Optimization in 4D

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

August 15, 2025

Contents

1 Optimization in 4D

1.1 Overview

This section describes optimization methods adapted to the integer Quadray lattice, emphasizing discrete convergence and information-geometric approaches. The methods leverage the IVM's natural quantization and extend to higher-dimensional spaces via Coxeter.4D embeddings.

1.2 Nelder–Mead on Integer Lattice

- **Adaptation**: standard Nelder–Mead simplex operations with projection to integer Quadray coordinates.
- **Projection**: after each reflection/expansion/contraction, snap to nearest integer lattice point via projective normalization.
- **Volume tracking**: monitor integer tetravolume as convergence diagnostic; discrete steps create stable plateaus.

1.2.1 Parameters

- **Reflection** $\alpha \approx 1$
- **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.

1.3 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.

1.4 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
```

1.4.1 Figures

As shown in Figure 9, the discrete Nelder–Mead converges on plateaus; Figure 8 summarizes the scaling behavior used in volume diagnostics.

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.

1.5 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)
```

1.6 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).

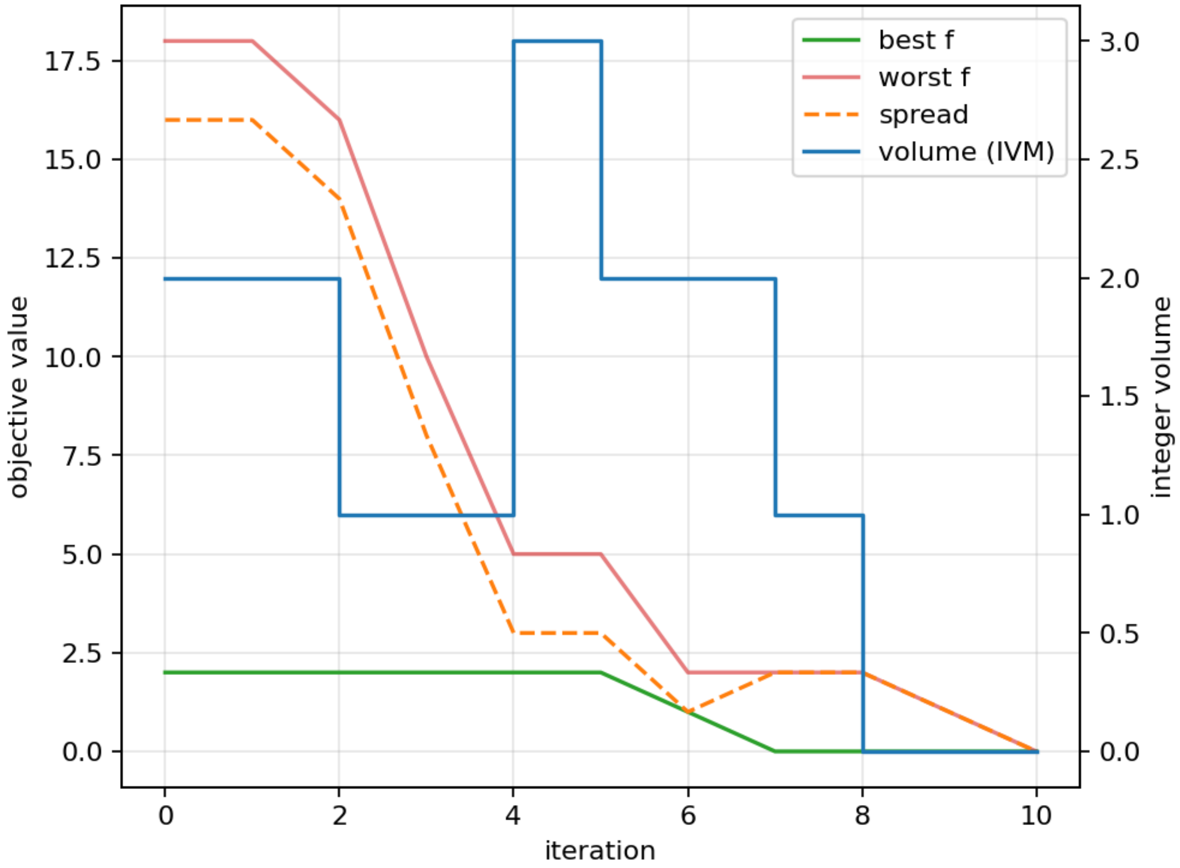

Figure 1: **Figure 7: Discrete Nelder–Mead optimization trajectory on the integer Quadray lattice**. This time-series plot tracks key diagnostic quantities across 12 optimization iterations for a simple quadratic objective $f(q) = (q.a - 1)^2 + q.b^2 + q.c^2 + q.d^2$ starting from initial simplex vertices $\{(5,0,0,0),(4,1,0,0),(0,4,1,0),(1,1,1,0)\}$. **Left y-axis (objective values)**: Blue line shows the best (minimum) objective value per iteration, demonstrating monotonic improvement as the simplex converges toward the minimum at $(1,0,0,0)$. Orange line shows the worst (maximum) objective value among the four simplex vertices. **Right y-axis (simplex volume)**: Green line tracks the integer tetravolume of the current simplex computed via Tom Ace's 5×5 determinant, showing characteristic discrete plateaus and step-wise reductions as the simplex contracts on the lattice. **Convergence signature**: The volume decreases in discrete integer steps, creating stable "energy levels" that regularize the optimization process, while the objective spread (difference between best and worst) narrows as vertices cluster near the optimum. The full 3D simplex trajectory animation is available as `simplex_animation.mp4` in the output directory.

