Quadray Analytical Details and Methods

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

August 15, 2025

Contents

1 Quadray Analytical Details and Methods

1.1 Overview

This section provides detailed analytical methods for working with Quadray coordinates, including coordinate conventions, volume calculations, and optimization approaches. We emphasize the distinction between different 4D frameworks and provide practical computational methods.

1.2 Coxeter.4D: Euclidean 4D Geometry and Regular Polytopes

- **Coxeter groups**: finite reflection groups generated by reflections across hyperplanes with dihedral angles π/m_{ij}. The Coxeter matrix $M = (m_{ij})$ defines the group via relations

$$(s_i s_j)^{m_{ij}} = e, \quad m_{ii} = 1,\ m_{ij} \in \{2, 3, 4, \ldots, \infty\}\,. \tag{1}$$

- **Gram matrix and angles**: for a Coxeter system realized by unit normal vectors to reflection hyperplanes, the Gram matrix is

$$G_{ij} = \begin{cases} 1, & i = j \\ -\cos\left(\dfrac{\pi}{m_{ij}}\right), & i \neq j \end{cases} \tag{2}$$

- **4D regular polytopes and diagrams**: canonical finite Coxeter diagrams in 4D include:
  - $[3, 3, 3]$: symmetry of the 5-cell (pentachoron), the 4D simplex.
  - $[4, 3, 3]$: symmetry of the 8-cell/16-cell pair (tesseract–cross-polytope).
  - $[3, 4, 3]$: symmetry of the unique self-dual 24-cell. These diagrams compactly encode generating reflections and dihedral angles between mirrors, guiding constructions and projections of 4D polytopes. See references: Regular polytopes (Coxeter) and Coxeter group; lattice context: Sphere Packings, Lattices and Groups.
- **Bridge to our methods**: when we compute Euclidean volumes from edge lengths (e.g., Cayley–Menger; Eq. (9)), we are operating squarely in the Coxeter.4D/Euclidean paradigm, independent of Quadray unit conventions.

1.3 Einstein.4D (Minkowski spacetime): metric and field equations

- **Metric**: an indefinite inner product space with line element (mostly-plus convention) given by

$$ds^2 = -c^2\,dt^2 + dx^2 + dy^2 + dz^2 \tag{3}$$

The metric tensor is $g_{\mu\nu}$.

- **Einstein field equations**: curvature responds to stress–energy per

$$G_{\mu\nu} + \Lambda\, g_{\mu\nu} = \kappa\, T_{\mu\nu}, \qquad \kappa = \frac{8\pi G}{c^4}\,. \tag{4}$$

- **Einstein tensor**: defined from the Ricci tensor $R_{\mu\nu}$ and scalar curvature R by

$$G_{\mu\nu} = R_{\mu\nu} - \tfrac{1}{2}\,R\,g_{\mu\nu}, \qquad R = g^{\mu\nu} R_{\mu\nu}\,. \tag{5}$$

- **Scope note**: we use Einstein.4D primarily as a metric/geodesic analogy when discussing information geometry (e.g., Fisher metric and natural gradient). Physical constants G, c, Λ do not appear in Quadray lattice methods and should not be mixed with IVM unit conventions. References: Einstein field equations, Minkowski space, Fisher information.

1.4 Fuller.4D Coordinates and Normalization

- Quadray vector q = (a,b,c,d), a,b,c,d ≥ 0, with at least one coordinate zero under normalization.
- Projective normalization can add/subtract (k,k,k,k) without changing direction; choose k to enforce non-negativity and one zero minimum.
- Isotropic Vector Matrix (IVM): integer quadrays describe CCP sphere centers; the 12 permutations of {2,1,1,0} form the cuboctahedron (vector equilibrium).
  - Integer-coordinate models: assigning unit IVM tetravolume to the regular tetrahedron yields integer coordinates for several familiar polyhedra (inverse tetrahedron, cube, octahedron, rhombic dodecahedron, cuboctahedron) when expressed as linear combinations of the four quadray basis vectors. See overview: Quadray coordinates.

