Equations and Math Supplement

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

August 15, 2025

Contents

1 Equations and Math Supplement (Appendix)

1.1 Volume of a Tetrahedron (Lattice)

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

Notes.

- $P_0, \ldots, P_3$ are vertex coordinates; the determinant computes the volume of the parallelepiped spanned by edge vectors, with the $1/6$ factor converting to tetra volume.

Tom Ace 5×5 tetravolume (IVM units):

$$V_{ivm} = \tfrac{1}{4}\left|\det\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & 1 \\ b_0 & b_1 & b_2 & b_3 & 1 \\ c_0 & c_1 & c_2 & c_3 & 1 \\ d_0 & d_1 & d_2 & d_3 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}\right| \tag{2}$$

Notes.

- Rows correspond to Quadray 4-tuples of the vertices; the last row encodes the affine constraint. Division by 4 returns IVM tetravolume.

XYZ determinant volume and S3 conversion:

$$V_{xyz} = \frac{1}{6}\left|\det\begin{pmatrix} x_a & y_a & z_a & 1 \\ x_b & y_b & z_b & 1 \\ x_c & y_c & z_c & 1 \\ x_d & y_d & z_d & 1 \end{pmatrix}\right|, \qquad V_{ivm} = S3\, V_{xyz}, \quad S3 = \sqrt{\frac{9}{8}} \tag{3}$$

Notes.

- Homogeneous determinant in Cartesian coordinates for tetra volume; conversion to IVM units uses $S3 = \sqrt{9/8}$ as used throughout.

See code: `tetra_volume_cayley_menger`. For tetrahedron volume background, see Tetrahedron – volume. Exact integer determinants in code use the Bareiss algorithm. External validation: these formulas align with implementations in `tetravolume.py` from the 4dsolutions ecosystem.

1.2 Fisher Information Matrix (FIM)

Background: Fisher information.

$$F_{i,j} = \mathbb{E}\left[\frac{\partial \log p(x;\theta)}{\partial \theta_i}\,\frac{\partial \log p(x;\theta)}{\partial \theta_j}\right] \tag{4}$$

Notes.

- Defines the Fisher information matrix as the expected outer product of score functions; see Fisher information.

Figure: empirical estimate shown in the FIM heatmap figure. See code: `fisher_information_matrix`.

See `src/information.py` — empirical outer-product estimator (`fisher_information_matrix`).

1.3 Natural Gradient

Background: Natural gradient (Amari).

$$\theta \leftarrow \theta - \eta\, F(\theta)^{-1}\, \nabla_\theta L(\theta) \tag{5}$$

Explanation.

- Natural gradient update: right-precondition the gradient by the inverse of the Fisher metric (Amari); see Natural gradient.

See code: `natural_gradient_step`.

See `src/information.py` — damped inverse-Fisher step (`natural_gradient_step`).

1.4 Free Energy (Active Inference)

$$\mathcal{F} = -\log P(o \mid s) + \mathrm{KL}\big[Q(s) \,\|\, P(s)\big] \tag{6}$$

Explanation.

- **Partition**: variational free energy decomposes into expected negative log-likelihood and KL between approximate posterior and prior; see Free energy principle.

See code: `free_energy`.

See `src/information.py` — discrete-state variational free energy (`free_energy`).

