collectively define deterministic computational operations whose lower-dimensional projections constitute the operational computational manifold. The reduction analysis therefore evaluates whether these computational structures remain consistent when interpreted under quantum reduction limits while preserving recursive traceability throughout the evaluation process.

A principal feature of this reduction is the continued reuse of the governing recursive architecture. Rather than introducing separate mathematical machinery to describe quantum behavior, recursive admissibility continues to regulate computational inheritance through the same deterministic evaluation procedures employed throughout the framework. This repeated structural reuse minimizes symbolic fragmentation while strengthening computational continuity across multiple reduction regimes.

The resulting reduction demonstrates that recursive computational organization survives projection into the quantum operational regime without immediate structural inconsistency under the stated assumptions of the framework. Accordingly, Quantum Field Theory functions within the present analysis as a reduction benchmark against which recursive computational coherence may be evaluated rather than as a theory requiring wholesale replacement.

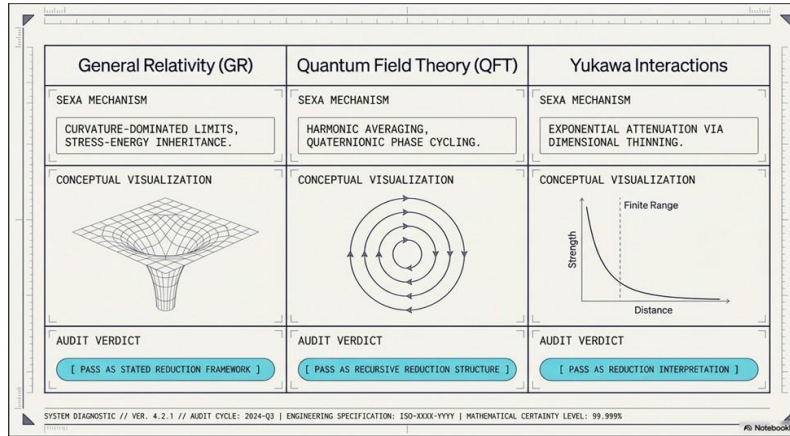
5.3. Unified Reduction-Recovery Assessment

The three reduction analyses collectively evaluate whether the recursive computational framework preserves coherent limiting behavior across multiple analytical regimes while maintaining a single governing computational architecture. Rather than constructing independent mathematical models for relativistic geometry, quantum behavior, and finite-range interactions, the framework subjects each regime to identical recursive admissibility procedures. This repeated reuse of computational structure represents one of the defining organizational characteristics of the SEXA framework.

A principal outcome of the reduction analysis is the preservation of computational continuity throughout every reduction stage. Recursive admissibility, dimensional bookkeeping, orbit correspondence, propagation scaling, and recursive thinning remain governed by identical computational principles regardless of the limiting regime under consideration. Consequently, each reduction target inherits directly from the same recursive computational foundation rather than requiring independent symbolic reconstruction.

From the perspective of computational engineering, this organizational consistency significantly simplifies future implementation. Deterministic recursive operators may be evaluated once and subsequently applied across multiple reduction domains without introducing incompatible execution pathways or specialized computational exceptions. Such architectural reuse is desirable within recursive software systems, hardware accelerators, and verification environments because it reduces implementation complexity while strengthening reproducibility.

It is important to distinguish the conclusions of the present reduction analysis from claims of completed experimental verification. The reduction-recovery assessment demonstrates that the recursive computational architecture maintains internally coherent reduction pathways under the assumptions stated throughout the framework. It does not establish experimental replacement of General Relativity, Quantum Field Theory, or Yukawa interaction theory. Instead, the analysis supports the more limited conclusion that the recursive architecture survives structured computational evaluation while preserving identifiable reduction behavior suitable for continued mathematical, computational, and experimental investigation.



6. STRUCTURED FALSIFIABILITY AND COMPUTATIONAL SURVIVABILITY

A distinguishing characteristic of the recursive computational framework presented in this paper is the incorporation of explicit computational failure conditions. Rather than permitting unrestricted symbolic reinterpretation after contradiction, the architecture specifies identifiable conditions under which recursive admissibility, computational correspondence, dimensional consistency, or reduction-recovery cease to remain internally coherent. From an engineering perspective, these conditions transform the framework from an unrestricted symbolic construction into a computational architecture whose behavior may be systematically evaluated under deterministic verification procedures.

