Introduction

Daniel Ari Friedman
ORCID: 0000-0001-6232-9096
Email: daniel@activeinference.institute

August 14, 2025

Introduction

Quadray coordinates provide a tetrahedral basis for modeling space and computation, standing in contrast to Cartesian frameworks. Originating in Buckminster Fuller's Synergetics, quadray coordinates enable the replacement of right-angle assumptions, with 60-degree coordination and a unit tetrahedron of volume 1. This reframing yields striking integer relationships among common polyhedra and provides a natural account of space via close-packed spheres and the isotropic vector matrix (IVM).

In this synthetic review, we distinguish three internal meanings of "4D," following a dot-notation that avoids cross-domain confusion:

- Coxeter.4D — four-dimensional Euclidean space (E), as in classical polytope theory. Coxeter emphasizes that Euclidean 4D is not spacetime; see the Dover edition of Regular Polytopes (p. 119) for a clear statement to this effect; background on lattice packings in four dimensions aligns with the treatment in Conway & Sloane's Sphere Packings, Lattices and Groups (Springer).
- Einstein.4D — Minkowski spacetime (3D + time) with an indefinite metric; appropriate for relativistic physics but distinct from Euclidean E .
- Fuller.4D — synergetics' tetrahedral accounting of space using Quadrays (four non-negative coordinates with at least one zero after normalization) and the IVM = CCP = FCC correspondence. This treats the regular tetrahedron as a natural unit container and emphasizes angle/shape relations independent of time/energy.

Kirby Urner's expositions and implementations have been influential in making Quadray coordinates practical and accessible across multiple programming languages and educational contexts. See the comprehensive 4dsolutions ecosystem:

- **Foundational materials**: Urner – Quadray intro, Quadrays and XYZ, Quadrays and the Philosophy of Mathematics

- **Python implementations**: Core modules `qrays.py` (Quadray vectors with SymPy support) and `tetravolume.py` (IVM volumes, BEAST modules)
- **Educational notebooks**: School_of_Tomorrow including Qvolume.ipynb (Tom Ace 5×5) and VolumeTalk.ipynb (bridging vs native)
- **Cross-language validation**: Rust implementation, Clojure functional approach, VPython visualizations
- **Historical context**: Python edu-sig post (May 2000)

This paper unifies three threads:

- **Foundations**: Quadray coordinates and their relation to 4D modeling more generally, with explicit namespace usage (Coxeter.4D, Einstein.4D, Fuller.4D) to maintain clarity.
- **Optimization framework**: leverages integer volume quantization on tetrahedral lattices to achieve robust, discrete convergence.
- **Information geometry**: tools (e.g., Fisher Information, free-energy minimization) for interpreting optimization as geodesic motion on statistical manifolds.

Contributions:

- **Namespaces mapping**: Coxeter.4D (Euclidean E), Einstein.4D (Minkowski spacetime), and Fuller.4D (Quadrays/IVM) → analytical tools and examples.
- **Quadray-adapted Nelder–Mead**: integer-lattice normalization and volume-level tracking.
- **Equations and methods**: comprehensive supplement with guidance for high-precision computation using `libquadmath`.
- **Discrete optimizer**: integer-valued variational descent over the IVM (`discrete_ivm_descent`) with animation tooling, connecting lattice geometry to information-theoretic objectives.

Related work and background

- **Tetrahedron volume formulas**: length-based Cayley–Menger determinant and determinant-based expressions on vertex coordinates (see Tetrahedron – volume).
- **Exact determinants**: Bareiss algorithm, used in our integer tetravolume implementations.
- **Information geometry**: Fisher information and natural gradient.
- **Optimization baseline**: the Nelder–Mead method, adapted here to the Quadray lattice.
- **4dsolutions ecosystem (comprehensive implementation suite)**: The 4dsolutions organization provides extensive computational resources for Quadrays and synergetic geometry:
  - **Primary Python modules**: `qrays.py` (Quadray vector operations, normalization, XYZ bridging) and `tetravolume.py` (IVM volumes,

BEAST modules, multiple algorithms)
  - **Educational framework**: School_of_Tomorrow with interactive tutorials and algorithm comparisons in Qvolume.ipynb and VolumeTalk.ipynb
  - **Cross-platform validation**: Independent implementations in Rust, Clojure, POV-Ray pipelines (quadcraft.py), and VPython animations
- **Related work**: QuadCraft is a tetrahedral voxel engine game using Quadray coordinates.

Ecosystem context: The 4dsolutions organization spans 29+ repositories with implementations across Python, Rust, Clojure, POV-Ray, and VPython, providing extensive cross-language validation and educational resources. Core algorithmic modules include vector operations, volume calculations, visualization pipelines, and pedagogical frameworks that complement and validate the methods developed in this manuscript. See the comprehensive catalog in `07_resources.md`.

Companion code and tests

The manuscript is accompanied by a fully-tested Python codebase under `src/` with unit tests under `tests/`. Key artifacts used throughout the paper:

- **Quadray APIs**: `src/quadray.py` (`Quadray`, `integer_tetra_volume`, `ace_tetravolume_5x5`).
- **Determinant utilities**: `src/linalg_utils.py` (`bareiss_determinant_int`).
- **Length-based volume**: `src/cayley_menger.py` (`tetra_volume_cayley_menger`, `ivm_tetra_volume_cayley_menger`).
- **XYZ conversion**: `src/conversions.py` (`urner_embedding`, `quadray_to_xyz`).
- **Examples**: `src/examples.py` (`example_ivm_neighbors`, `example_volume`, `example_optimize`).

Fig. 1 (graphical abstract): Panel A shows Quadray axes (A,B,C,D) under a symmetric embedding with wireframe context. Panel B shows close-packed spheres at the tetrahedron vertices (IVM/CCP/FCC, "twelve around one").

Tests illustrate expected behavior and edge cases (see `tests/`), and coverage is enforced at 100% for `src/`.

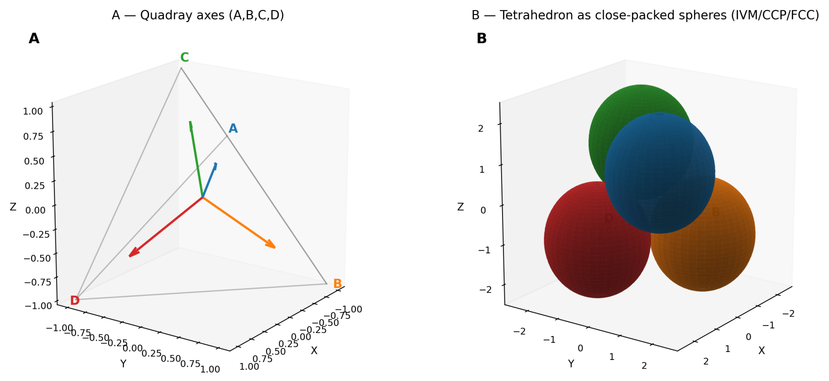
A — Quadray axes (A,B,C,D)
A
C
A
B
D
X
Y
Z
B — Tetrahedron as close-packed spheres (IVM/CCP/FCC)
B
X
Y
Z

Figure 1: Graphical abstract (2×1 panels). A: Quadray axes (A,B,C,D) with wireframe. B: close-packed spheres at tetrahedron vertices (IVM/CCP/FCC).