1.4.1 Figures

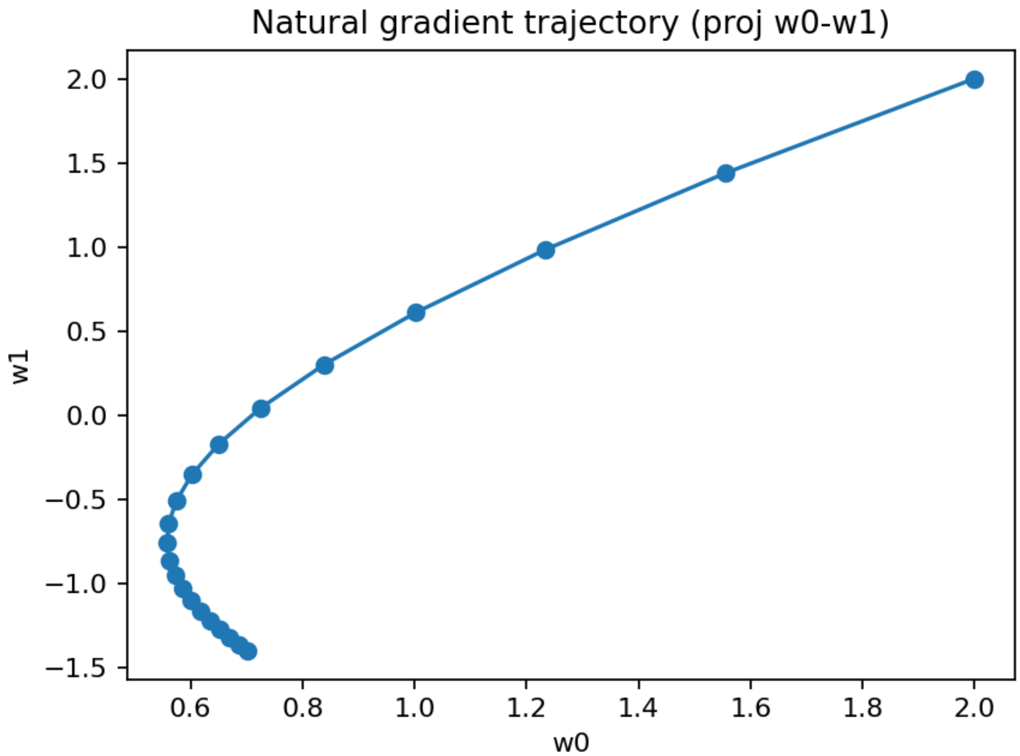

Figure 1: **Figure 14: Natural gradient trajectory demonstrating information-geometric optimization**. This trajectory shows natural gradient descent (Eq. 5) converging on a quadratic objective, projected to the (w_0, w_1) parameter plane. **Mathematical setup**: Quadratic form matrix $A = \begin{bmatrix} 3 & 0.5 & 0 \\ 0.5 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, step size $\eta = 0.5$, damped Fisher inverse $F + 10^{-3} I$ for numerical stability. **Trajectory characteristics**: The curved path demonstrates curvature-adaptive steps—larger strides in low-curvature directions, smaller steps in high-curvature directions—contrasting with uniform Euclidean gradient steps. **Information geometry**: Each step follows approximate geodesics on the parameter manifold equipped with the Fisher metric, achieving more efficient convergence than standard gradient descent on ill-conditioned problems. **Data artifacts**: Complete 3D trajectory data saved as `natural_gradient_path.csv` and `natural_gradient_path.npz` for reproducibility and further analysis.

1.5 Quadray Normalization (Fuller.4D)

Given $q = (a, b, c, d)$, choose $k = \min(a, b, c, d)$ and set $q' = q - (k, k, k, k)$ to enforce at least one zero with non-negative entries.

1.6 Distance (Embedding Sketch; Coxeter.4D slice)

Choose linear map M from quadray to $\mathbb{R}^3$ (or $\mathbb{R}^4$) consistent with tetrahedral axes; then $d(q_1, q_2) = \|M(q_1) - M(q_2)\|_2$.

