Projection-Induced Memory in Reversible Dynamical Systems: Depth, Strength, and an Information Conservation Law

Kaifan XIE

2026.02.28

Abstract

We study memory induced by coarse-grained observation of finite reversible dynamical systems. Given a bijective map F on a finite state space X and a non-injective projection $\pi : X \to Y$, the projected process $\{Y_t\}$ may exhibit non-Markovian statistical memory even though the microscopic dynamics are perfectly reversible. We define a scale-resolved penetration measure $I_k = I(Y_{t+k+1} ; Y_t \mid Y_{t+1}, \ldots, Y_{t+k})$ and the associated memory depth $D = \sup\{k : I_k > 0\}$. We construct explicit residual-window register systems achieving arbitrary finite depth k for any modular scale $m \geq 2$, prove that in finite reversible systems $D \leq N - 2$ (where $N = |X|$), show this bound is tight, and establish an information conservation identity: the total penetration strength $\sum I_k = H(Y_0)$, bounded above by $\log_2 m$. This reveals that increasing memory depth merely redistributes a fixed information budget across scales.

1. Introduction

A fundamental question in statistical physics and information theory is how macroscopic memory arises from microscopic dynamics. When a reversible (bijective) microscopic system is observed through a coarse-grained projection, the resulting macroscopic process may exhibit statistical dependencies extending over multiple time steps—even though the underlying dynamics are perfectly deterministic and reversible.

This phenomenon is well-known in the Mori–Zwanzig formalism and hidden Markov model theory, but explicit, constructive characterizations of the relationship between projection structure and memory depth have been lacking. In this paper, we provide:

- A rigorous definition of memory depth via conditional mutual information penetration
- An explicit parametric construction (the residual window register) achieving any desired finite memory depth
- A tight upper bound $D \leq N - 2$ for finite reversible systems
- An example (the even shift) achieving infinite memory depth with reversible dynamics
- An information conservation identity: total penetration strength equals $H(Y_0)$

Convention on time direction. We adopt a forward-looking definition of penetration depth: $I_k = I(Y_{t+k+1} ; Y_t \mid Y_{t+1:t+k})$, measuring how past information penetrates forward through a future window. This differs from the backward-looking convention $I(Y_{t-k-1} ; Y_t \mid Y_{t-1:t-k})$ used in some prior work. For stationary processes the two are equivalent by time-reversal symmetry of mutual information.

2. Setup and Definitions

2.1 Microscopic System

Let X be a finite set with $|X| = N$. Let $F : X \to X$ be a bijection (permutation). The pair (X, F) defines a reversible discrete dynamical system with trajectory $X_{t+1} = F(X_t)$.

2.2 Projection and Macroscopic Process

Let $\pi : X \to Y$ be a (generally non-injective) map with $|Y| = m$. Define the macroscopic process $Y_t = \pi(X_t)$. We call π the *observation projection* and m the *observation scale*.

2.3 Memory Depth (Definition D1)

For integer $k \geq 0$, define the *penetration measure*:

$$I_k := I(Y_{t+k+1} ; Y_t \mid Y_{t+1}, \ldots, Y_{t+k})$$

where $I(\cdot ; \cdot \mid \cdot)$ denotes conditional mutual information under the stationary distribution. Define the *memory depth*:

$$D := \sup\{ k \geq 0 : I_k > 0 \}$$

If the set is unbounded, $D = \infty$. Intuitively, I_k measures how much information from Y_t penetrates

through a blocking window of length k to influence Y_{t+k+1}.

3. The Residual Window Register Construction

3.1 Setup

Given scale $m \geq 2$ and target depth $k \geq 1$, set $X = (\mathbb{Z}_m)^{k+1}$ with states $s = (x^{(0)}, x^{(1)}, \ldots, x^{(k)})$. Choose $a \in \mathbb{Z}_m^*$ ($\gcd(a, m) = 1$) and an arbitrary function $g : (\mathbb{Z}_m)^k \to \mathbb{Z}_m$.

3.2 Dynamics

Define the reversible map:

$$F(x^{(0)}, \ldots, x^{(k)}) = (x^{(1)}, \ldots, x^{(k)}, a \cdot x^{(0)} + g(x^{(1)}, \ldots, x^{(k)})) \pmod m$$

with observation $\pi(s) = x^{(0)} \in \mathbb{Z}_m$ (the residual projection).

