Q1 Exact: The Same Example Through $E \to \mathbb{N}$

Claude and MJC

11 February 2026

Abstract

We repeat the Protocol C worked example (position $t = 42$, predicting `m`) using exact arithmetic via the $E \to \mathbb{N}$ prime encoding and GMP. Each section defines a program, states a prediction from the f32 channel analysis, and reports the result. This lets us measure exactly where and how much entropy bleeds through the f32 quotient.

1 Program 1: Forward Pass at $t = 42$

What we compute. Run the sat-rnn forward pass from $t = 0$ to $t = 42$ twice: once in f32 (as the RNN actually does) and once in GMP rational arithmetic (exact). Record:

- h_{42}^{f32} and h_{42}^{exact}: the hidden state at position 42.
- $\Delta h_{42} = h_{42}^{\text{exact}} - h_{42}^{\text{f32}}$: the accumulated f32 error after 42 timesteps.
- Per-neuron: $|\Delta h_{42}[j]|$ and $|\Delta h_{42}[j]|/|h_{42}[j]|$ (absolute and relative error).
- The sign vector: does $\text{sgn}(h_{42}^{\text{f32}}) = \text{sgn}(h_{42}^{\text{exact}})$? Any sign disagreement means a binary ES decision flipped.

Prediction 1 (Sign agreement)**.** *All 128 sign bits agree. The sign channel survives 42 steps of f32 because the sat-rnn is saturated: most $|h_j|$ are near 1, so the sign is determined with large margin. Sign flips would require the f32 error to exceed the neuron's distance from zero, which is $\sim$1 for saturated neurons vs $\sim 10^{-5}$ accumulated error.*

Prediction 2 (Exponent agreement)**.** *All 128 exponents agree. The exponent of h_j is determined by* tanh*: for $|h_j| > 0.5$ (most neurons), the exponent is $E = 126$ (value in $[0.5, 1)$). The f32 error $\sim 10^{-5}$ cannot shift the value across a power-of-2 boundary.*

Prediction 3 (Mantissa divergence)**.** *The mantissa diverges by $\sim$5–7 bits (of 23). Each of the 42 forward steps mixes 128 multiply-add operations, each introducing $\sim 2^{-24}$ relative error. The error grows as $\sqrt{42 \times 128} \cdot 2^{-24} \approx 73 \cdot 2^{-24} \approx 4 \times 10^{-6}$ (random walk). For $|h_j| \sim 0.9$, this is $\sim$4 ppm, or $\sim$18 correct mantissa bits out of 23.*

Tool. `q1_exact_p1.c`: loads model weights as GMP rationals (exact conversion from f32 bit pattern), loads data, runs forward pass in both f32 and GMP, reports per-neuron comparison at $t = 42$.

2 Program 2: Output Gradient at $t = 42$

What we compute. Compute g_{42} in both f32 and GMP:

$$[g_{42}]_j = W_y[109, j] - \sum_{o=0}^{255} W_y[o, j] \cdot P_{42}(o)$$

where P_{42} is computed from h_{42} (f32 or exact respectively).

Report:

- Per-neuron: $[g_{42}]_j^{\text{f32}}$ vs $[g_{42}]_j^{\text{exact}}$.
- Channel decomposition: sign agreement, exponent agreement, mantissa divergence.
- The entropy content of g_{42}: how many bits of the 23-bit mantissa carry signal vs noise?

Prediction 4 (Gradient sign agreement)**.** *All 128 signs agree. The gradient sign determines whether neuron j helps or hurts the prediction. The f32 error in h_{42} propagates through softmax and the W_y subtraction, but the gradient magnitudes ($\sim$0.1–2) are much larger than the propagated error ($\sim 10^{-5}$).*

Prediction 5 (Gradient entropy)**.** *The gradient g_{42} has $\sim$15–18 bits of mantissa entropy per component. The softmax sum over 256 terms introduces $\sim$8 bits of mixing (each $P(o)$ contributes $\sim 2^{-8}$ to the sum on average). The subtraction $W_y[y, j]$ − sum may cancel, reducing mantissa precision by the number of cancelled bits. For neurons where the gradient is large (> 0.5), $\sim$18 bits are valid. For small gradients (< 0.01), $\sim$10 bits.*

Tool. `q1_exact_p2.c`: extends P1 to compute the output gradient in both representations.