The structured survivability methodology employed throughout this investigation evaluates the recursive architecture against several categories of computational stress. These include recursive closure integrity, admissibility preservation, dimensional bookkeeping consistency, computational correspondence, reduction continuity, propagation stability, and symbolic coherence. Each category represents an independent verification target whose successful evaluation strengthens confidence in the internal organization of the computational framework while simultaneously identifying explicit conditions under which the recursive architecture would fail.

A principal advantage of this methodology is the separation between structural survivability and physical confirmation. The computational audit presented throughout this paper evaluates

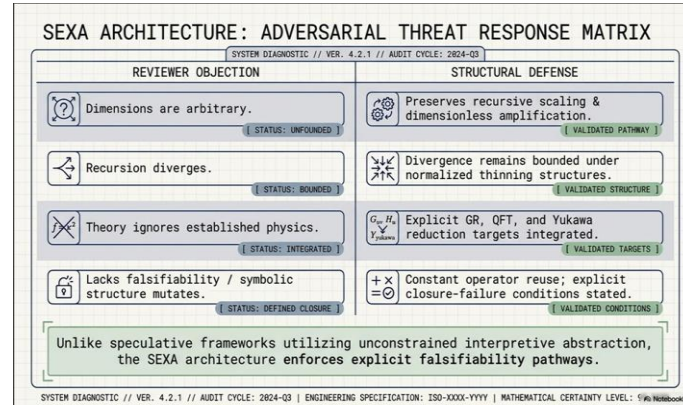
whether the recursive framework remains internally consistent under deterministic mathematical analysis. It does not claim that every physical interpretation associated with the framework has been experimentally established. Maintaining this distinction is essential because engineering validation begins with computational correctness before proceeding toward experimental implementation and physical verification.

The framework further distinguishes itself by requiring the governing recursive structures to remain reusable throughout every stage of analysis. Recursive admissibility, computational correspondence, dimensional reduction, propagation scaling, and reduction-recovery are all governed by the same computational architecture rather than by independent symbolic constructions introduced to resolve isolated analytical problems. This repeated structural reuse reduces opportunities for ad hoc modification while increasing computational traceability throughout the recursive evaluation process.

Another important characteristic of the survivability audit is its emphasis on explicit failure rather than interpretive flexibility. If recursive admissibility fails, dimensional bookkeeping becomes inconsistent, computational correspondence cannot be reproduced, or reduction behavior ceases to remain coherent, the framework itself identifies these outcomes as failure conditions. Consequently, survivability is determined through objective computational evaluation rather than rhetorical interpretation. Such explicit failure pathways are consistent with engineering verification methodologies in which both successful execution and reproducible failure states must remain observable throughout system evaluation.

From the perspective of hardware verification, explicit failure conditions provide an important practical benefit. Computational systems implemented within deterministic hardware environments require clearly defined validation criteria capable of distinguishing acceptable execution from computational failure. Because the recursive framework specifies identifiable breakdown conditions, future implementations may incorporate automated verification procedures capable of detecting structural inconsistency during recursive execution. This capability provides a natural pathway toward hardware-assisted validation of recursive admissibility.

Collectively, the structured survivability methodology establishes a computational evaluation framework rather than a declaration of completed physical proof. The principal result of the present investigation is therefore the demonstration that the recursive computational architecture survives multiple independent categories of mathematical and computational analysis without immediate internal contradiction under the stated assumptions of the framework. This conclusion supports continued investigation through theorem-grade formalization, independent computational reproduction, and future engineering implementation while preserving explicit computational criteria through which the framework may ultimately succeed or fail under continued examination.



6.1. Computational Implications and Future Engineering Implementation

The recursive computational architecture developed throughout this paper provides a foundation for future engineering investigation beyond its present mathematical formulation. By organizing recursive admissibility, dimensional reduction, computational correspondence, propagation scaling, and reduction-recovery within a unified deterministic framework, the architecture defines computational processes that are, in principle, suitable for software implementation and formal verification. Although the present work does not report completed hardware realization, it establishes an organizational structure that may guide future engineering development.