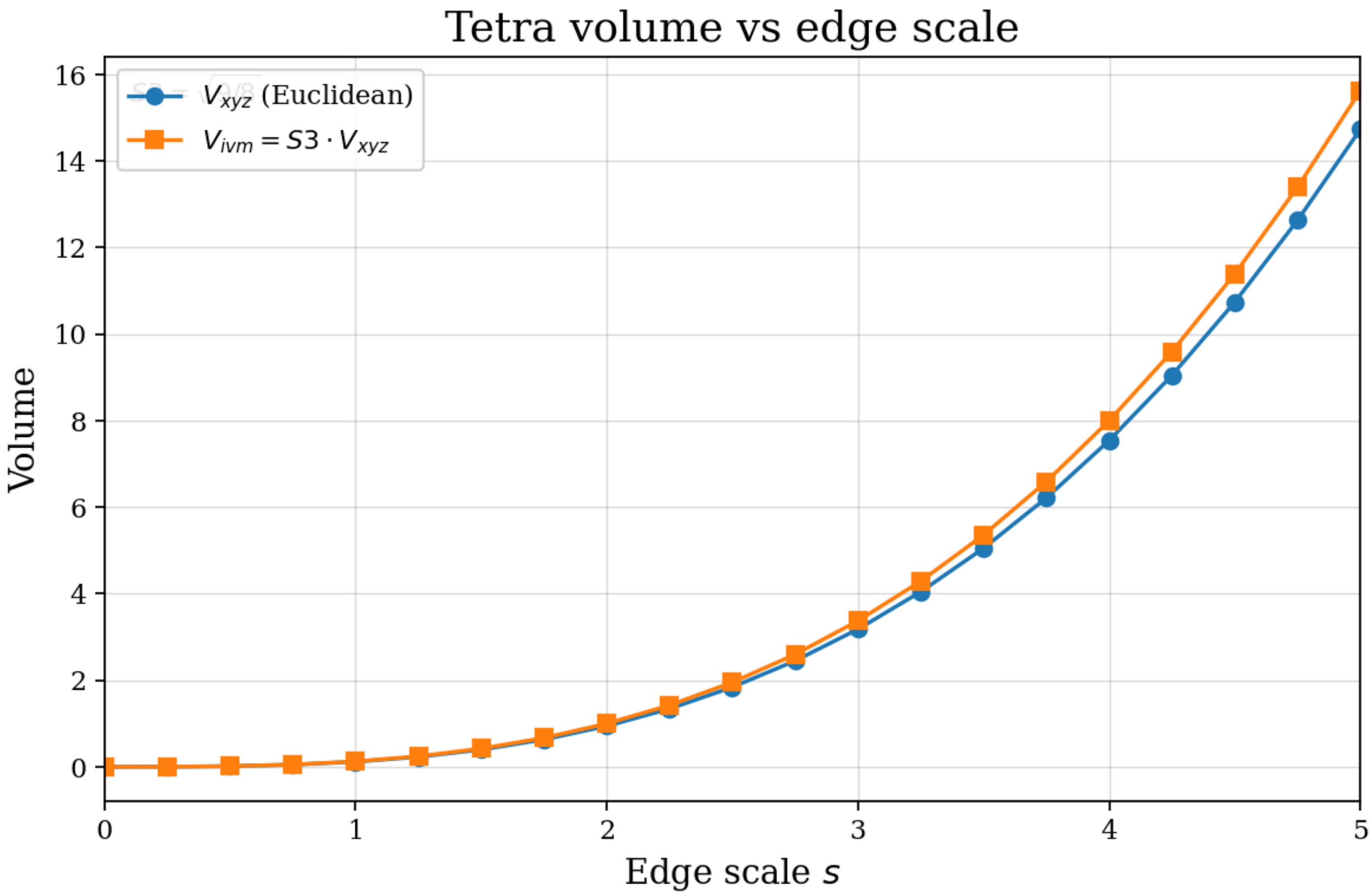

Figure 2: **Figure 8: Tetrahedron volume scaling relationships: Euclidean vs IVM unit conventions**. This plot demonstrates the mathematical relationship between edge length scaling and tetravolume under both Euclidean (Coxeter.4D) and synergetics (Fuller.4D) unit conventions. **X-axis**: Edge scale factor ranging from 0.5 to 2.0 applied to a regular tetrahedron. **Y-axis**: Computed tetravolume in respective units. **Blue curve** (V_{xyz}): Euclidean tetravolume computed via standard geometric formulas, showing the expected cubic scaling relationship $V \propto \text{edge}^3$. **Orange curve** (V_{ivm}): IVM tetravolume obtained by converting the Euclidean volume via the synergetics factor $S3 = \sqrt{9/8} \approx 1.061$, following the relationship $V_{ivm} = S3 \cdot V_{xyz}$. **Scaling verification**: Both curves maintain their proportional relationship across all scales, confirming the consistency of the S3 conversion factor used throughout the manuscript to bridge between Coxeter.4D (Euclidean) and Fuller.4D (IVM) volume measurements. The parallel cubic curves validate the unit conversion methods employed in bridging vs native tetravolume comparisons. Raw numerical data available as `volumes_scale_data.csv` and `volumes_scale_data.npz`.