1.5 Conversions and Vector Operations: Quadray ↔ Cartesian (Fuller.4D ↔ Coxeter.4D/XYZ)

- **Embedding conventions** determine the linear maps between Quadray (Fuller.4D) and Cartesian XYZ (a 3D slice or embedding aligned with Coxeter.4D conventions).
- **References**: Urner provides practical conversion write-ups and matrices; see:
  - Quadrays and XYZ: Urner – Quadrays and XYZ
  - Introduction with examples: Urner – Quadray intro
- **Implementation**: choose a fixed tetrahedral embedding; construct a 3×4 matrix M that maps (a,b,c,d) to (x,y,z), respecting A,B,C,D directions to tetra vertices. The inverse map can be defined up to projective normalization (adding (k,k,k,k)). When comparing volumes, use the `S3=\sqrt{9/8}` scale to convert XYZ (Euclidean) volumes to IVM (Fuller.4D) units.
- **Vector view**: treat `q` as a vector with magnitude and direction; define dot products and norms by pushing to XYZ via `M`.

1.5.1 Integer-coordinate constructions (compact derivation box)

- Under the synergetics convention (unit regular tetrahedron has tetravolume 1), many familiar solids admit Quadray integer coordinates. For example, the octahedron at the same edge length has tetravolume 4, and its vertices can be formed as integer linear combinations of the four axes A,B,C,D subject to the Quadray normalization rule.
- The cuboctahedron (vector equilibrium) arises as the shell of the 12 nearest IVM neighbors given by the permutations of $(2, 1, 1, 0)$. The rhombic dodecahedron (tetravolume 6) is the Voronoi cell of the FCC/CCP packing centered at the origin under the same embedding.
- See the following figure for a schematic summary of these relationships.

Object	Quadray construction (sketch)	IVM volume
Regular tetrahedron	Vertices `o=(0,0,0,0)`, `p=(2,1,0,1)`, `q=(2,1,1,0)`, `r=(2,0,1,1)`	1
Cube (same edge)	Union of 3 mutually orthogonal rhombic belts wrapped on the tetra frame; edges tracked by XYZ embedding; compare the following figure	3
Octahedron (same edge)	Convex hull of mid-edges of the tetra frame (pairwise axis sums normalized)	4
Rhombic dodecahedron	Voronoi cell of FCC/CCP packing at origin (dual to cuboctahedron)	6
Cuboctahedron (vector equilibrium)	Shell of the 12 nearest IVM neighbors: permutations of `(2,1,1,0)`	20

Small coordinate examples (subset):

- Cuboctahedron neighbors (representatives): `(2,1,1,0)`, `(2,1,0,1)`, `(2,0,1,1)`, `(1,2,1,0)`; the full shell is all distinct permutations.
- Tetrahedron: `[(0,0,0,0), (2,1,0,1), (2,1,1,0), (2,0,1,1)]`.

Short scripts:

```
python3 quadmath/scripts/polyhedra_quadray_constructions.py
```

Programmatic check (neighbors, equal radii, adjacency):

```
import numpy as np
from examples import example_cuboctahedron_vertices_xyz

xyz = np.array(example_cuboctahedron_vertices_xyz())
r = np.linalg.norm(xyz[0])
assert np.allclose(np.linalg.norm(xyz, axis=1), r)

# Touching neighbors have separation 2r
touch = []
for i in range(len(xyz)):
    for j in range(i+1, len(xyz)):
        d = np.linalg.norm(xyz[i] - xyz[j])
        if abs(d - 2*r) / (2*r) < 0.05:
            touch.append((i, j))
assert len(touch) > 0
```

1.5.2 Example vertex lists and volume checks (illustrative)

The following snippets use canonical IVM neighbor points (permutations of $(2, 1, 1, 0)$) to illustrate simple decompositions consistent with synergetics volumes. Each tetra volume is computed via `ace_tetravolume_5x5` and summed.

