Python Code

```
"""
Gray-Scott Habituation: Frequency Sweep
=======================================
The observer's protocol choice determines whether "habituation" appears.

We tap the same Gray-Scott system at different frequencies and show:
- Too slow: full relaxation, no habituation
- Sweet spot: incomplete relaxation, declining response = "habituation"
- Too fast: system can't respond, constant displacement

The Stentor researchers chose 45s because it matches the cell's
recovery time. We show the same frequency-dependent window in a PDE.

Author: D. Neale / Goleudy.ai
Date: March 2026
"""

import numpy as np

# ---- Gray-Scott engine ----
def laplacian(f):
    return np.roll(f,1,0)+np.roll(f,-1,0)+np.roll(f,1,1)+np.roll(f,-1,1)-4*f

def gs_step(U, V, F, k):
    Lu = laplacian(U); Lv = laplacian(V); uvv = U*V*V
    return (np.clip(U + 0.16*Lu - uvv + F*(1-U), 0, 1),
            np.clip(V + 0.08*Lv + uvv - (F+k)*V, 0, 1))

def grow_pattern(F, k, gs=64, steps=3000, seed=42):
    rng = np.random.RandomState(seed)
    U = np.ones((gs,gs)); V = np.zeros((gs,gs))
    sz = gs//10; r = gs//2
    U[r-sz:r+sz,r-sz:r+sz] = 0.5+0.02*rng.randn(2*sz,2*sz)
    V[r-sz:r+sz,r-sz:r+sz] = 0.25+0.02*rng.randn(2*sz,2*sz)
    U = np.clip(U,0,1); V = np.clip(V,0,1)
    for _ in range(steps):
        U, V = gs_step(U, V, F, k)
    return U, V

def apply_tap(U, V, centre, radius=3, strength=0.15):
    U_t, V_t = U.copy(), V.copy()
    r, c = centre; gs = U.shape[0]
    for dr in range(-radius, radius+1):
        for dc in range(-radius, radius+1):
            if dr*dr+dc*dc <= radius*radius:
                rr, cc = (r+dr)%gs, (c+dc)%gs
                U_t[rr,cc] = max(0, U_t[rr,cc]-strength)
                V_t[rr,cc] = min(1, V_t[rr,cc]+strength)
    return U_t, V_t

def measure_response(U0, V0, U1, V1):
    e0 = np.array([np.mean(U0), np.mean(V0), np.var(V0)])
    e1 = np.array([np.mean(U1), np.mean(V1), np.var(V1)])
    return np.sqrt(np.mean((e1-e0)**2))

# ---- Frequency sweep ----
def run_at_frequency(U_base, V_base, F, k, tap_centre,
                     interval, n_taps=12, relax=50,
                     strength=0.15, radius=3):
    """Run n_taps at given interval, return normalised responses."""
```

```
    U, V = U_base.copy(), V_base.copy()
    responses = []

    for tap in range(n_taps):
        U_pre, V_pre = U.copy(), V.copy()
        U, V = apply_tap(U, V, tap_centre, radius, strength)

        # Relax
        actual_relax = min(relax, interval)
        for _ in range(actual_relax):
            U, V = gs_step(U, V, F, k)

        response = measure_response(U_pre, V_pre, U, V)
        responses.append(response)

        # Remaining interval
        remaining = interval - actual_relax
        for _ in range(max(0, remaining)):
            U, V = gs_step(U, V, F, k)

    responses = np.array(responses)
    if responses[0] > 1e-10:
        return responses / responses[0]
    return responses

def main():
    F, k = 0.035, 0.065

    print("=" * 70)
    print("  Gray-Scott Habituation: Frequency Sweep")
    print("  Same system. Different observer protocol. Different inference.")
    print("=" * 70)

    print("\n  Growing stable spot pattern...")