3.3 Core Theorems

Theorem 1 (Reversibility). For any g, if $a \in \mathbb{Z}_m^*$, then F is a bijection. The inverse is given by $x^{(0)} = a^{-1}(u^{(k)} - g(u^{(0)}, \ldots, u^{(k-1)})) \pmod m$, with $x^{(i)} = u^{(i-1)}$ for $i \geq 1$.

Theorem 2 (Window Recurrence). Under the above construction, $Y_{t+k+1} \equiv a \cdot Y_t + g(Y_{t+1}, \ldots, Y_{t+k}) \pmod m$. This is the explicit memory kernel.

Theorem 3 (k-order Markovianity). Under the stationary distribution, $\{Y_t\}$ is k-order Markov. The microscopic state at time t equals $(Y_t, Y_{t+1}, \ldots, Y_{t+k})$, so the window of length k+1 is a sufficient statistic.

Theorem 4 (Strictness). The process is not (k−1)-order Markov. Since $Y_{t+k+1} = a \cdot Y_t + c$ (where c depends only on the window), knowing Y_t determines Y_{t+k+1} given the window. Without Y_t, the conditional distribution changes.

Theorem 5 (Penetration Identity). For this construction, $I_k = H(Y_t \mid Y_{t+1}, \ldots, Y_{t+k})$. If the conditional entropy is positive, then $I_k > 0$.

4. Numerical Example: m = 3, k = 2

Set $m = 3$, $k = 2$, $a = 2$, $g(y_1, y_2) = y_1 + 2y_2 \pmod 3$. The state space has $|X| = 27$ states. The window recurrence becomes:

$$Y_{t+3} = 2Y_t + Y_{t+1} + 2Y_{t+2} \pmod 3$$

Exact computation over all 27 states yields:

k	I_k (bits)	Expected	Match
0	0.000000	0	✓
1	$-2.66 \times 10^{-15} \approx 0$	0	✓
2	1.584963	$\log_2 3 = 1.584963$	✓

3	0.000000	0	✓

Memory depth D = 2, exactly as constructed. The microscopic dynamics are fully reversible.

5. Maximum Memory Depth in Finite Reversible Systems

5.1 Synchronization Length

For a cyclic word w of period p over alphabet Y, define the *synchronization length* L(w) as the smallest L ≥ 1 such that for all phases i, j: if the future blocks of length L at positions i and j are identical, then $w_i = w_j$. Then D(w) = L(w) − 1.

5.2 The Tight Bound

Theorem 6 (Upper Bound). For any cyclic word w of period p, L(w) ≤ p − 1, and therefore D ≤ p − 2.

Proof sketch. Consider the future block $B_i = (w_{i+1}, \ldots, w_{i+p-1})$ of length p − 1. If $B_i = B_j$, then the cyclic word shifted by j − i matches itself on p − 1 consecutive positions, forcing $w_i = w_j$. Hence L = p − 1 always suffices. □

Theorem 7 (Achievability). The single-marker word w = 000…0001 achieves L(w) = p − 1, so D = p − 2. For the full system with |X| = N and a single cycle: $D_{max} = N - 2$.

Theorem 8 (L = p is never achievable). Exhaustive enumeration confirms that for m = 2 (verified to p = 19) and m = 3 (verified to p = 13), no cyclic word achieves L = p. This is consistent with the structural proof above.

5.3 Exhaustive Verification Results

Table 1: m = 2 scan

p	L_max	D_max	D=p−2?	L=p?	Optimal w
3	2	1	✓	Never	001
5	4	3	✓	Never	00001
7	6	5	✓	Never	0000001
10	9	8	✓	Never	0000000001
13	12	11	✓	Never	0000000000001
15	14	13	✓	Never	000000000000001
19	18	17	✓	Never	0000000000000000001

6. Infinite Memory Depth: The Even Shift

The even shift is defined on the bilateral state space $X = \{x \in \{0,1\}^{\mathbb{Z}} :$ the number of 0s between any two consecutive 1s is even$\}$. The dynamics F is the left shift $(Fx)_n = x_{n+1}$, which is bijective. The observation is $\pi(x) = x_0 \in \{0,1\}$.