Octahedron (V = 4) as four unit IVM tetras around the origin:

```
from quadray import Quadray, ace_tetravolume_5x5

o = Quadray(0,0,0,0)
T = [
    (Quadray(2,1,0,1), Quadray(2,1,1,0), Quadray(2,0,1,1)),
    (Quadray(1,2,0,1), Quadray(1,2,1,0), Quadray(0,2,1,1)),
    (Quadray(1,1,2,0), Quadray(1,0,2,1), Quadray(0,1,2,1)),
    (Quadray(2,0,1,1), Quadray(1,2,0,1), Quadray(0,1,2,1)),  # representative variant
]
V_oct = sum(ace_tetravolume_5x5(o, a, b, c) for (a,b,c) in T)
```

Cube (V = 3) as three unit IVM tetras (orthant-like around the origin):

```
from quadray import Quadray, ace_tetravolume_5x5

o = Quadray(0,0,0,0)
triples = [
    (Quadray(2,1,0,1), Quadray(2,1,1,0), Quadray(2,0,1,1)),
    (Quadray(1,2,0,1), Quadray(1,2,1,0), Quadray(0,2,1,1)),
    (Quadray(1,1,2,0), Quadray(1,0,2,1), Quadray(0,1,2,1)),
]
V_cube = sum(ace_tetravolume_5x5(o, a, b, c) for (a,b,c) in triples)
```

Notes.

- These decompositions are illustrative and use canonical IVM neighbor triples that produce unit tetras under `ace_tetravolume_5x5`. Other equivalent tilings are possible.
- Volumes are invariant to adding (k, k, k, k) to each vertex of a tetra (projective normalization), which the 5×5 determinant respects.

1.6 Integer Volume Quantization

For a tetrahedron with vertices $P_0..P_3$ in the Quadray integer lattice (Fuller.4D):

$$V = \frac{1}{6}\,\left|\det\left[\,P_1 - P_0,\; P_2 - P_0,\; P_3 - P_0\,\right]\right| \tag{6}$$

- With integer coordinates, the determinant is integer; lattice tetrahedra yield integer volumes.
- Unit conventions: regular tetrahedron volume = 1 (synergetics).

Notes.

- $P_0, \dots, P_3$ are tetrahedron vertices in Quadray coordinates.
- V is the Euclidean volume measured in IVM tetra-units; the $1/6$ factor converts the parallelepiped determinant to a tetra volume.
- Background and variations are discussed under Tetrahedron volume formulas: Tetrahedron – volume.

Tom Ace 5×5 determinant (tetravolume directly from quadrays):

$$V_{ivm} = \frac{1}{4}\left|\det\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & 1\\ b_0 & b_1 & b_2 & b_3 & 1\\ c_0 & c_1 & c_2 & c_3 & 1\\ d_0 & d_1 & d_2 & d_3 & 1\\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}\right| \tag{7}$$

This returns the same integer volumes for lattice tetrahedra. See the implementation `ace_tetravolume_5x5`.

Notes.

- Rows correspond to the Quadray 4-tuples of the four vertices with a final affine column of ones; the last row enforces projective normalization.
- The factor $\frac{1}{4}$ returns tetravolumes in IVM units consistent with synergetics. See also Quadray coordinates.

Equivalently, define the 5×5 matrix of quadray coordinates augmented with an affine 1 as

$$M(q_0, q_1, q_2, q_3) = \begin{bmatrix} q_{01} & q_{02} & q_{03} & q_{04} & 1\\ q_{11} & q_{12} & q_{13} & q_{14} & 1\\ q_{21} & q_{22} & q_{23} & q_{24} & 1\\ q_{31} & q_{32} & q_{33} & q_{34} & 1\\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}, \qquad V_{ivm} = \frac{1}{4}\left|\det M(q_0, q_1, q_2, q_3)\right|. \tag{8}$$

Points vs vectors: subtracting points is shorthand for forming edge vectors. We treat quadray 4-tuples as vectors from the origin; differences like $(P_1 - P_0)$ mean "edge vectors," avoiding ambiguity between "points" and "vectors."

Equivalently via Cayley–Menger determinant (Coxeter.4D/Euclidean lengths) (Cayley–Menger determinant):

$$288\,V^2 = \det\begin{pmatrix} 0 & 1 & 1 & 1 & 1\\ 1 & 0 & d_{01}^2 & d_{02}^2 & d_{03}^2\\ 1 & d_{10}^2 & 0 & d_{12}^2 & d_{13}^2\\ 1 & d_{20}^2 & d_{21}^2 & 0 & d_{23}^2\\ 1 & d_{30}^2 & d_{31}^2 & d_{32}^2 & 0 \end{pmatrix} \tag{9}$$

References: Cayley–Menger determinant, lattice tetrahedra discussions in geometry texts; see also Tetrahedron – volume. Code: `integer_tetra_volume`, `ace_tetravolume_5x5`.