    U_base, V_base = grow_pattern(F, k, gs=64, steps=3000)

    # Find tap location
    V_threshold = np.mean(V_base) + np.std(V_base)
    active = np.argwhere(V_base > V_threshold)
    tap_centre = tuple(active[len(active)//2]) if len(active) > 0 else (32, 32)
    print(f"  Tap location: {tap_centre}")

    # Frequency sweep
    intervals = [30, 50, 80, 120, 200, 400, 800]
    n_taps = 12

    print(f"\n  Tapping {n_taps} times at each frequency...")
    print(f"  Strength=0.15, radius=3, relax=50 steps")

    results = {}

    for interval in intervals:
        responses = run_at_frequency(
            U_base, V_base, F, k, tap_centre,
            interval=interval, n_taps=n_taps, relax=50
        )
        results[interval] = responses

        # Characterise: does the observer see habituation?
        first_3 = np.mean(responses[:3])
        last_3 = np.mean(responses[-3:])
        mid_3 = np.mean(responses[4:7])

        if mid_3 < first_3 * 0.7:
```

```
            if last_3 > mid_3 * 1.2:
                pattern = "HABITUATION + RECOVERY"
            else:
                pattern = "HABITUATION (sustained)"
        elif last_3 > first_3 * 1.1:
            pattern = "SENSITISATION"
        else:
            pattern = "NO CHANGE"

        print(f"\n  Interval = {interval:4d} steps:")
        print(f"    First 3 avg: {first_3:.4f}   Mid 3 avg: {mid_3:.4f}   Last 3 avg: {last_3:.4f}")
        print(f"    Pattern: {pattern}")
        print(f"    Responses: {' '.join(f'{r:.3f}' for r in responses)}")

    # ============================================================
    # THE OBSERVER'S FREQUENCY-DEPENDENT NARRATIVE
    # ============================================================

    print("\n" + "=" * 70)
    print("  THE OBSERVER'S FREQUENCY-DEPENDENT NARRATIVE")
    print("=" * 70)

    print("""
The same physical system. The same perturbation. The same emissions.
The only thing that changed was the observer's choice of tap frequency.

At some frequencies, the observer sees declining response and infers
"this system habituated — it learned to ignore the stimulus."

At other frequencies, the observer sees constant response and infers
"this system shows no learning."

At still other frequencies, the observer sees increasing response and
infers "this system sensitised — it became MORE responsive."

The system didn't change. The observer's protocol did. The cognitive
attribution — learning, habituation, sensitisation, no learning — is
determined by the observer's experimental design, not by the system.
    """)

    # Summary table
    print("  SUMMARY TABLE (for paper)")
    print(f"  {'Interval':>8s}  {'Response decline':>16s}  {'Observer infers':>20s}")
    print(f"  {'-'*50}")

    for interval in intervals:
        r = results[interval]
        first = np.mean(r[:3])
        minimum = np.min(r[2:8])
        last = np.mean(r[-3:])

        decline = (first - minimum) / first * 100 if first > 0 else 0

        if decline > 20:
            if last > minimum * 1.2:
                inference = "habituation + recovery"
            else:
                inference = "habituation"
        elif last > first * 1.05:
            inference = "sensitisation"
        else:
            inference = "no learning"

        print(f"  {interval:8d}  {decline:14.1f}%  {inference:>20s}")
```

```
    # Connection to Stentor
    print(f"""
  ----------------------------------------------------------------
  CONNECTION TO STENTOR:

  Eckert et al. (2024) tapped Stentor every 45 seconds — the time
  it takes for the cell to re-extend after contracting. This is the
  cell's natural recovery timescale.

  If they had tapped every 5 seconds: no time to respond → no decline
  If they had tapped every 5 minutes: full recovery → no decline
  They found habituation at 45 seconds because that frequency falls
  within the system's relaxation window.

  Our Gray-Scott system shows the same frequency window. The observer
  who taps within the relaxation window sees "habituation." The
  observer who taps outside it sees "no learning." The system is the
  same. The emissions depend on the protocol. The cognitive
  attribution depends on the emissions.

  The question "does this system habituate?" is incomplete without
  specifying "at what frequency is the observer probing?" — which
  means the answer is partly about the observer, not just about
  the system.
  ----------------------------------------------------------------
    """)


if __name__ == '__main__':
    main()

```