Optimization in 4D

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

August 14, 2025

Optimization in 4D (Namespaces and Quadray-Lattice Methods)

Here we review the optimization methods used in this manuscript. We review the Nelder–Mead method, which is a simple and robust optimization method that is well-suited to the Quadray 4D lattice. We also review the discrete IVM descent method, which is a more sophisticated optimization method that is well-suited to the Quadray lattice as well.

Quadray-Adaptive Nelder–Mead (Fuller.4D)

- Initialization: choose 4 integer quadray vertices forming a non-degenerate simplex (e.g., basis + one mixed point such as (1,1,1,0)).
- Reflection/Expansion/Contraction: compute candidate; round to nearest integer; renormalize by adding/subtracting (k,k,k,k) to enforce non-negativity and at least one zero.
- Shrink: discrete contraction of all vertices toward the best vertex along tetrahedral axes.

Standard Nelder–Mead coefficients (typical choices):

- **Reflection** $\alpha = 1$
- **Expansion** $\gamma \approx 2$
- **Contraction** $\rho \approx 0.5$
- **Shrink** $\sigma \approx 0.5$

References: original Nelder–Mead method and common parameterizations in optimization texts and survey articles; see overview: Nelder–Mead method.

Volume-Level Dynamics

- Simplex volume decreases in discrete integer steps, creating stable plateaus (“energy levels”).

- Termination: when volume stabilizes at a minimal level and function spread is below tolerance.
- Monitoring: track integer simplex volume and the objective spread at each iteration for convergence diagnostics.

Pseudocode (Sketch)

```
while not converged:
  order vertices by objective
  centroid of best three
  propose reflected (then possibly expanded/contracted) point
  project to integer quadray; renormalize with (k,k,k,k)
  accept per standard tests; else shrink toward best
  update integer volume and function spread trackers
```

Figures

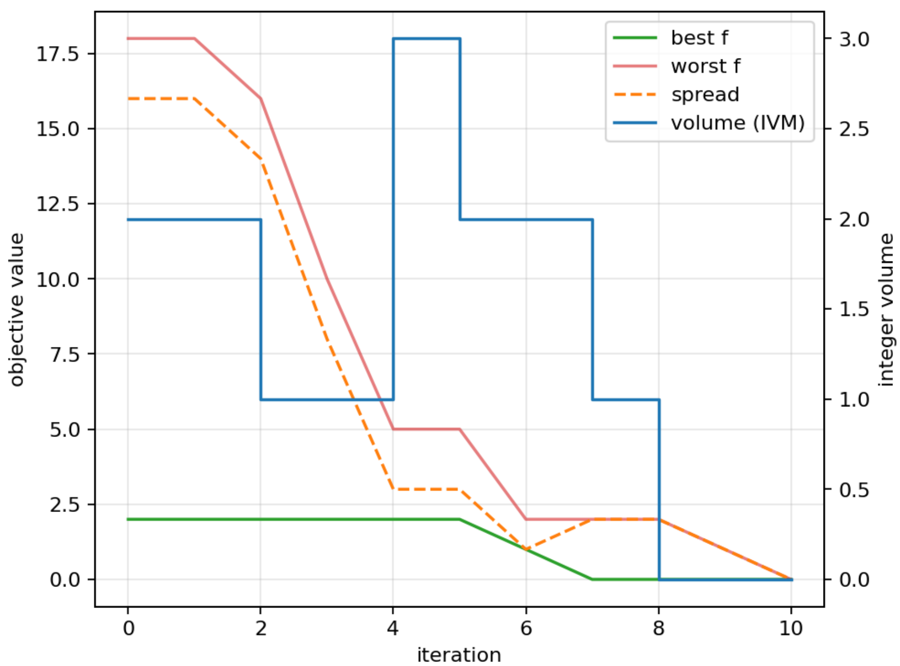

Figure 1: Optimization trace for discrete Nelder–Mead in the Quadray lattice. Best/worst objective values and spread (left axis) with integer tetra-volume (right axis) per iteration. See the MP4 for the full simplex trajectory.

As shown in Fig. 3, the discrete Nelder–Mead converges on plateaus; Fig. 2 summarizes the scaling behavior used in volume diagnostics.