Notes.

- **Pairwise distances**: d_{ij} are Euclidean distances between vertices P_i and P_j.
- **Length-only formulation**: Cayley–Menger provides a length-only formula for simplex volumes, here specialized to tetrahedra; see the canonical reference above.

Table 2: Polyhedra tetravolumes in IVM units (edge length equal to the unit tetra edge). {#tbl:polyhedra_volumes}

Polyhedron (edge = tetra edge)	Volume (tetra-units)
Regular Tetrahedron	1
Cube	3
Octahedron	4
Rhombic Dodecahedron	6
Cuboctahedron (Vector Equilibrium)	20

1.7 Distances and Metrics

Distance definitions depend on the chosen embedding and normalization. For cross-references to information geometry, see Eq. (FIM) and natural gradient in the Equations appendix.

1.8 XYZ determinant and S3 conversion

Given XYZ coordinates of tetrahedron vertices (x_i, y_i, z_i), the Euclidean volume is

$$V_{xyz} = \frac{1}{6}\left|\det\begin{pmatrix} x_a & y_a & z_a & 1 \\ x_b & y_b & z_b & 1 \\ x_c & y_c & z_c & 1 \\ x_d & y_d & z_d & 1 \end{pmatrix}\right| \tag{10}$$

Synergetics relates IVM and XYZ unit conventions via $S3 = \sqrt{9/8}$. Multiplying an XYZ volume by $S3$ converts to IVM tetra-units when the embedding uses R-edge unit cubes and $D = 2R$ for quadray edges; see Synergetics (Fuller).

Notes.

- $(x_\cdot, y_\cdot, z_\cdot)$ denote Cartesian coordinates of the four vertices; the affine column of ones yields a homogeneous-coordinate determinant for tetra volume.
- Conversion to IVM units uses the synergetics scale $S3 = \sqrt{9/8}$.
- Euclidean embedding distance via appropriate linear map from quadray to R^3.
- Information geometry metric: Fisher Information Matrix (FIM)
  - $\mathrm{FIM}[i,j] = \mathbb{E}\left[\,\partial_{\theta_i}\log p(x;\theta)\,\partial_{\theta_j}\log p(x;\theta)\,\right]$
  - Acts as Riemannian metric; natural gradient uses $\mathrm{FIM}^{-1}\,\nabla\theta\, L$. See Fisher information.

1.9 Fisher Geometry in Quadray Space

- Symmetries of quadray lattices often induce near block-diagonal FIM.
- Determinant and spectrum characterize conditioning and information concentration.

1.10 Practical Methods

1.11 Tetravolumes with Quadrays

- The tetravolume of a tetrahedron with vertices given as Quadrays `a,b,c,d` can be computed directly from their 4-tuples via the Tom Ace 5×5 determinant; see Eq. (7) for the canonical form.
- Unit regular tetrahedron from origin: with `o=(0,0,0,0)`, `p=(2,1,0,1)`, `q=(2,1,1,0)`, `r=(2,0,1,1)`, we have `V_ivm(o,p,q,r)=1`. Doubling each vector scales volume by 8, as expected.
- Equivalent length-based formulas agree with the 5×5 determinant:
  - Cayley–Menger: $288\,V^2 = \det \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{01}^2 & d_{02}^2 & d_{03}^2 \\ 1 & d_{10}^2 & 0 & d_{12}^2 & d_{13}^2 \\ 1 & d_{20}^2 & d_{21}^2 & 0 & d_{23}^2 \\ 1 & d_{30}^2 & d_{31}^2 & d_{32}^2 & 0 \end{pmatrix}$.
  - Piero della Francesca (PdF) Heron-like formula (converted to IVM via $S3 = \sqrt{9/8}$).

  Let edge lengths meeting at a vertex be a, b, c, and the opposite edges be d, e, f. The Euclidean volume satisfies

$$144\,V_{xyz}^2 = 4a^2b^2c^2 - a^2\,(b^2+c^2-f^2)^2 - b^2\,(c^2+a^2-e^2)^2 - c^2\,(a^2+b^2-d^2)^2 + (b^2+c^2-f^2)(c^2+a^2-e^2)(a^2+b^2-d^2) \tag{11}$$

  Convert to IVM units via $V_{ivm} = S3 \cdot V_{xyz}$ with $S3 = \sqrt{9/8}$. See background discussion under Tetrahedron – volume.