1.7 Minkowski Line Element (Einstein.4D analogy)

$$ds^2 = -c^2\, dt^2 + dx^2 + dy^2 + dz^2 \tag{7}$$

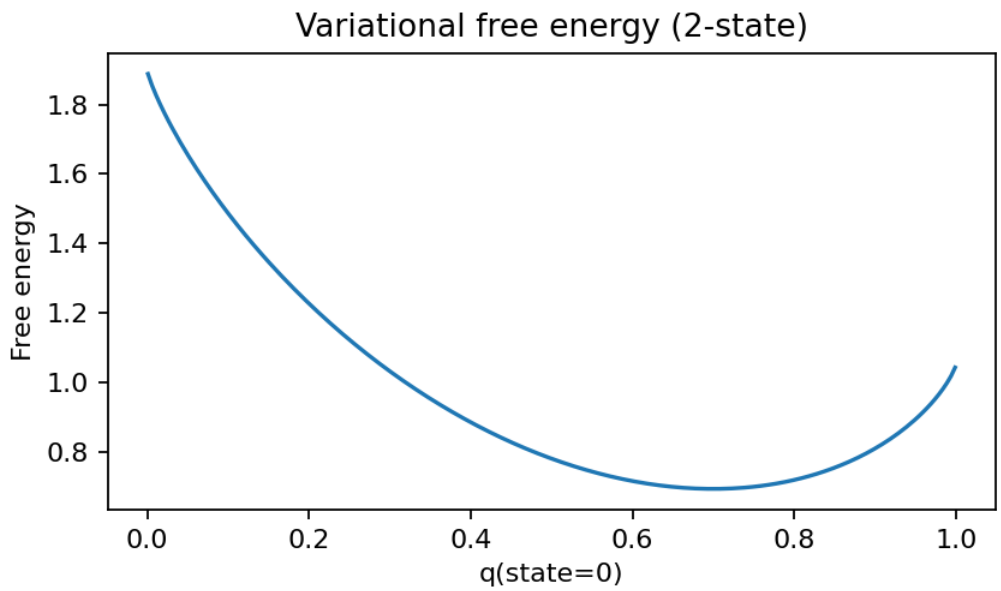

Figure 2: **Figure 15: Variational free energy functional for discrete binary states (Eq. 6)**. This curve illustrates the free energy landscape $\mathcal{F} = -\log P(o|s) + \mathrm{KL}[Q(s)||P(s)]$ for a 2-state system as a function of variational posterior probability $q(\text{state} = 0) \in [0.001, 0.999]$. **Model specification**: True likelihood $\log P(o|s) = \log[0.7, 0.3]$, uniform prior $P(s) = [0.5, 0.5]$, variational posterior $Q(s) = [q, 1-q]$. **Free energy interpretation**: The convex curve shows the trade-off between likelihood accuracy (observation explanation) and complexity penalty (KL divergence from prior). **Optimization**: The global minimum represents the optimal variational approximation where beliefs match the true posterior distribution. **Active Inference**: This functional drives belief updating in the four-fold partition framework, with the minimum achieved through gradient-based inference or discrete lattice descent methods. The convex structure ensures reliable convergence for variational optimization in discrete state spaces.

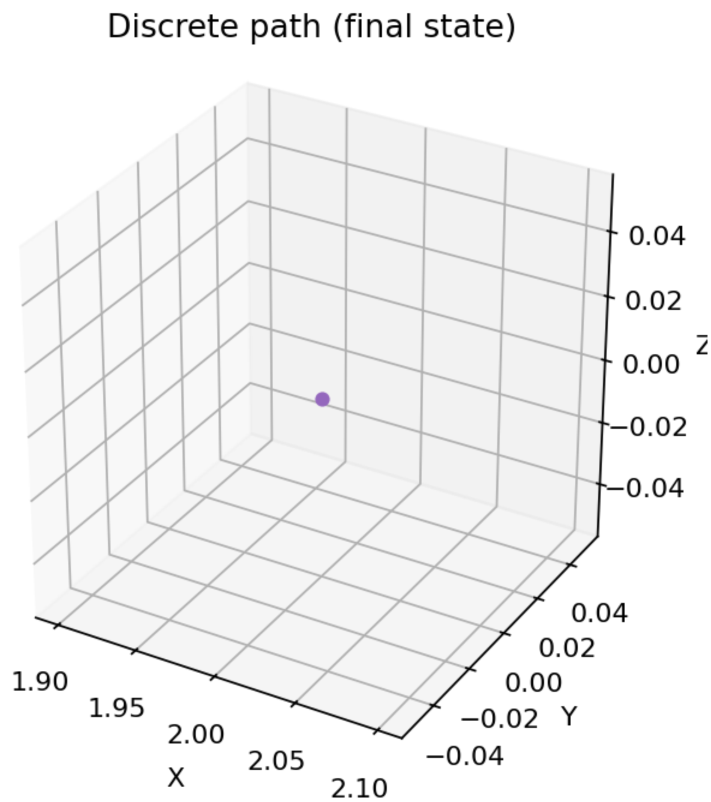

Figure 3: **Figure 16: Discrete IVM descent optimization path (final converged state)**. This static frame shows the final position of a discrete variational descent algorithm operating on the integer Quadray lattice. **Algorithm**: `discrete_ivm_descent` performs greedy optimization using the 12 canonical IVM neighbor moves (permutations of {2,1,1,0}), ensuring all iterates remain on integer lattice points with proper Quadray normalization. **Objective**: Simple quadratic function $f(q) = (x - 0.5)^2 + (y + 0.2)^2 + (z - 0.1)^2$ where (x, y, z) are the embedded Euclidean coordinates of Quadray q. **Convergence**: The final point represents the best lattice approximation to the continuous optimum, demonstrating discrete convergence within the integer-constrained feasible region. **Fuller.4D significance**: This method exemplifies optimization directly on the Quadray integer lattice without continuous relaxation, maintaining exact arithmetic and leveraging the discrete "energy level" structure of integer tetravolumes. **Animation**: The complete optimization trajectory is available as `discrete_path.mp4` with corresponding trajectory data in `discrete_path.csv` and `discrete_path.npz`.

