Overview

- Notebook builds a lightweight, fully analytic demo of a "liquid" neuron (single-step LTC) trained to invert noise on synthetic expectation values, entirely with NumPy and manual gradients, no quantum runtime or autograd involved (notebooks/demo_math.ipynb:1).

Data Pipeline

- Synthetic targets come from drawing random angles theta in [0, 2pi) and mapping to ideal expectations y = cos(theta) (notebooks/demo_math.ipynb:38-45).
- Noisy observations u are produced via an affine shrink + bias plus Gaussian shot noise, then clipped back to the physical interval [-1, 1], mimicking amplitude damping toward +1 (notebooks/demo_math.ipynb:46-55).
- Dataset is shuffled and split 80/20 into train/val splits (notebooks/demo_math.ipynb:56-60).

Model Structure

- The forward routine implements a single LTC update: starting from x0 = u, it computes the gating term g = tanh(wr*x0 + w*u + b), evolves one Euler step x1 = (x0 + dt*g*A)/(1 + dt*(1/tau + g)), and projects the readout through tanh(v*x1 + c) to stay within observable bounds (notebooks/demo_math.ipynb:62-82).
- Helper mse returns batch mean-squared error (notebooks/demo_math.ipynb:81).

Manual Gradients

- gradients caches activations from forward, then backpropagates analytically:
  - Starts with dL/dyhat = 2*(yhat - y)/N for the batch (notebooks/demo_math.ipynb:84-91).
  - Uses tanh' = 1 - tanh^2 to differentiate the output stage for parameters v and c (notebooks/demo_math.ipynb:92-101).
  - Applies quotient-rule sensitivities for x1 = num/den with respect to g, A, and tau, including the tau-only term in the denominator (notebooks/demo_math.ipynb:102-124).
  - Chains g-derivatives back to weights w, wr, b while aggregating over the batch (notebooks/demo_math.ipynb:125-135).
  - Returns the 7-parameter gradient vector alongside the latest predictions (notebooks/demo_math.ipynb:134-136).

Training Loop

- Parameters are initialized near zero with tau > 0, learning rate 0.02, 30 epochs, batch size 200, and updated via vanilla SGD (params -= lr * grads) with a post-update projection enforcing tau >= 0.05 (notebooks/demo_math.ipynb:138-160).
- After each epoch it logs train/val MSE using fresh forward passes, storing the curves for plotting (notebooks/demo_math.ipynb:161-166).
- A Matplotlib cell plots and saves training_loss_math.png, illustrating convergence of both splits (notebooks/demo_math.ipynb:167-178).

Validation Analysis

- A second cell compares mean absolute error on raw noisy validation data versus the mitigated predictions, printing both metrics and saving a bar chart (math_mae_comparison.png) that quantifies the improvement (notebooks/demo_math.ipynb:221-234).
- The final cell simply echoes y_clean, leaving the clean targets in the output for quick

inspection (notebooks/demo_math.ipynb:257).