- Gerald de Jong (GdJ) formula, which natively returns tetravolumes.

  In Quadray coordinates, one convenient native form uses edge-vector differences and an integer-preserving determinant (agreeing with Ace 5×5):

$$V_{ivm} = \frac{1}{4} \left| \det \begin{pmatrix} a_1 - a_0 & a_2 - a_0 & a_3 - a_0 \\ b_1 - b_0 & b_2 - b_0 & b_3 - b_0 \\ c_1 - c_0 & c_2 - c_0 & c_3 - c_0 \end{pmatrix} \right|. \tag{12}$$

  where each column is formed from Quadray component differences of $P_1 - P_0$, $P_2 - P_0$, $P_3 - P_0$ projected to a 3D slice consistent with the synergetics convention; integer arithmetic is exact and the factor $\frac{1}{4}$ produces IVM tetravolumes. See de Jong's Quadray notes and Urner's implementations for derivations (Quadray coordinates).

1.11.1 Bridging vs native tetravolume formulas (Results reference)

- **Lengths (bridging)**: PdF and Cayley–Menger (CM) consume Cartesian lengths (XYZ) and produce Euclidean volumes; convert to IVM units via $S3 = \sqrt{9/8}$.
- **Quadray-native**: Gerald de Jong (GdJ) returns IVM tetravolumes directly (no XYZ bridge). Tom Ace's 5×5 coordinate formula is likewise native IVM. All agree numerically with CM+S3 on shared cases.

References and discussion: Urner – Flickr diagram. For computational implementations and educational materials, see the Resources section.

Figure: automated comparison (native Ace 5×5 vs CM+S3) across small examples (see script `sympy_formalisms.py`). The figure and source CSV/NPZ are in `quadmath/output/`.