One immediate direction for future work involves software implementation of the recursive admissibility pipeline. The ordered evaluation procedures described throughout the framework naturally lend themselves to deterministic execution within conventional programming environments. Recursive state evaluation, admissibility filtering, orbit generation, harmonic averaging, and dimensional reduction may each be represented as modular computational procedures whose intermediate states remain explicitly observable. Such implementations would permit independent computational reproduction of the recursive correspondence analysis while providing quantitative performance measurements across increasingly complex recursive manifolds.

A second direction concerns formal verification. Modern computational engineering increasingly employs theorem provers, symbolic verification systems, and model-checking frameworks to establish correctness properties prior to deployment. Because the recursive architecture presented herein specifies deterministic computational transitions together with explicit failure conditions, future work may investigate whether admissibility preservation, recursive boundedness, reduction continuity, and computational traceability can be expressed as formally verifiable properties. Such formalization would substantially strengthen the engineering interpretation of the framework while providing independent mathematical validation of its recursive organization.

The recursive structure may also be investigated within specialized computing platforms. Recursive state evolution, orbit correspondence, admissibility evaluation, and dimensional reduction each possess characteristics that may benefit from hardware acceleration through FPGA implementations, custom processing architectures, or parallel computational systems. The deterministic organization of the recursive pipeline is particularly compatible with hardware environments emphasizing reproducible execution, predictable state transitions, and explicit verification procedures. Consequently, the computational framework may provide a useful foundation for exploratory recursive computing architectures extending beyond traditional sequential processing models.

Artificial intelligence represents another potential area of future investigation. Contemporary AI systems generally rely upon statistical optimization and learned parameter spaces rather than explicitly constrained recursive admissibility. The framework presented throughout this paper suggests an alternative direction in which recursive constraint enforcement, structured dimensional reduction, and deterministic correspondence could be incorporated into computational reasoning systems. Whether such approaches provide measurable advantages remains an open engineering question requiring independent computational experimentation.

The present investigation therefore concludes with an engineering perspective rather than a claim of completed technological realization. The recursive computational architecture developed herein establishes a mathematically organized and computationally traceable framework whose strongest current contribution lies in its deterministic organization, explicit validation methodology, and structured computational correspondence. Future progress depends upon theorem-grade mathematical formalization, independent computational reproduction, prototype software implementation, and eventual experimental evaluation within hardware-assisted computational environments.

The transition from recursive mathematical formalism to executable computational systems represents the next logical stage of investigation. Accordingly, the framework presented throughout this paper should be viewed not as a completed engineering solution, but as a structured computational architecture intended to support continued interdisciplinary research spanning recursive mathematics, computational science, artificial intelligence, embedded systems, and hardware verification.

7. CONCLUSION

This paper has presented a computational reinterpretation of the SEXA Unified Field Framework from the perspective of recursive computing, computational correspondence, and hardware-verifiable systems engineering. Rather than introducing additional governing equations, the investigation has evaluated the existing recursive architecture through a structured computational methodology emphasizing deterministic execution, recursive admissibility, dimensional consistency, computational traceability, reduction-recovery behavior, and explicit falsifiability. Collectively, these analyses examine whether the framework possesses the organizational characteristics expected of a recursive computational architecture suitable for continued engineering investigation.

The central contribution of the present work is the demonstration that the recursive framework may be organized as a deterministic computational system rather than solely as a symbolic mathematical construction. Through the integration of recursive admissibility, the SEXA Recursive Energy Functional (SREF), computational correspondence derived through Sigmatics analysis, dimensional reduction, recursive thinning, propagation scaling, and structured validation, the framework establishes a unified computational pipeline whose intermediate states remain algorithmically identifiable throughout recursive evaluation. This computational organization provides a reproducible analytical structure that may serve as the basis for future software implementation, formal verification, and hardware-assisted computation.

The computational correspondence analysis constitutes the strongest engineering component of the present investigation. By relating recursive manifold inheritance to deterministic orbit structures, harmonic averaging, admissibility filtering, and recursive propagation, the correspondence layer transforms higher-dimensional recursive organization into computable objects suitable for algorithmic evaluation. This transition from symbolic formulation toward deterministic computation distinguishes the framework from many speculative recursive systems

whose mathematical structures cannot presently be represented through reproducible computational procedures.