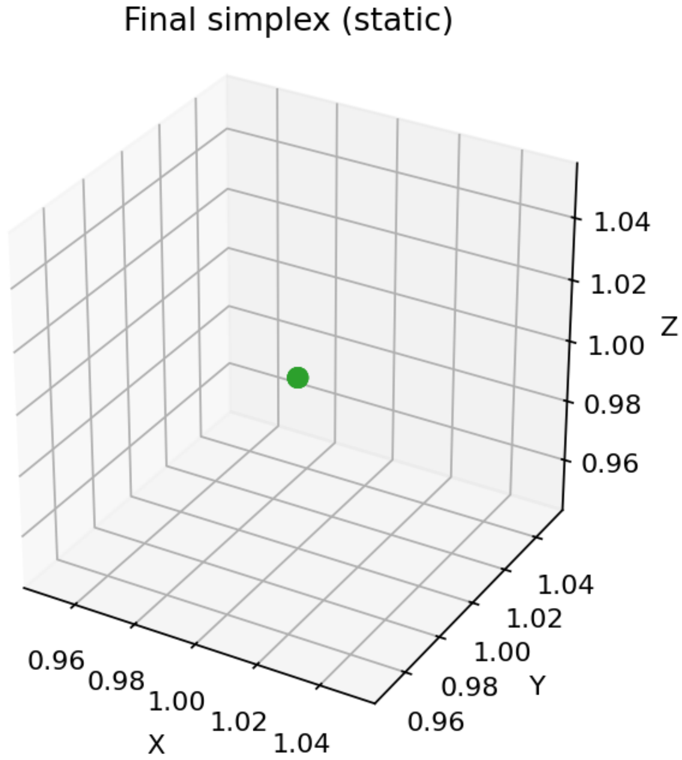

Figure 3: **Figure 9: Final converged simplex configuration in 3D embedding space**. This 3D scatter plot shows the four vertices of the Nelder–Mead simplex after 12 iterations of discrete optimization on the integer Quadray lattice, projected into Euclidean 3D space via the default embedding matrix. The green points represent the converged simplex vertices clustered near the objective minimum, connected by green lines to emphasize the tetrahedral structure. All vertices are constrained to integer Quadray coordinates and maintain the projective normalization (at least one zero component). The tight clustering demonstrates successful convergence while the discrete lattice constraint ensures numerical stability. This static view complements the dynamic trajectory shown in the full animation (`simplex_animation.mp4`) and the diagnostic traces in Figure 7. The final simplex volume is minimal on the integer lattice, representing a stable "energy level" where further discrete moves do not improve the objective function.

1.7 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 (Figure 10) and eigenspectrum (Figure 11). Raw data available as NPZ/CSV in `quadmath/output/`.