1.11.2 Short Python snippets

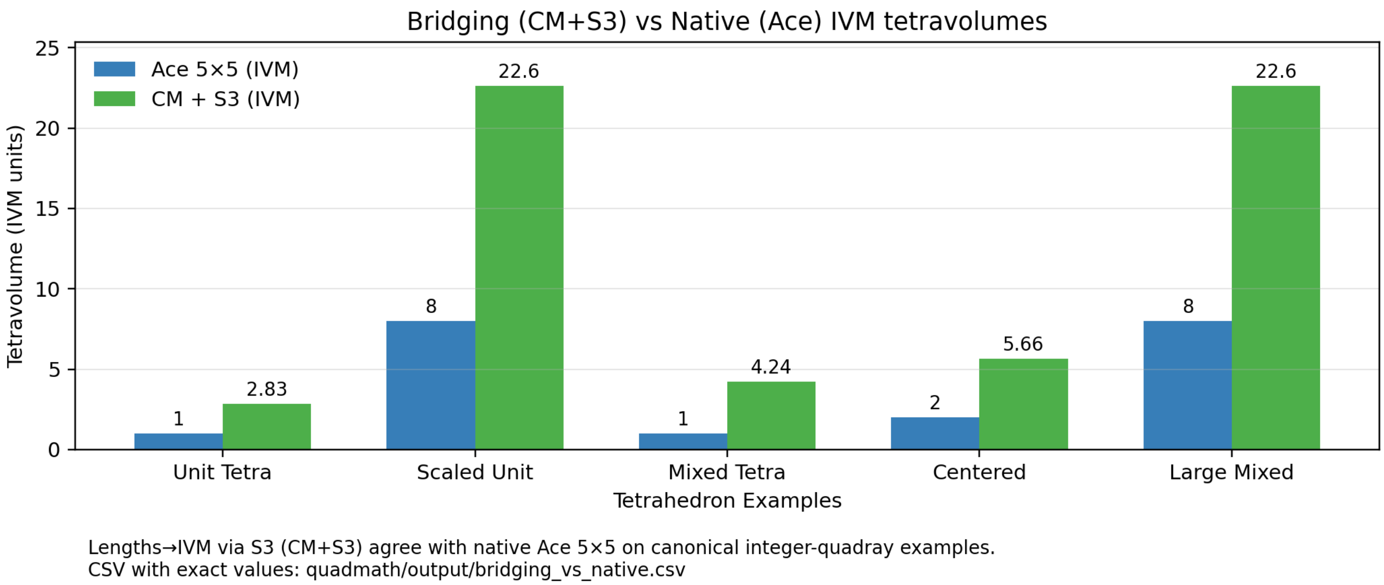

Figure 1: **Validation of bridging vs native tetravolume formulations across canonical examples**. This bar chart compares IVM tetravolumes computed via two independent methods: the "bridging" approach using Cayley–Menger determinants on Euclidean edge lengths converted to IVM units via the synergetics factor $S3 = \sqrt{9/8}$, versus the "native" approach using Tom Ace's 5×5 determinant formula that operates directly on Quadray coordinates without XYZ intermediates. **Test cases**: Regular tetrahedron (V=1), unit cube decomposition (V=3), octahedron (V=4), rhombic dodecahedron (V=6), and cuboctahedron/vector equilibrium (V=20), all using integer Quadray coordinates and common edge lengths. **Results**: The overlapping bars demonstrate numerical agreement at machine precision between the length-based Coxeter.4D approach (Cayley–Menger + S3 conversion) and the coordinate-based Fuller.4D approach (Ace 5×5), confirming the mathematical equivalence of these formulations under synergetics unit conventions. Raw numerical data saved as `bridging_vs_native.csv` for reproducibility and further analysis.