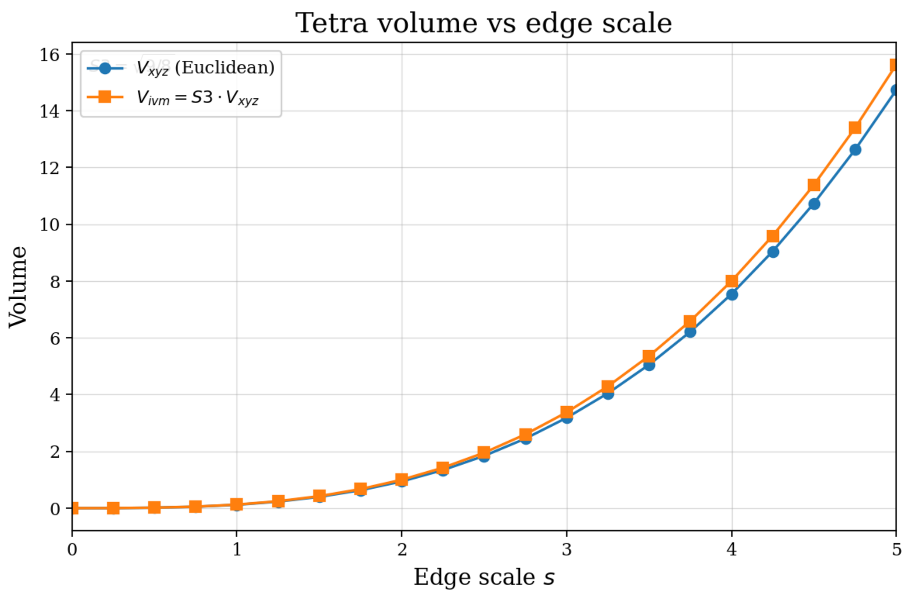

Figure 2: Tetra volume vs edge scale. Two curves: Euclidean volume V_{xyz} and IVM-converted $V_{ivm} = S3 \cdot V_{xyz}$; axes labeled; $S3$ annotated; data saved as CSV/NPZ.

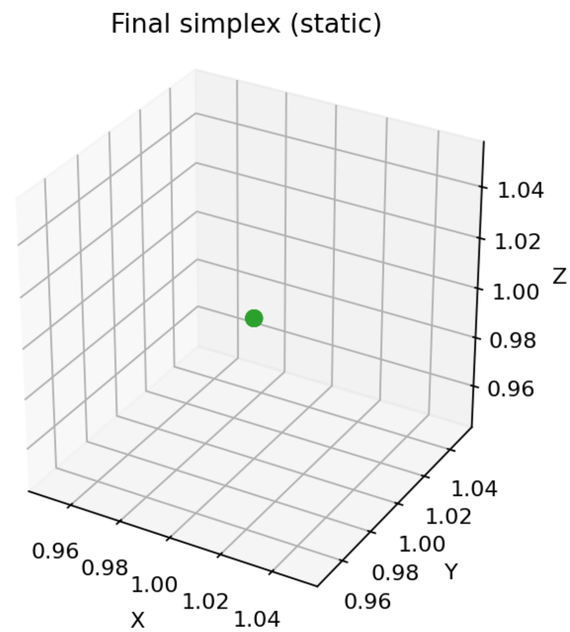

Figure 3: Final converged simplex configuration.

Raw artifacts: the full trajectory animation `simplex_animation.mp4` and per-frame vertices (`simplex_animation_vertices.csv`/`.npz`) are available in `quadmath/output/`. The full optimization trajectory is provided as an animation (MP4) in the repository's output directory.

Discrete Lattice Descent (Information-Theoretic Variant)

- Integer-valued descent over the IVM using the 12 neighbor moves (permutations of {2,1,1,0}), snapping to the canonical representative via projective normalization.
- Objective can be geometric (e.g., Euclidean in an embedding) or information-theoretic (e.g., local free-energy proxy); monotone decrease is guaranteed by greedy selection.
- API: `discrete_ivm_descent` in `src/discrete_variational.py`. Animation helper: `animate_discrete_path` in `src/visualize.py`.

Short snippet (paper reproducibility):

```
from quadray import Quadray, DEFAULT_EMBEDDING, to_xyz
from discrete_variational import discrete_ivm_descent
from visualize import animate_discrete_path

def f(q: Quadray) -> float:
    x, y, z = to_xyz(q, DEFAULT_EMBEDDING)
    return (x - 0.5)**2 + (y + 0.2)**2 + (z - 0.1)**2

path = discrete_ivm_descent(f, Quadray(6,0,0,0))
animate_discrete_path(path)
```

Convergence and Robustness

- Discrete steps reduce numerical drift; improved stability vs. unconstrained Cartesian.
- Natural regularization from volume quantization; fewer wasted evaluations.
- Compatible with Gauss–Newton/Natural Gradient guidance using FIM for metric-aware steps (Amari, natural gradient).

Information-Geometric View (Einstein.4D analogy in metric form)

- **Fisher Information as metric**: use the empirical estimator `F = (1/N) \sum g g^\top` from `fisher_information_matrix` to analyze curvature of the objective with respect to parameters. See Fisher information.
- **Curvature directions**: leading eigenvalues/eigenvectors of `F` (see `fim_eigenspectrum`) reveal stiff and sloppy directions; this supports step-size selection and preconditioning.

