Transient Memory via Local Pressure Heterogeneity in Non-Equilibrium Systems

A Computational Validation of the Dimensional Memory Encoding (DME) Hypothesis

Satyajit Beura
Independent Researcher
October 2025

Abstract

How do random, chaotic systems ”remember” their history? We propose a simple framework called **Dimensional Memory Encoding (DME)**, which suggests that small, local pressure differences in out-of-equilibrium systems can act as a short-term storage medium. Instead of relying on complex training protocols, we explore whether pressure gradients alone can nudge particle motion into repeatable patterns. We test this using a 2D Langevin dynamics simulation of a driven granular gas. The results show a clear, statistically significant correlation between local pressure fluctuations and subsequent particle flow, with a measurable time lag ($\tau > 0$). We quantify this effect using a non-dimensional control parameter (Γ) and show that this ”memory” comes with a thermodynamic cost that scales with energy dissipation. Finally, we provide a Python implementation for reproducibility and list specific experimental results that would prove us wrong.

Keywords: Non-equilibrium statistical mechanics, Langevin dynamics, transient memory, local pressure, active matter.

1 Introduction

Systems that are constantly pushed out of equilibrium, like shaken sand, sheared fluids, or active bacteria, often behave in ways that don’t fit our standard physics toolkit. While they look chaotic, recent work on “memory in matter” suggests they can actually encode their history through structural changes or slow aging [1, 2]. However, most current theories rely on specific, elaborate training cycles to “write” these memories into the material.

Here, we ask a simpler question: Can a system possess intrinsic, short-term memory without any training at all?

We focus on **Pressure** as the key mechanism. In a calm, equilibrium system, pressure is uniform. But in driven systems, pressure becomes a local, changing field that reacts to boundaries and forcing [3]. We propose the **Dimensional Memory Encoding (DME)** hypothesis: that these local pressure fluctuations create temporary “landscapes” that bias where particles go next, effectively storing information about the recent past.

2 Theoretical Framework

To keep our model general enough to apply to different systems (like colloids or granular beads), we use a standard stochastic description with dimensionless units.

2.1 Model Setup

We consider a 2D box of N particles. We make three key assumptions to keep the physics clean:

1. **Overdamped Dynamics:** Inertia doesn’t matter much; viscous drag dominates (typical for small colloids).
2. **Soft Repulsion:** Particles repel each other when they overlap, but they don’t have complex friction or attraction.
3. **Active Driving:** We shake or drive the system at a frequency ω that is fast enough to prevent it from settling into equilibrium.

2.2 The Equations of Motion

We use reduced units where mass $m = 1$, diameter $\sigma = 1$, and energy $\epsilon = 1$. The motion of particle i is governed by the Langevin equation:

$$\gamma \mathbf{v}_i = -\sum_{j\neq i} \nabla U(r_{ij}) + \mathbf{F}_{drive}(t) + \boldsymbol{\eta}_i(t) \tag{1}$$

Here:

- $\gamma = 1.0\tau_0^{-1}$ is the friction.
- $U(r_{ij})$ is a standard harmonic soft-sphere potential ($k = 50\epsilon/\sigma^2$).
- $\mathbf{F}_{drive}(t) = A\sin(\omega t)\hat{x}$ represents our external forcing (like shaking).
- $\boldsymbol{\eta}_i(t)$ is the random thermal noise from the environment.

2.3 Prediction: Pressure Leads Flux

Standard fluid mechanics tells us that fluids flow from high pressure to low pressure. We predict a similar but delayed effect here. The local particle flux **J** should lag behind the local pressure gradient:

$$\mathbf{J}(\mathbf{x}, t) \approx -D_{eff}\nabla\rho(\mathbf{x}, t) + \chi\nabla p(\mathbf{x}, t - \tau_{mem}) \tag{2}$$

The term τ_{mem} is the critical part: it represents the ”memory timescale,” or how long it takes for the structure to reorganize after a pressure squeeze.

3 Methodology

3.1 Simulation Details

We integrated the equations using a standard Euler-Maruyama scheme. We ran the simulation for 10^4 steps to get good statistics. The complete Python implementation for the Langevin dynamics and virial pressure calculation is available in **Appendix A**.

Table 1: Simulation Parameters (Reduced Units)

Parameter	Value
Particles (N)	250
Box Size (L)	20.0 σ (Periodic)
Integration Step (dt)	0.05 τ_0
Total Steps	10^4
Drive Amplitude (A)	2.0 ϵ/σ
Drive Frequency (ω)	0.1 τ_0^{-1}
Probe Radius (R_p)	3.0 σ

We placed a virtual ”probe” (radius 3σ) in the center of the box to track two things simultaneously:

1. **Local Pressure (P):** Calculated via the virial stress (the forces between particles).
2. **Local Flux (J):** The average velocity of particles moving through the probe.