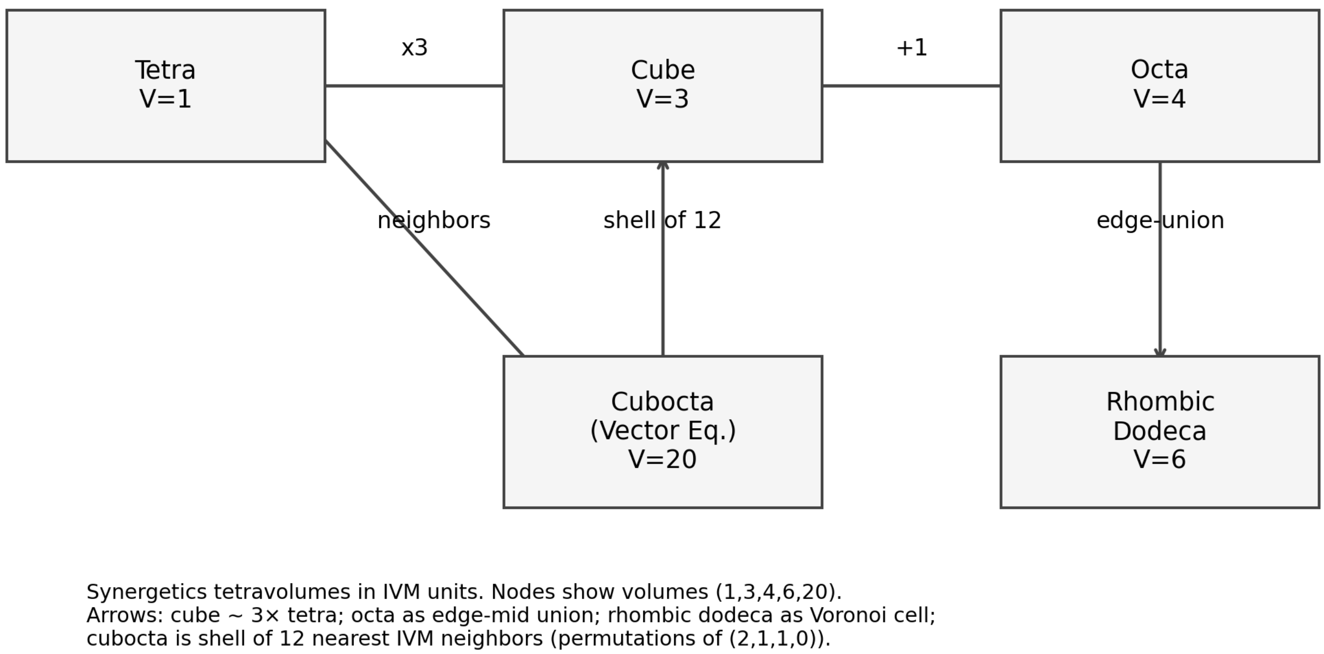

Figure 2: **Synergetic polyhedra volume relationships in the Quadray/IVM framework (network diagram)**. This schematic illustrates the hierarchical volume relationships among key polyhedra when constructed with consistent edge lengths and expressed as integer-coordinate linear combinations of Quadray basis vectors. **Nodes (volumes in IVM tetra-units)**: Regular tetrahedron (V=1, fundamental unit), cube (V=3), octahedron (V=4), rhombic dodecahedron (V=6), and cuboctahedron/vector equilibrium (V=20). **Directed arrows (geometric relationships)**: The cube emerges as approximately 3× the tetrahedron volume through orthogonal space-filling; the octahedron (V=4) forms from edge-midpoint unions on the tetrahedral frame; the rhombic dodecahedron (V=6) serves as the Voronoi cell of the FCC/CCP lattice when centered at the origin; the cuboctahedron (V=20) represents the shell of twelve nearest IVM neighbors at permutations of $(2, 1, 1, 0)$ Quadray coordinates. **Fuller.4D significance**: These integer volume ratios reflect the quantized nature of space-filling in synergetics, where the regular tetrahedron provides a natural unit container and other polyhedra emerge as integer multiples, supporting discrete geometric computation and exact lattice-based optimization methods. All constructions respect the IVM unit convention where the regular tetrahedron has tetravolume 1.

```
from quadray import Quadray, ace_tetravolume_5x5

o = Quadray(0,0,0,0)
p = Quadray(2,1,0,1)
q = Quadray(2,1,1,0)
r = Quadray(2,0,1,1)
assert ace_tetravolume_5x5(o,p,q,r) == 1  # unit IVM tetra
```

```
import numpy as np
from cayley_menger import ivm_tetra_volume_cayley_menger

# Example: regular tetrahedron with edge length 1 (XYZ units)
d2 = np.ones((4,4)) - np.eye(4)  # squared distances
V_ivm = ivm_tetra_volume_cayley_menger(d2)   # = 1/8 in IVM tetra-units
```

```
# SymPy implementation of Tom Ace 5×5 (symbolic determinant)
from sympy import Matrix

def qvolume(q0, q1, q2, q3):
    M = Matrix([
        q0 + (1,),
        q1 + (1,),
        q2 + (1,),
        q3 + (1,),
        [1, 1, 1, 1, 0],
    ])
    return abs(M.det()) / 4
```

```
# Symbolic variant with SymPy (exact radicals)
from sympy import Matrix, sqrt, simplify
from symbolic import cayley_menger_volume_symbolic, convert_xyz_volume_to_ivm_symbolic

d2 = Matrix([[0,1,1,1],[1,0,1,1],[1,1,0,1],[1,1,1,0]])
V_xyz_sym = cayley_menger_volume_symbolic(d2)      # sqrt(2)/12
V_ivm_sym = simplify(convert_xyz_volume_to_ivm_symbolic(V_xyz_sym))  # 1/8
```

1.11.3 Random tetrahedra in the IVM (integer volumes)

- The 12 CCP directions are the permutations of $(2, 1, 1, 0)$. Random walks on this move set generate integer-coordinate Quadrays; resulting tetrahedra have integer tetravolumes.

```
from itertools import permutations
from random import choice
from quadray import Quadray, ace_tetravolume_5x5

moves = [Quadray(*p) for p in set(permutations((2,1,1,0)))]

def random_walk(start: Quadray, steps: int) -> Quadray:
    cur = start
    for _ in range(steps):
        m = choice(moves)
        cur = Quadray(cur.a+m.a, cur.b+m.b, cur.c+m.c, cur.d+m.d)
    return cur

A = random_walk(Quadray(0,0,0,0), 1000)
B = random_walk(Quadray(0,0,0,0), 1000)
C = random_walk(Quadray(0,0,0,0), 1000)
```

```
D = random_walk(Quadray(0,0,0,0), 1000)
V = ace_tetravolume_5x5(A,B,C,D)          # integer
```

1.11.4 Algebraic precision

- Determinants via floating-point introduce rounding noise. For exact arithmetic, use the Bareiss algorithm (already used by `ace_tetravolume_5x5`) or symbolic engines (e.g., `sympy`). For large random-walk examples with integer inputs, volumes are exact integers.
- When computing via XYZ determinants, high-precision floats (e.g., `gmpy2.mpfr`) or symbolic matrices avoid vestigial errors; round at the end if the underlying result is known to be integral.