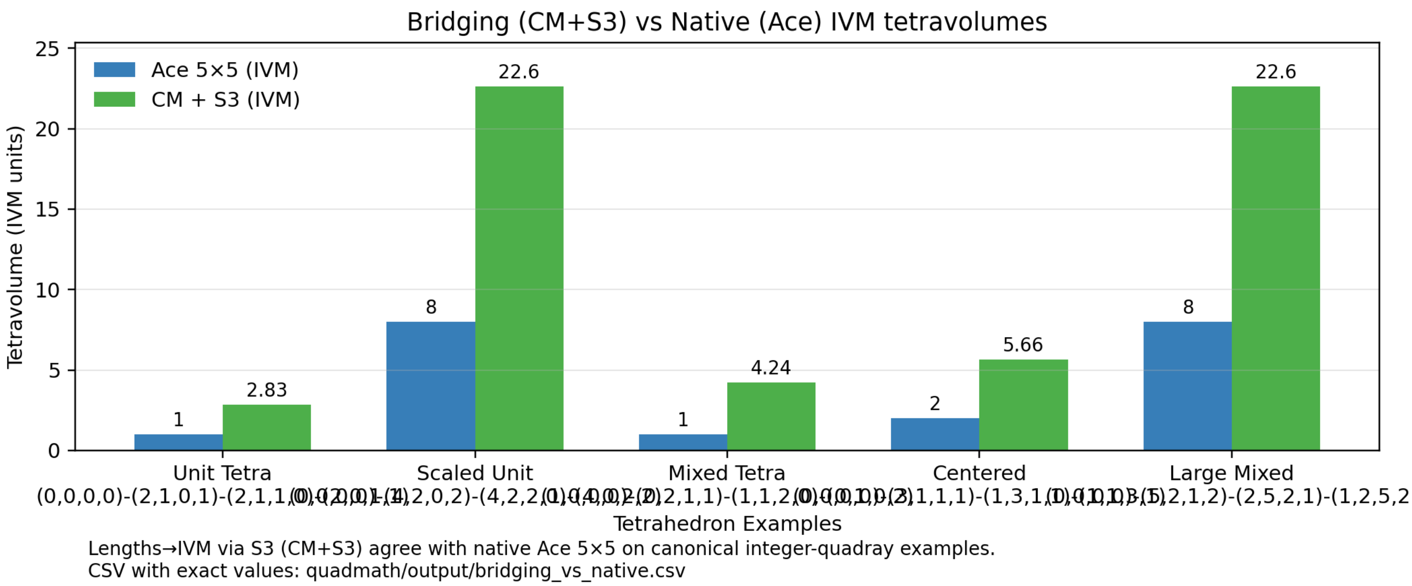

Figure 4: **Figure 17: Bridging (CM+S3) vs Native (Ace) IVM tetravolumes across canonical integer-quadray examples**. Bars compare V_{ivm} computed via Cayley–Menger on XYZ edge lengths with $S3 = \sqrt{9/8}$ conversion (bridging) against Tom Ace's native 5×5 determinant (IVM). The overlaid bars coincide to numerical precision, illustrating the equivalence of length-based and Quadray-native formulations under synergetics units. Source CSV: `bridging_vs_native.csv`.

Background: Minkowski space.

1.8 High-Precision Arithmetic Note

When evaluating determinants, FIMs, or geodesic distances for sensitive problems, use quad precision (binary128) via GCC's `libquadmath` (`__float128`, functions like `expq`, `sqrtq`, and `quadmath_snprintf`). See GCC libquadmath. Where possible, it is useful to use symbolic math libraries like SymPy to compute exact values.

1.8.1 Reproducibility artifacts and external validation

- **This manuscript's artifacts**: Raw data in `quadmath/output/` for reproducibility and downstream analysis:
  - `fisher_information_matrix.csv` / `.npz`: empirical Fisher matrix and inputs
  - `fisher_information_eigenvalues.csv` / `fisher_information_eigensystem.npz`: eigenspectrum and eigenvectors
  - `natural_gradient_path.png` with `natural_gradient_path.csv` / `.npz`: projected trajectory and raw coordinates
  - `bridging_vs_native.csv`: Ace 5×5 vs CM+S3 tetravolume comparisons
  - `ivm_neighbors_data.csv` / `ivm_neighbors_edges_data.npz`: neighbor coordinates (Quadray and XYZ)
  - `polyhedra_quadray_constructions.png`: synergetics volume relationships schematic
- **External validation resources**: The 4dsolutions ecosystem provides extensive cross-validation:
  - `Qvolume.ipynb`: Independent Tom Ace 5×5 implementations with random-walk demonstrations
  - `VolumeTalk.ipynb`: Comparative tetravolume algorithm analysis
  - Cross-language implementations in Rust and Clojure for algorithmic verification

1.9 Namespaces summary (notation)

- Coxeter.4D: Euclidean E^4; regular polytopes; not spacetime (cf. Coxeter, Regular Polytopes, Dover ed., p. 119). Connections to higher-dimensional lattices and packings as in Conway & Sloane.
- Einstein.4D: Minkowski spacetime; indefinite metric; used here only as a metric analogy when discussing geodesics and information geometry.
- Fuller.4D: Quadrays/IVM; tetrahedral lattice with integer tetravolume; unit regular tetrahedron has volume 1; synergetics scale relations (e.g., S3).