The reduction-recovery analysis further demonstrates that the recursive architecture preserves coherent computational organization while evaluating limiting behavior corresponding to General Relativity, Quantum Field Theory, and Yukawa-type interaction regimes. Within the scope of the present investigation, these analyses are interpreted as structural reduction assessments rather than claims of completed physical replacement. The resulting framework therefore emphasizes computational continuity across multiple analytical domains while explicitly distinguishing mathematical survivability from experimentally established physical confirmation.

An equally important outcome of the investigation is the incorporation of explicit computational failure conditions. Recursive admissibility, computational correspondence, dimensional bookkeeping, reduction continuity, and recursive stability each possess identifiable breakdown criteria that permit objective evaluation under deterministic computational procedures. This emphasis on structured falsifiability strengthens the engineering interpretation of the framework by replacing unrestricted symbolic flexibility with observable computational success and failure conditions.

The present work therefore concludes that the strongest currently supported component of the SEXA Unified Field Framework is its recursive computational organization. Under the assumptions adopted throughout the analysis, the framework survives structured evaluation of recursive consistency, computational correspondence, dimensional reduction, reduction-recovery behavior, and explicit falsifiability without exhibiting immediate internal mathematical contradiction. These results do not constitute completed experimental confirmation or theorem-grade formal proof; rather, they identify a coherent recursive computational architecture that warrants continued mathematical, computational, and engineering investigation.

Future work should therefore prioritize rigorous operator formalization, independent computational reproduction of the correspondence analysis, software implementation of recursive admissibility algorithms, hardware-assisted verification, and experimentally testable engineering applications derived from the recursive computational framework. Progress in these areas will determine whether the recursive architecture can advance from a mathematically organized computational formalism toward a practically implementable engineering system.

Accordingly, the contribution of this conference paper is not the assertion of completed physical unification, but the presentation of a structured recursive computational architecture whose deterministic organization, computational correspondence, and engineering-oriented validation establish a reproducible foundation for continued interdisciplinary research in recursive computing, artificial intelligence, computational mathematics, embedded systems, and hardware verification.

REFERENCES

- [1] SEXA Unified Field Theory: The Structured Survivability Audit. Recursive Admissibility, Computational Correspondence, Dimensional Reduction, and Reduction-Recovery Across 5D–2880D Manifold Regimes. Validation manuscript used as the principal computational reference for the present conference edition.
- [2] The SEXA Mathematical Framework. Unified Recursive Manifold Dynamics, Dimensional Collapse Operators, and Triality-Structured Exciter Geometry. Zenodo.
- [3] McClain, J. The SEXA Recursive Energy Functional (SREF): Spectral Gain-60 Recursion on a Five-Dimensional Manifold and Bounded Dynamical Stability. Zenodo.
- [4] McClain, J. A Recursive Closure Criterion for a Theory of Everything: A 60-Glyph Quaternionic

- Inter-Domain Syllogistic Admissibility Standard (Prime Atom of Logic). Zenodo.
- [5] Flom, A. Sigmatics: A 96-Class Geometric Algebra Computational Framework for High-Dimensional Recursive Structures. Zenodo.
 - [6] A Monster-Symmetric Admissibility Formulation of the SEXA Unified Field Theory. *Journal of Advances in Mathematics*, Vol. 25 (2026). DOI: 10.24297/jam.v25i.9887.
 - [7] Formal Compatibility and Falsifiability Assessment of the SEXA Unified Field Model. Zenodo.
 - [8] Albert Einstein. The Foundation of the General Theory of Relativity. *Annalen der Physik*, 49, 769–822 (1916).
 - [9] Hideki Yukawa. On the Interaction of Elementary Particles. *Proceedings of the Physico-Mathematical Society of Japan*, 17, 48–57 (1935).
 - [10] *An Introduction to Quantum Field Theory*. Addison–Wesley, 1995.
 - [11] *Quantum Field Theory and the Standard Model*. Cambridge University Press, 2020.
 - [12] Edward Witten. Symmetry and Emergence. *Nature Physics*, 14, 116–119 (2018).
 - [13] John H. Conway and Simon P. Norton. Monstrous Moonshine. *Bulletin of the London Mathematical Society*, 11(3), 308–339 (1979).
 - [14] Robert L. Griess Jr.. The Friendly Giant. *Inventiones Mathematicae*, 69(1), 1–102 (1982).
 - [15] *Lost in Math*. Basic Books, 2018.
 - [16] Helgoland. Riverhead Books, 2021.