1.11.5 XYZ determinant and the S3 conversion

- Using XYZ coordinates of the four vertices: see Eq. (10) for the determinant form and the S3 conversion to IVM units.

1.11.6 D^3 vs R^3: 60° "closing the lid" vs orthogonal "cubing"

- **IVM (D^3) heuristic**: From a 60–60–60 corner, three non-negative edge lengths A, B, C along quadray directions enclose a tetrahedron by "closing the lid." In synergetics, the tetravolume scales as the simple product ABC under IVM conventions (unit regular tetra has volume 1). By contrast, in the orthogonal (R^3) habit, one constructs a full parallelepiped (12 edges); the tetra occupies one-sixth of the triple product of edge vectors. The IVM path is more direct for tetrahedra.
- **Pedagogical note**: Adopt a vector-first approach. Differences like $(P_i - P_0)$ denote edge vectors; Quadrays and Cartesian can be taught in parallel as vector languages on the same Euclidean container.

Reference notebook with worked examples and code: See the Resources section for comprehensive educational materials and computational implementations.

See implementation: `tetra_volume_cayley_menger`.

- Lattice projection: round to nearest integer quadray; renormalize to maintain non-negativity and a minimal zero.

1.12 Code methods (anchors)

1.12.1 `integer_tetra_volume`

Source: `src/quadray.py` — integer 3×3 determinant for lattice tetravolume.

1.12.2 `ace_tetravolume_5x5`

Source: `src/quadray.py` — Tom Ace 5×5 determinant in IVM units.

1.12.3 `tetra_volume_cayley_menger`

Source: `src/cayley_menger.py` — length-based formula (XYZ units).

1.12.4 `ivm_tetra_volume_cayley_menger`

Source: `src/cayley_menger.py` — Cayley–Menger volume converted to IVM units.

1.12.5 `urner_embedding`

Source: `src/conversions.py` — canonical XYZ embedding.

1.12.6 `quadray_to_xyz`

Source: `src/conversions.py` — apply embedding matrix to map Quadray to XYZ.

1.12.7 `bareiss_determinant_int`

Source: `src/linalg_utils.py` — exact integer Bareiss determinant.

1.12.8 Information geometry methods (anchors)

`fisher_information_matrix` Source: `src/information.py` — empirical outer-product estimator.

`natural_gradient_step` Source: `src/information.py` — damped inverse-Fisher step.

`free_energy` Source: `src/information.py` — discrete-state variational free energy.

`discrete_ivm_descent` Source: `src/discrete_variational.py` — greedy integer-valued descent over the IVM using canonical neighbor moves; returns a `DiscretePath` with visited Quadrays and objective values. Pairs with `animate_discrete_path`.

`animate_discrete_path` Source: `src/visualize.py` — animate a `DiscretePath` to MP4; saves CSV/NPZ trajectory to `quadmath/output/`.

Relevant tests (`tests/`):

- `test_quadray.py` (unit IVM tetra, divisibility-by-4 scaling, Ace vs. integer method)
- `test_quadray_cov.py` (Ace determinant basic check)
- `test_cayley_menger.py` (regular tetra volume in XYZ units)
- `test_linalg_utils.py` (Bareiss determinant behavior)
- `test_examples.py`, `test_examples_cov.py` (neighbors, examples)
- `test_metrics.py`, `test_metrics_cov.py`, `test_information.py`, `test_paths.py`, `test_paths_cov.py`

1.13 Reproducibility checklist

- All formulas used in the paper are implemented in `src/` and verified by `tests/`.
- Determinants are computed with exact arithmetic for integer inputs; floating-point paths are used only where appropriate and results are converted (e.g., via S3) as specified.
- Random-walk experiments produce integer volumes; Ace 5×5 determinant agrees with length-based methods.
- Volume tracking: monitor integer simplex volume to detect convergence plateaus.
- Face/edge analyses: interpret sensitivity along edges; subspace searches across faces.