3 Program 3: One BPTT Step ($d = 0 \to 1$)

What we compute. Compute $g_{42,1}$ in both f32 and GMP:

$$[g_{42,1}]_j = \sum_{k=0}^{127} (1 - h_k(42)^2) \cdot W_h[k, j] \cdot [g_{42}]_k$$

Report per-neuron comparison and channel decomposition.

Prediction 6 (Exponent-level saturation filtering)**.** *Of the 128 terms in the sum for each j, we predict $\sim$60–80 are killed by saturation gates (exponent of $(1 - h_k^2)$ drops below $\sim E = 100$, i.e. gate $< 10^{-8}$). The surviving $\sim$50–70 terms determine the sum.*

Prediction 7 (Mantissa loss from cancellation)**.** *Among the surviving terms, $\sim$half are positive and $\sim$half negative (the gradient has mixed signs). If the positive and negative sums are of similar magnitude, the cancellation loses $\sim$3–5 mantissa bits. Result: $g_{42,1}$ has $\sim$12–15 valid mantissa bits in f32, vs 23+ in GMP.*

Prediction 8 (Gradient norm ratio)**.** $\|g_{42,1}\|/\|g_{42}\| \approx 0.23$ *(matching Protocol A depth profile). The 77% loss is entropy bleeding through saturation (exponent channel) and cancellation (mantissa channel).*

Tool. `q1_exact_p3.c`: one Jacobian step in both representations.

4 Program 4: Entropy Smuggling in the Jacobian ($d = 1 \to 50$)

What we compute. The Jacobian $J_s = \text{diag}(1 - h_s^2) \cdot W_h$ propagates gradient backward. We compute the accumulated product $\prod_{k=0}^{d-1} J_{t-k}$ in both f32 and GMP for $d = 1$ to 50. At each depth we report the singular values, the per-neuron gradient decomposed into sign/exponent/mantissa, and—crucially—the *cross-channel mutual information* between these three fields at consecutive depths.

Bit-position weighting. Each of the 32 bits in an f32 value has a different dynamical leverage: how much entropy it can inject into the chain per Jacobian step. Multiply each bit by its position index (0–31) to weight by leverage. This gives a scalar per f32 value—the effective dynamical entropy—which we can then factor two ways:

1. **Hardware factorization (exp/mant):** group bits 31 (sign), 30–23 (exponent), 22–0 (mantissa). This is the representation-level decomposition.
2. **Dynamical factorization (stable/unstable):** for each bit position b across all 128 neurons, compare the bit in f32 and exact at depth d. A bit is *stable* at depth d if it agrees between f32 and exact across all 128 neurons (or $> 90\%$ of them). Otherwise unstable.

If the channels were independent, these two factorizations would align: exponent bits = stable, mantissa bits = unstable. The degree to which they *don't* align is the entropy smuggling.

Why channels are not independent. Over a single Jacobian step, the channels nearly decouple. Over a longer chain in u_{sup}, they couple through two mechanisms:

1. **Gate → exponent shift.** For $h_j = 0.998$, the gate $(1 - h_j^2) = 0.004$ (exponent 117). A mantissa-level change to $h_j = 0.999$ gives gate = 0.002 (exponent 116). Mantissa at depth d becomes exponent at depth d+1.
2. **Mix → cancellation.** The 128-term dot product $\sum_i J_{ij} g_i$ has positive and negative terms. The exponent of the result depends on mantissa-level details of the summands (whether they cancel or not).

Over d steps this creates feedback: mantissa noise at step k shifts an exponent at k+1, changes a gate at k+2, flips a sign at k+5. Entropy smuggles across channels.