The system has a hidden phase variable $S_t \in \{E, O\}$ tracking whether an even or odd number of 0s have occurred since the last 1. Under the Parry (maximal entropy) measure with eigenvalue $\varphi = (1+\sqrt{5})/2$:

- Transition probabilities: $P(E \to E) = 1/\varphi$ (emit 1), $P(E \to O) = 1/\varphi^2$ (emit 0), $P(O \to E) = 1$ (emit 0)
- Stationary state distribution (via Parry measure, $\mu_i = l_i r_i / \sum l_j r_j$): $\pi_E = \varphi^2/(\varphi^2+1) \approx 0.7236$, $\pi_O = 1/(\varphi^2+1) \approx 0.2764$

Theorem 9. For the even shift under the Parry measure, $I_k > 0$ for all $k \geq 0$. Hence $D = \infty$.

Proof sketch. The window 0^k cannot synchronize the hidden phase for any k. Given window 0^k, if $Y_t = 1$ then $S_t = E$ with certainty, but if $Y_t = 0$ then S_t remains ambiguous between E and O. This difference propagates to $P(Y_{t+k+1} = 1 \mid W, Y_t)$ regardless of k. Therefore $I_k > 0$ for every k. Note: while $I_k > 0$ holds for all k, it does not follow that a uniform lower bound $I_k \geq c > 0$ exists; numerical evidence suggests I_k decays (though remaining strictly positive). □

6.1 The Finite–Infinite Boundary

Theorem 10. If X is finite and F is a bijection, then $D < \infty$ under any stationary measure that assigns positive probability to all states (e.g., uniform over X, or uniform over a single periodic orbit). When X decomposes into multiple periodic orbits, the measure may be either a single-orbit uniform phase distribution or a mixture across orbits; in both cases $D \leq \max(\text{cycle lengths}) - 2 < \infty$. The true boundary for infinite depth is |X| finite vs |X| infinite, not reversibility vs irreversibility.

Condition	D = ∞ possible?	Example
Finite X + reversible	No	Register systems
Infinite X + reversible	Yes	Even shift
Finite X + irreversible	Yes	Certain HMMs

7. The Information Conservation Law

7.1 Telescoping Identity

Define $\Phi_k = H(Y_0 \mid Y_1, \ldots, Y_k)$. By the chain rule for conditional mutual information:

$$I_k = I(Y_{k+1} ; Y_0 \mid Y_1, \ldots, Y_k) = H(Y_0 \mid Y_1, \ldots, Y_k) - H(Y_0 \mid Y_1, \ldots, Y_{k+1}) = \Phi_k - \Phi_{k+1}$$

Summing over $k = 0, \ldots, p - 2$:

$$S = \sum_k I_k = \Phi_0 - \Phi_{p-1}$$

7.2 Boundary Values and Main Result

We have $\Phi_0 = H(Y_0)$ (the entropy of the observed symbol) and $\Phi_{p-1} = 0$ (since the full future window determines Y_0 under the maximal-depth condition $L = p - 1$). Therefore:

Main Theorem (Information Conservation). For any cyclic word w of period p:

$$\sum_{k=0}^{p-2} I_k = H(Y_0)$$

For binary alphabet ($m = 2$): $S \leq 1$ bit. For general alphabet: $S \leq \log_2 m$.

7.3 Interpretation

This identity reveals a fundamental structural constraint: **increasing memory depth merely redistributes a fixed information budget across scales.** No additional information is created by projection. The total penetration strength equals the entropy of the initial observation, regardless of how deep the memory extends.

7.4 Numerical Verification

Table 2: Conservation identity verification

Word w	$\sum I_k$	$H(Y_0)$	Match
001010010101 (p=12)	0.97987	0.97987	✓ exact
0010101001010101 (p=16)	0.98870	0.98870	✓ exact
000000000001 (p=12)	0.41382	0.41382	✓ exact

The identity holds with machine-precision exactness in all cases, not approximately.

8. Experimental Validation Protocol

Given a trajectory $Y_0, Y_1, \ldots, Y_T$ of the macroscopic process, construct sample triples $(Z_i, W_i, U_i) = (Y_i, Y_{i+1:i+k}, Y_{i+k+1})$ with $N = T - k - 1$ samples. The empirical estimate is:

$$\hat{I}_k = \hat{H}(U, W) + \hat{H}(W, Z) - \hat{H}(W) - \hat{H}(U, W, Z)$$

where $\hat{H}$ denotes plug-in entropy. The finite-sample bias is $O(m^k/N)$, requiring $N \gg 10 \cdot m^k$. We recommend permutation testing (shuffle Z_i 200–1000 times, accept $I_k > 0$ if the real value exceeds the permutation mean + 2σ). The internal consistency check $I_k \approx H(Y_t \mid Y_{t+1:t+k})$ provides a built-in verification for the residual window construction.