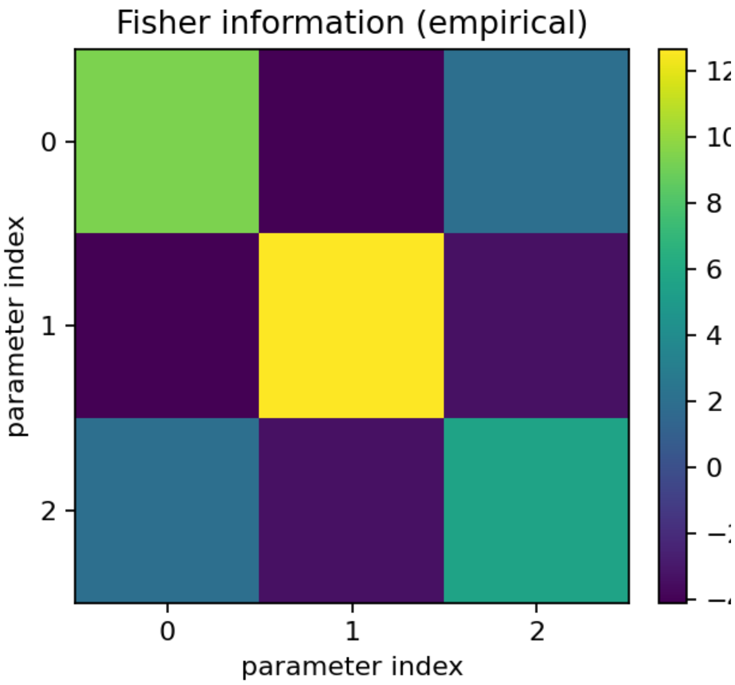

Figure 4: **Figure 10: Empirical Fisher Information Matrix (FIM) for noisy linear regression**. This heatmap visualizes the 3×3 Fisher information matrix F_{ij} estimated from per-sample gradients of a misspecified linear regression model. **Setup**: Ground truth parameters $w_{\text{true}} = [1.0, -2.0, 0.5]$, evaluated at estimation point $w_{\text{est}} = [0.3, -1.2, 0.0]$, with 200 samples and Gaussian noise (σ=0.1). **Matrix elements**: Each F_{ij} entry represents the expected outer product $\mathbb{E}[\partial_i \log p \cdot \partial_j \log p]$ where gradients are computed from squared loss with respect to model parameters. **Interpretation**: The colorbar scale shows local curvature magnitudes—brighter entries indicate directions of higher sensitivity/information content. Diagonal dominance suggests the parameters are approximately decoupled at this evaluation point. **Information geometry**: This FIM serves as a Riemannian metric tensor for natural gradient descent (see Eq. (**??**) in the equations appendix), enabling curvature-aware optimization steps that adapt to the local geometry of the parameter manifold. Raw matrix data saved as `fisher_information_matrix.csv` and `fisher_information_matrix.npz` for reproducibility.

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

1.8 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.