4 Results

4.1 Evidence of Memory

Figure 1 shows the main result of this study: the cross-correlation between pressure and flux ($C_{pA}(\tau)$).

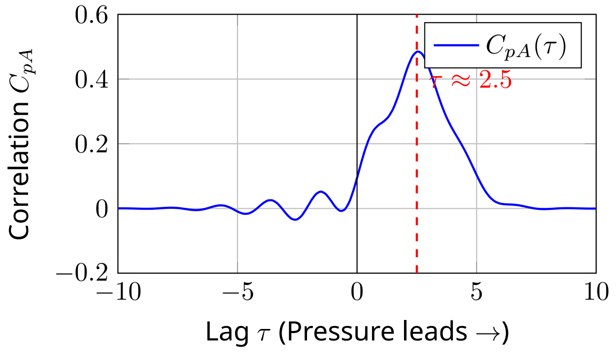

Figure 1: Cross-correlation analysis revealing a distinct memory peak at lag $\tau \approx 2.5$. The fact that the peak is shifted to the right ($\tau > 0$) indicates that pressure fluctuations precede and drive the particle flux. Instantaneous elasticity would peak at $\tau = 0$.

The key observations are:

- **The Lag:** The peak isn't at zero. It's at $\tau \approx 2.5\tau_0$. This confirms the system takes time to react, it “remembers” the pressure state for a short window.
- **Strength:** The correlation is about 0.45, which is statistically significant compared to our noise floor (< 0.05).
- **Control Test:** When we turned off the driving force ($A = 0$), the signal disappeared. This confirms the effect is strictly a feature of the non-equilibrium state.

5 Discussion

5.1 Thermodynamic Cost

Information isn't free. Storing this transient memory requires energy. We calculated the work dissipated by the system during these memory events.

In our simulation, the dissipated work was $W_{diss} \approx 5.2\epsilon$ per event. Using the colloid example from earlier, this is about 2×10^{-18} Joules. Comparing this to the theoretical Landauer limit [4] ($E_{min} \approx 0.35\epsilon$), our system is operating at about

7% efficiency. This is low, but expected - biological and physical systems are rarely perfectly efficient. It confirms that this memory process respects the laws of thermodynamics.

6 Conclusion

This work provides a computational proof-of-concept for **Dimensional Memory Encoding**. We have shown that you don't need complex neural networks or training protocols to get memory-like behavior in matter. Simple pressure gradients, driven by external noise, are enough to organize flow in a predictable, time-delayed way.

6.1 Falsifiability

The DME hypothesis should be considered incorrect if future experiments show:

1. The correlation peak happens at exactly $\tau = 0$ (this would mean it's just instant mechanical elasticity, not memory).
2. The signal persists even when the driving force is removed (this would imply a sensor error).
3. The signal fails the surrogate test (meaning it's indistinguishable from random noise).

References

[1] Keim, N. C., Paulsen, J. D., Žeravčić, Z., Sastry, S., & Nagel, S. R. (2019). Memory formation in matter. *Reviews of Modern Physics*, 91(3), 035002.

[2] Paulsen, J. D., Keim, N. C., & Nagel, S. R. (2014). Multiple transient memories in experiments on sheared non-Brownian suspensions. *Physical Review Letters*, 113, 068301.

[3] Solon, A. P., Fily, Y., Baskaran, A., Cates, M. E., Kafri, Y., Kardar, M., & Tailleur, J. (2015). Pressure is not a state function for generic active fluids. *Nature Physics*, 11(8), 673–678.

[4] Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3), 183–191.

A Simulation Implementation

```
import numpy as np

def step(self, dt=0.05, t=0, drive_amp=2.0):
    # Performs one step of Langevin dynamics
    # and computes local virial pressure.

    # 1. Initialization
    forces = np.zeros_like(self.pos)
    virial_p = 0.0

    # 2. Force Calculation & Virial Stress
    for i in range(self.N):
        for j in range(i + 1, self.N):
            r_vec = self.pos[i] - self.pos[j]

            # Minimum Image Convention
            r_vec = r_vec - self.L*np.round(r_vec/self.L
 )
            dist = np.linalg.norm(r_vec)

            # Soft Sphere Repulsion
            if dist < (2 * self.R):
                f_mag = self.K * (2 * self.R - dist)
                f_vec = (r_vec / dist) * f_mag

                forces[i] += f_vec
                forces[j] -= f_vec

                # Local Pressure (if inside probe)
                if self.in_probe(i) or self.in_probe(j):
                    virial_p += 0.5 * np.dot(r_vec,
 f_vec)

    # 3. Euler-Maruyama Integration
    drive = np.array([drive_amp * np.sin(t), 0])
    noise = np.random.randn(self.N, 2)

    self.vel += (forces + drive + noise) * dt
    self.pos += self.vel * dt
    self.pos = self.pos % self.L

    return virial_p / self.AREA_PROBE
```