9. Discussion

We have established a complete characterization of projection-induced memory in reversible systems:

- **Arbitrary depth is constructible:** the residual window register achieves any target depth k with fully reversible microscopic dynamics.
- **Finite systems have bounded depth:** $D \leq N - 2$, and this bound is achieved by single-marker constructions.
- **Infinite depth requires infinite state space:** the even shift provides the canonical example.
- **Total strength is conserved:** $S = H(Y_0) \leq \log_2 m$, independent of depth.

The conservation law is perhaps the most surprising result. It means that memory depth and memory strength are fundamentally different quantities. A system can have memory extending over hundreds of steps, yet the total information content of that memory cannot exceed one bit (for binary observations). Depth expansion is redistribution, not amplification.

10. Conclusion

Memory is not a property of microscopic dynamics, but an emergent property of the interaction between observation scale, projection structure, and arithmetic residuals. A completely reversible system, under coarse-grained observation, can produce arbitrary-depth statistical memory. However, the total information strength of this memory is strictly conserved: it equals the entropy of the observed symbol and is bounded by the logarithm of the alphabet size. This establishes a precise sense in which projection creates the structure of memory but not its substance.

Appendix A: Verification Code (Python)

The following self-contained Python script verifies all core theorems. Run with: python full_verification.py

```
"""
Complete verification of the projection-induced memory theory.
Checks: reversibility, window recurrence, state equivalence,
I_k = H(Y_t|window) identity, numerical example, sync length,
D_max = p-2, L=p impossibility, periodicity, I_k upper bound.
"""
import math
from itertools import product
from collections import Counter
import random

def F_map(x, m, a, g_func):
    k = len(x) - 1
    new_last = (a * x[0] + g_func(x[1:])) % m
    return tuple(x[1:]) + (new_last,)

def F_inv(u, m, a, g_func):
    k = len(u) - 1
    a_inv = pow(a, -1, m)
    x0 = (a_inv * (u[k] - g_func(u[:k]))) % m
    return (x0,) + tuple(u[:k])

def compute_joint_entropy(samples):
    counts = Counter(samples)
    total = sum(counts.values())
    return -sum((c/total)*math.log2(c/total)
                for c in counts.values() if c > 0)

def compute_cond_MI(Z_list, W_list, U_list):
    UW = list(zip(U_list, W_list))
    ZW = list(zip(Z_list, W_list))
    UWZ = list(zip(U_list, W_list, Z_list))
    return (compute_joint_entropy(UW)
          + compute_joint_entropy(ZW)
          - compute_joint_entropy(W_list)
          - compute_joint_entropy(UWZ))

def sync_length(w):
    p = len(w)
    for L in range(1, p + 1):
        bucket = {}
        ok = True
        for i in range(p):
            b = tuple(w[(i+t)%p] for t in range(1, L+1))
            if b in bucket and bucket[b] != w[i]:
                ok = False; break
```

```
            bucket[b] = w[i]
        if ok: return L
    return p

# --- Run all checks ---
print("CHECK 1: F bijection")
ok = True
for m in [2, 3, 5]:
    for k in [1, 2]:
        a = next(c for c in range(1, m) if math.gcd(c, m)==1)
        gt = {args: random.randint(0,m-1)
              for args in product(range(m), repeat=k)}
        g = lambda args, gt=gt: gt[tuple(args)]
        X = list(product(range(m), repeat=k+1))
        imgs = set()
        for x in X:
            fx = F_map(x, m, a, g)
            imgs.add(fx)
            if F_inv(fx, m, a, g) != x: ok = False
        if len(imgs) != len(X): ok = False
print(f"  {'PASS' if ok else 'FAIL'}")

print("CHECK 2: Window recurrence")
ok = True
for m in [2, 3, 5]:
    for k in [1, 2]:
        a = next(c for c in range(1,m) if math.gcd(c,m)==1)
        gt = {args: random.randint(0,m-1)
              for args in product(range(m), repeat=k)}
        g = lambda args, gt=gt: gt[tuple(args)]
        for _ in range(20):
            s = tuple(random.randint(0,m-1) for _ in range(k+1))
            traj = [s]
            for t in range(k+5):
                s = F_map(s, m, a, g); traj.append(s)
            Y = [traj[t][0] for t in range(len(traj))]
            for t in range(len(Y)-k-1):
                lhs = Y[t+k+1]
                rhs = (a*Y[t] + g(tuple(Y[t+1:t+k+1]))) % m
                if lhs != rhs: ok = False
print(f"  {'PASS' if ok else 'FAIL'}")

print("CHECK 3: I_k identity (m=3,k=2)")
m, k, a = 3, 2, 2
g = lambda args: (args[0] + 2*args[1]) % 3
X_all = list(product(range(m), repeat=k+1))
for tk in range(4):
    Z, W, U = [], [], []
    for s0 in X_all:
        s = s0; traj = [s]
        for t in range(tk+3):
            s = F_map(s, m, a, g); traj.append(s)
```

```
        Y = [traj[t][0] for t in range(len(traj))]
        Z.append(Y[0])
        W.append(tuple(Y[1:tk+1]) if tk>0 else ())
        U.append(Y[tk+1])
    Iv = compute_cond_MI(Z, W, U)
    exp = math.log2(3) if tk==2 else 0
    status = "ok" if abs(Iv-exp)<1e-6 else "FAIL"
    print(f"  I_{tk} = {Iv:.6f} (expected {exp:.4f}) {status}")

print("CHECK 4: D_max = p-2")
ok = True
for m_val in [2, 3]:
    mx = 13 if m_val==2 else 8
    for p in range(3, mx+1):
        best_L = max(sync_length(list(w))
                     for w in product(range(m_val), repeat=p))
        if best_L != p-1: ok = False
print(f"  {'PASS' if ok else 'FAIL'}")

print("CHECK 5: Conservation S = H(Y0)")
def I_k_bits(w, k):
    p = len(w)
    counts = {}
    for t in range(p):
        key = (w[t], tuple(w[(t+i)%p] for i in range(1,k+1)),
               w[(t+k+1)%p])
        counts[key] = counts.get(key, 0) + 1
    c_w, c_zw, c_uw = {}, {}, {}
    for (z,ctx,u),v in counts.items():
        c_w[ctx] = c_w.get(ctx,0)+v
        c_zw[(z,ctx)] = c_zw.get((z,ctx),0)+v
        c_uw[(u,ctx)] = c_uw.get((u,ctx),0)+v
    I = sum(v/p*math.log2(v/p*c_w[ctx]/p
            / (c_zw[(z,ctx)]/p) / (c_uw[(u,ctx)]/p))
            for (z,ctx,u),v in counts.items())
    return max(I, 0)

for ws in ["001010010101", "000000000001", "0010101001010101"]:
    w = [int(c) for c in ws]
    p = len(w)
    S = sum(I_k_bits(w, k) for k in range(p-1))
    H0 = compute_joint_entropy([w[t] for t in range(p)])
    print(f"  w={ws}: S={S:.6f}, H(Y0)={H0:.6f}, "
          f"match={abs(S-H0)<1e-9}")

print("\nAll checks complete.")
```