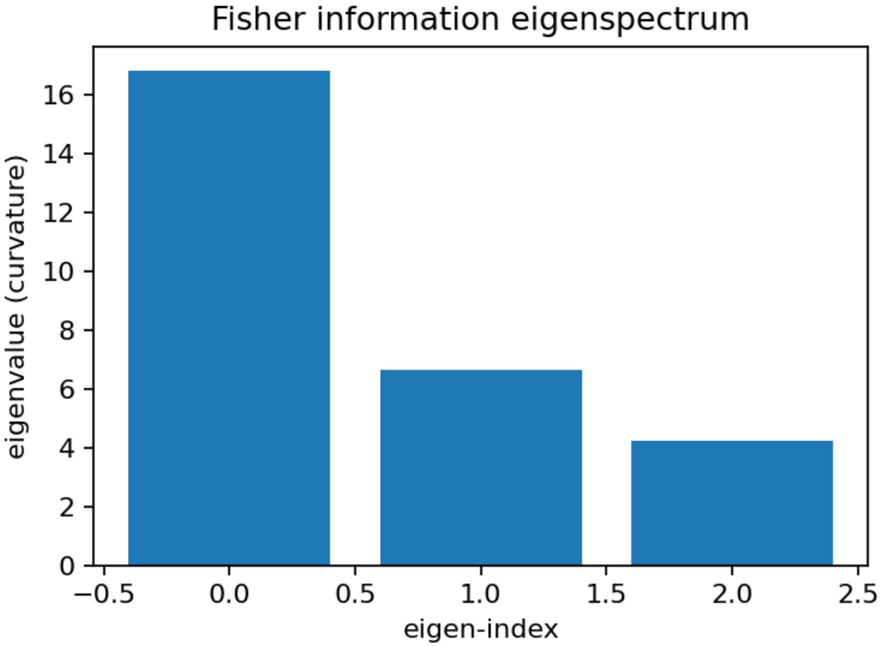

Figure 5: **Figure 11: Fisher Information Matrix eigenspectrum: principal curvature directions**. This bar chart displays the eigenvalue decomposition of the empirical Fisher information matrix from Figure 10, revealing the principal curvature directions of the parameter manifold. **X-axis**: Eigenvalue indices (0, 1, 2) sorted in descending order of magnitude. **Y-axis**: Eigenvalue magnitudes representing the curvature strength along corresponding eigenvector directions. **Interpretation**: Large eigenvalues indicate "stiff" parameter directions where small changes significantly affect the objective function, while small eigenvalues correspond to "sloppy" directions with minimal impact. **Information geometry insights**: The eigenspectrum reveals the conditioning of the FIM and guides natural gradient preconditioning—directions with high curvature (large λ_i) require smaller step sizes, while low-curvature directions tolerate larger updates. **Optimization implications**: The eigenvalue spread suggests the degree of parameter coupling and optimal step-size scaling for each principal direction. Well-conditioned problems show uniform eigenvalues, while ill-conditioned problems exhibit large eigenvalue spreads requiring careful preconditioning. Raw eigenvalue data available as `fisher_information_eigenvalues.csv` and `fisher_information_eigensystem.npz`.