- **Figures**: empirical FIM heatmap (Fig. 4) and eigenspectrum (Fig. 5). Raw data available as NPZ/CSV in `quadmath/output/`.

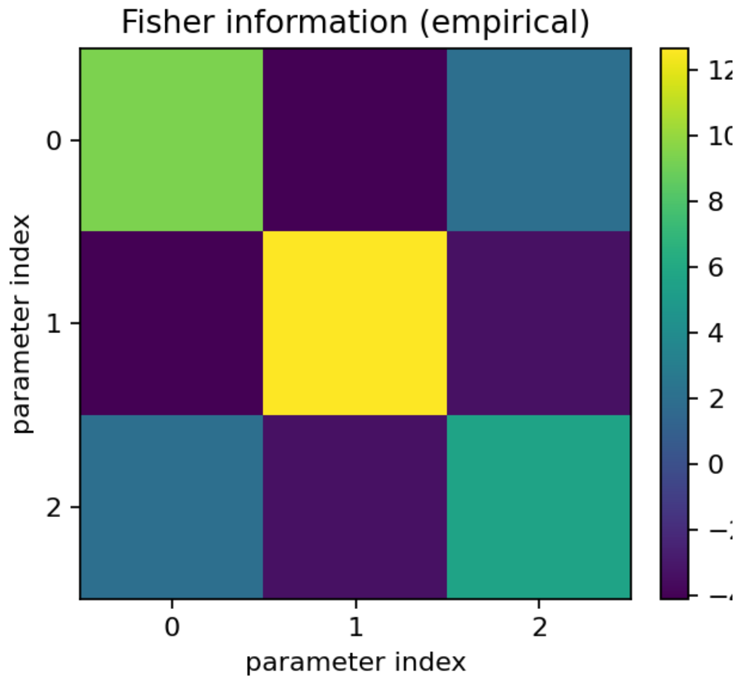

Figure 4: Empirical Fisher information heatmap (entries F_{ij} estimated via outer products of per-sample gradients; model: noisy linear regression; evaluated at misspecified parameters w_{est} vs ground truth w_{true}; colorbar shows curvature scale).

- **Quadray relevance**: block-structured and symmetric patterns often arise under quadray parameterizations, simplifying `F` inversion for natural-gradient steps.

Multi-Objective and Higher-Dimensional Notes (Coxeter.4D perspective)

- Multi-objective: vertices encode trade-offs; simplex faces approximate Pareto surfaces; integer volume measures solution diversity.
- Higher dimensions: decompose higher-dimensional simplexes into tetrahedra; sum integer volumes to extend quantization.

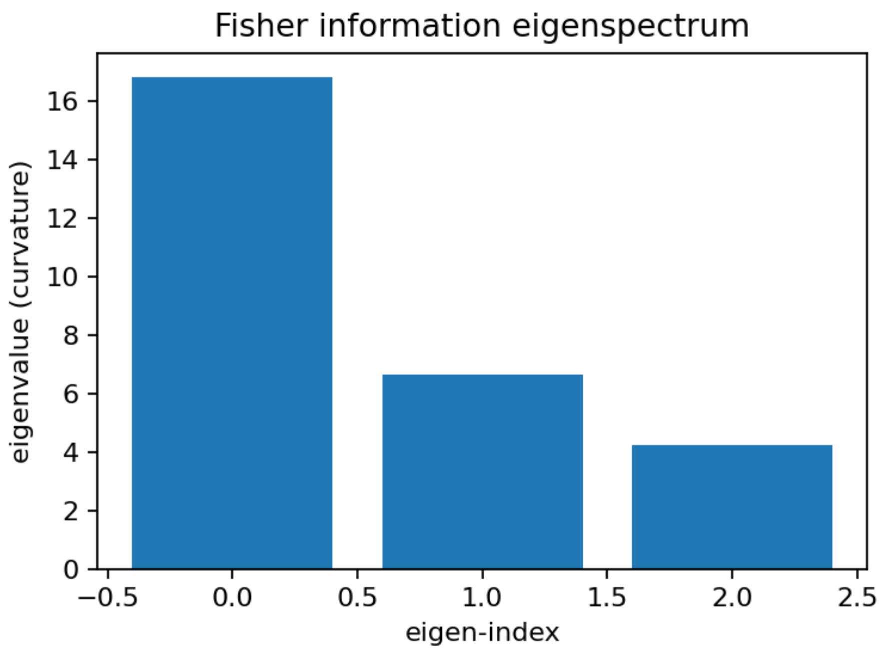

Figure 5: Fisher information eigenspectrum (principal curvatures along eigenvectors of F; eigenvalues λ_i sorted descending; highlights stiff vs. sloppy directions).