Appendix B: Maximum Depth Scan Code (Python)

Brute-force scan for maximum synchronization length L(w) across all binary/ternary cyclic words.

```
"""
Scan maximum sync length L(w) for cyclic words.
"""
from itertools import product
import time

def sync_length(w):
    p = len(w)
    for L in range(1, p + 1):
        bucket = {}
        ok = True
        for i in range(p):
            b = tuple(w[(i+t)%p] for t in range(1, L+1))
            if b in bucket and bucket[b] != w[i]:
                ok = False; break
            bucket[b] = w[i]
        if ok: return L
    return p

def brute_force_max_L(m, p):
    best_L, best_w = 0, [0]*p
    for w in product(range(m), repeat=p):
        w = list(w)
        L = sync_length(w)
        if L > best_L:
            best_L, best_w = L, w
    return best_L, best_L-1, "".join(str(x) for x in best_w)

if __name__ == "__main__":
    for label, m, max_p in [("m=2", 2, 19), ("m=3", 3, 13)]:
        print(f"\n{'='*50}\n{label}\n{'='*50}")
        print(f"{'p':>4} {'L':>5} {'D':>5} {'D=p-2':>6} word")
        for p in range(2, max_p+1):
            t0 = time.time()
            L, D, w = brute_force_max_L(m, p)
            t = time.time()-t0
            eq = "YES" if D==p-2 else ""
            print(f"{p:>4} {L:>5} {D:>5} {eq:>6} {w} ({t:.1f}s)")
            if t > 30:
                print("  (stopping)"); break
```