Prediction 9 (The two factorizations diverge with depth)**.** *At $d = 1$: stable bits ≈ exponent bits. The hardware factorization works. At $d = 10$: ∼3–5 mantissa bits are stable (locked to the dynamics) and ∼1–2 exponent bits are unstable (near power-of-2 boundaries). At $d = 30$: the overlap between the two factorizations drops below 70%—nearly a third of the "stable" bits are mantissa bits, and some exponent bits are unstable. The hardware factorization is no longer the right decomposition.*

Prediction 10 (Jacobian singular values: top agree, bottom diverge)**.** *The top 5–10 singular values of $\prod J$ agree between f32 and exact. The bottom singular values hit the f32 noise floor at $\sim d \cdot 2^{-23}$. When a singular value crosses this floor, its entropy does not vanish—it redistributes into cross-channel correlations among the surviving directions.*

Prediction 11 (Fewer effective chains in f32)**.** *Mantissa-to-exponent smuggling randomly kills chains (gate exponent drops below threshold) and randomly creates others (gate opens). The net effect is fewer chains: the saturation gates are mostly closed, so random perturbation is more likely to close a marginal gate than open one. We predict ∼10–20% fewer surviving chains at $d = 30$ in f32 vs exact.*

Tool. `q1_exact_p4.c`: Jacobian product in both representations, SVD at each depth, bit-level stability analysis (both factorizations), cross-channel MI.

5 Program 5: Pattern Attribution Comparison

What we compute. Using the exact and f32 backward gradients, compute per-pattern attributions (as in Protocol A) and compare:

- For each pattern p, compute $a^{\text{f32}}(p)$ and $a^{\text{exact}}(p)$.
- Rank correlation: do f32 and exact agree on which patterns are most important?
- Threshold agreement: at $\tau = 0.01$, do they agree on which patterns are active?
- The "phantom patterns": patterns active in f32 but not in exact (or vice versa).

Prediction 12 (High rank correlation for strong patterns)**.** *For the top 50 patterns (by attribution), the rank correlation between f32 and exact is > 0.95. The strong patterns have large attributions ($\gg$ f32 error), so the ranking is robust.*

Prediction 13 (Threshold disagreement for weak patterns)**.** *At $\tau = 0.01$, ∼5–20% of the active pattern set differs between f32 and exact. These are patterns near the threshold whose attributions are comparable to the f32 error. At $\tau = 0.1$, disagreement drops to $< 5\%$.*

Prediction 14 (Phantom patterns at depth)**.** *Most phantom patterns (f32-only or exact-only) come from deep offsets ($d > 20$). At shallow offsets ($d \leq 5$), there are no phantoms because f32 precision is sufficient. This confirms that the entropy bleeding from Protocol C's channel analysis is* measurably real*: the f32 quotient creates and destroys patterns at depth.*

Tool. `q1_exact_p5.c`: pattern attribution in both representations, with comparison.

6 Program 6: Entropy of the Quotient

What we compute. The exact UM and the f32 UM make different predictions at each position (because h_t diverges). Compute:

- $H^{\text{exact}}(t)$ and $H^{\text{f32}}(t)$: per-position output entropy.
- $\text{bpc}^{\text{exact}}$ and bpc^{f32}: overall compression.
- $D_{\text{KL}}(P_t^{\text{exact}} \| P_t^{\text{f32}})$: per-position KL divergence.

Prediction 15 (bpc difference $< 10^{-4}$)**.** *The f32 quotient costs $< 10^{-4}$ bpc. The entropy difference is negligible because the strong patterns (which dominate the predictions) are represented accurately in f32. The quotient's cost is paid only by weak patterns at deep offsets, which contribute < 0.001 bpc.*

Prediction 16 (KL spikes at specific positions)**.** *D_{KL} is near zero at most positions but spikes at positions where the prediction depends on deep patterns ($d > 20$). These are the positions where f32 entropy bleeding changes the prediction.*

Tool. `q1_exact_p6.c`: full forward pass and prediction in both representations, entropy comparison.

Implementation Note

All programs use GMP (`libgmp`) for exact rational arithmetic. The model weights are f32 values, which are exactly representable as rationals ($w = m \cdot 2^e$ for integer m and exponent e). The tanh function is computed via its Taylor series to sufficient precision, or via $\tanh(x) = 1 - 2/(e^{2x} + 1)$ using GMP's `mpfr_exp`. The softmax is computed in exact rationals.

The programs are independent and can be run in any order; each one tests specific predictions from the f32 channel analysis in Protocol C.