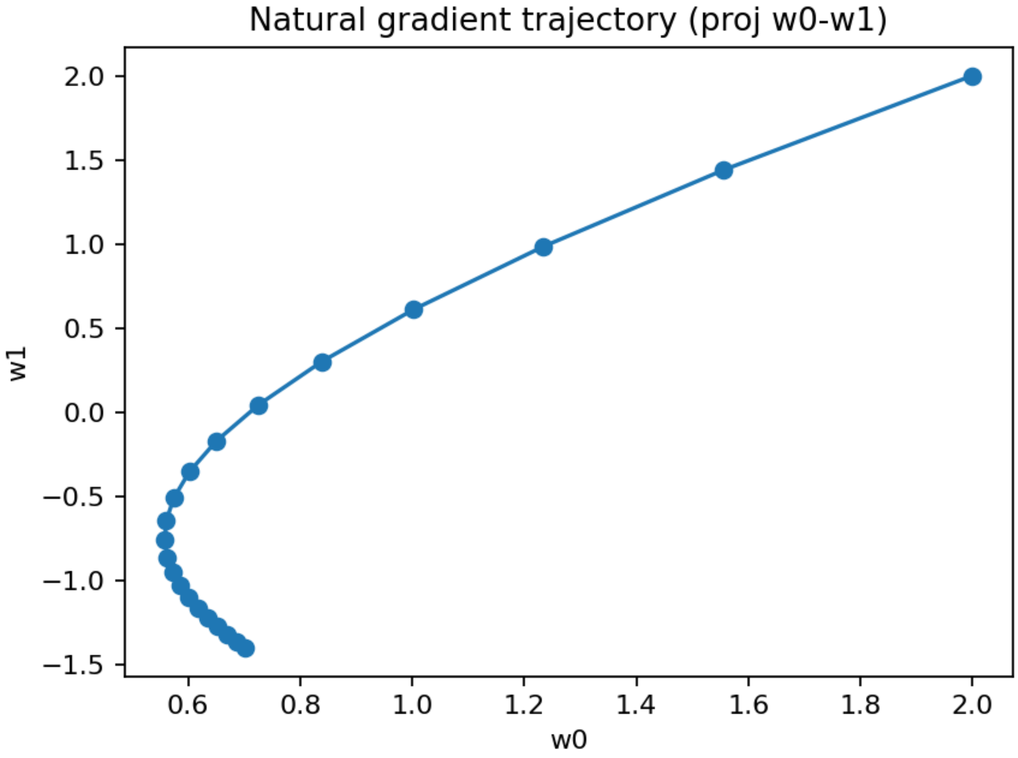

Figure 6: **Figure 12: Natural gradient descent trajectory on a quadratic objective (2D projection)**. This line plot with markers shows the parameter trajectory of natural gradient descent converging to the optimum of a quadratic bowl-shaped objective function. **Setup**: Starting point $(w_0, w_1) = (2, 2)$ with target $(w_0, w_1) = (1, -2)$; quadratic form defined by matrix $A = \begin{bmatrix} 3 & 0.5 & 0 \\ 0.5 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$; step size $\eta = 0.5$; regularized Fisher matrix $F + 10^{-3}I$ for numerical stability. **Trajectory analysis**: The curved path demonstrates how natural gradient descent (see Eq. (**??**) in the equations appendix) adapts to the local curvature structure, taking larger steps in low-curvature directions and smaller steps in high-curvature directions compared to vanilla gradient descent. **Information geometry**: The trajectory follows approximate geodesics on the parameter manifold equipped with the Fisher metric, resulting in more efficient convergence than Euclidean gradient descent on ill-conditioned problems. **Projection note**: This visualization shows the (w_0, w_1) projection of the full 3D parameter trajectory. Each marker represents one optimization step, with the curvature-aware steps visible as the adaptive stride lengths along the path. Complete trajectory data saved as `natural_gradient_path.csv` and `natural_gradient_path.npz`.