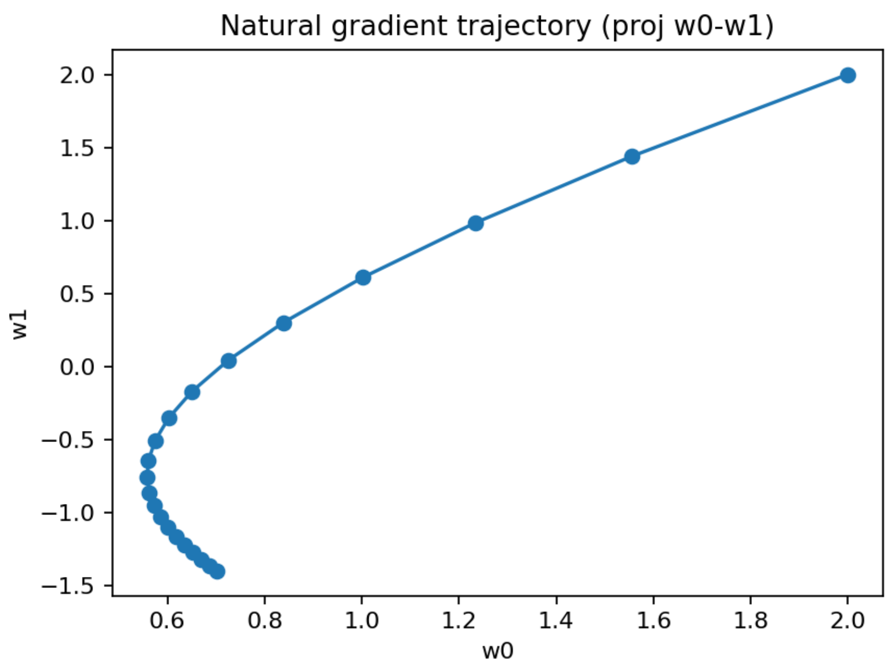

Figure 6: Natural gradient trajectory on a quadratic bowl (projection in w_0–w_1 plane); $A = \begin{bmatrix} 3 & 0.5 & 0 \\ 0.5 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, step size $\eta = 0.5$, damped inverse Fisher $F + 10^{-3}I$; raw path in CSV/NPZ.

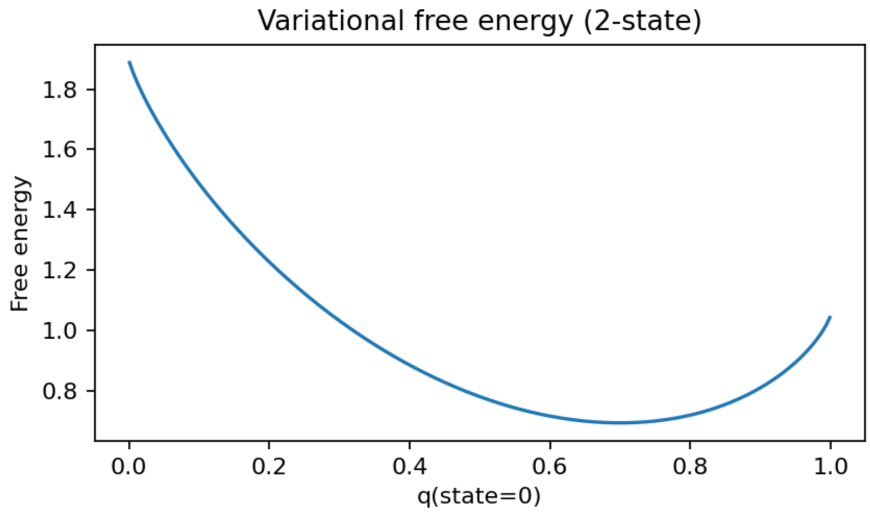

Figure 7: Free energy curve for a 2-state model.

4dsolutions optimization context and educational implementations

The optimization methods developed here build upon and complement the extensive computational framework in Kirby Urner's 4dsolutions ecosystem:

- **Algorithmic foundations**: Our `nelder_mead_quadray` and `discrete_ivm_descent` methods extend the vector operations and volume calculations implemented in `qrays.py` and `tetravolume.py`.
- **Educational precedents**: Interactive optimization demonstrations appear in School_of_Tomorrow notebooks, particularly volume tracking and CCP navigation in `QuadCraft_Project.ipynb`.
- **Cross-platform validation**: Independent implementations in Rust and Clojure provide performance baselines and algorithmic verification for optimization primitives.

Results

- The simplex-based optimizer exhibits discrete volume plateaus and converges to low-spread configurations; see Fig. 3 and the MP4/CSV artifacts in `quadmath/output/`.
- The greedy IVM descent produces monotone trajectories with integer-valued objectives; see Fig. **??**.