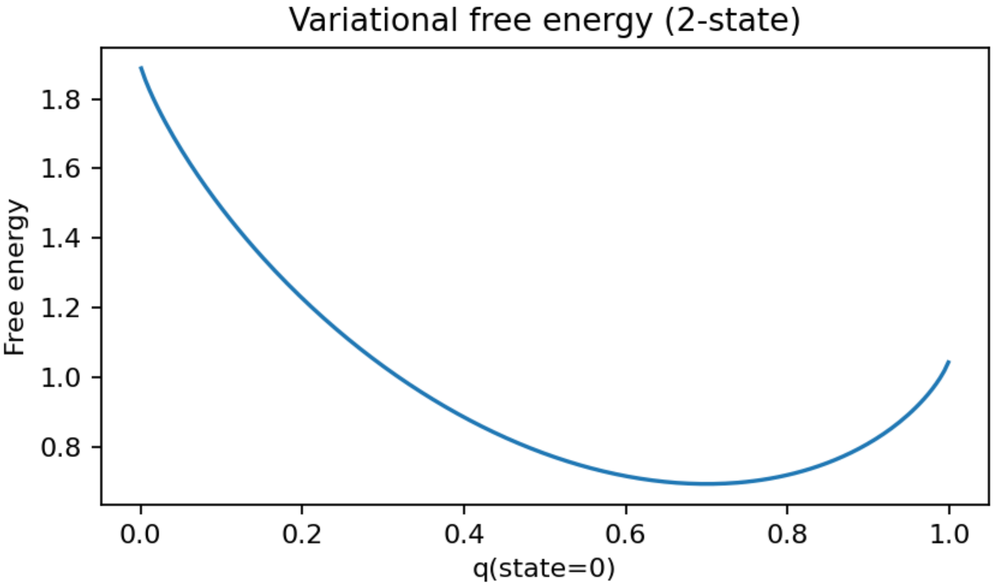

Figure 7: **Figure 13: Variational free energy landscape for a discrete 2-state system**. This curve shows the variational free energy $\mathcal{F} = -\log P(o|s) + \mathrm{KL}[Q(s)||P(s)]$ (see Eq. (**??**) in the equations appendix) as a function of the variational posterior probability $q(\text{state} = 0)$ for a simple binary system. **Setup**: True observation probabilities log $P(o|s) = \log[0.7, 0.3]$ and uniform prior $P(s) = [0.5, 0.5]$; variational posterior $Q(s) = [q, 1 - q]$ parameterized by $q \in [0.001, 0.999]$. **Free energy decomposition**: The curve reflects the balance between likelihood accuracy (how well Q explains observations) and KL complexity penalty (deviation from prior beliefs). **Minimum**: The global minimum occurs where the variational posterior matches the true posterior, achieving optimal trade-off between accuracy and complexity. **Active Inference connection**: In the context of the four-fold partition (see the Active Inference appendix), this free energy functional drives both perceptual inference (belief updates) and action selection (environmental steering). **Optimization**: The convex shape enables gradient-based minimization for belief updating, with the minimum representing the optimal variational approximation. This toy example illustrates the general principle underlying variational optimization in active inference and the free energy minimization framework.

1.9 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.

1.10 Results

- The simplex-based optimizer exhibits discrete volume plateaus and converges to low-spread configurations; see Figure 9 and the MP4/CSV artifacts in `quadmath/output/`.
- The greedy IVM descent produces monotone trajectories with integer-valued objectives; see Figure 16.