Individual-level behavioral phenotyping in *Drosophila* larvae using simulation-based inference: Supplementary Material

Gil Raitses Mirna Mihovilovic Skanata

Supplementary Methods

Simulation Parameters

[Details of simulation parameters will be added here]

Kernel Fitting Details

[Detailed kernel fitting procedures and optimization parameters]

Clustering Algorithm Details

[Detailed clustering parameters and validation metrics]

Supplementary Results

Individual Track Kernel Fits

[Summary statistics of individual kernel fits]

Cross-Validation Results

[Detailed cross-validation results tables]

Clustering Stability Metrics

[Detailed clustering stability analysis]

Supplementary Figures

[Supplementary figures will be added here]

Computational Field Notes

This section documents implementation decisions made during the analysis pipeline development, with emphasis on preserving statistical rigor.

GPU Vectorization for Power Analysis

0.0.0.1 Initial serial implementation. The power analysis was initially implemented in sequential Python (NumPy/SciPy). After 9+ hours of CPU execution, only 2 of 8 event-count conditions had completed. Projected total runtime exceeded 70 hours, making this approach impractical. The serial script was terminated and reimplemented for GPU execution. This section documents the reimplementation to confirm that statistical rigor was preserved.

0.0.0.2 Computational requirements. The simulation-based power analysis (main text, Methods: Power Analysis) requires fitting 100 population tracks $\times$ 100 fast-responder tracks $\times$ 8 event counts $\times$ 50 bootstrap replicates = 800,000 model fits. Sequential CPU execution was estimated at $>$70 hours.

To accelerate computation, the pipeline was migrated to GPU using JAX with full vectorization. This implementation choice maintains 100% statistical equivalence to sequential execution:

- **Simulation fidelity**: Each track is simulated from the same gamma-difference kernel model with reproducible random seeds via JAX's PRNG key splitting.
- **Optimization equivalence**: MLE optimization uses identical objective functions (log-likelihood with integral term) and convergence criteria.
- **Bootstrap procedure**: Parametric bootstrap samples are generated identically—simulate from fitted parameters, refit, collect parameter estimates.
- **CI calculation**: Confidence intervals computed as 2.5th–97.5th percentiles of bootstrap distributions.
- **Error rate definitions**: Type I error = proportion of population tracks whose CI excludes true τ_1; Power = proportion of fast-responder tracks whose CI excludes population mean.

The only difference is *execution strategy*: `vmap(process_track)(keys)` processes all 100 tracks in parallel on GPU, whereas a Python `for` loop processes them sequentially on CPU. Mathematically, these are identical—`vmap` is a parallel map operator with no side effects that could alter results.

Random number handling ensures reproducibility:

```
key = PRNGKey(42)
keys = split(key, 100)
# Sequential: for i in range(100): simulate(keys[i])
# Vectorized: vmap(simulate)(keys)  # Same keys, same results
```

Final runtime with GPU vectorization: $\sim$30 minutes (Tesla T4) vs $>$70 hours (CPU sequential), a $>$140$\times$ speedup with zero impact on statistical validity.

Event Definition Verification

During validation pipeline development, a critical inconsistency was identified: some scripts used `klein_run_table/time0` (run start times) while others used `is_reorientation_start` (reorientation onset times). These are distinct events:

- **Run start**: The larva begins a forward locomotion bout.

- **Reorientation onset**: The larva transitions from run to turn (the event modeled by the kernel).

All pipelines were verified to use `is_reorientation_start` consistently, matching the event definition in the original study.

Multi-Start Optimization

The 6-parameter gamma-difference kernel produces a non-convex likelihood surface. Initial implementations used single-start optimization, which converged to local minima in ∼15–20% of tracks, producing qualitatively incorrect kernel shapes (e.g., inverted polarity, implausible time constants).

The corrected implementation uses grid search initialization:

- $\tau_1 \in \{0.3, 0.6, 0.9\}$ s
- $\tau_2 \in \{1.0, 2.0, 3.0\}$ s
- $A/B \in \{1.0, 2.0\}$

yielding 18 initial points. The solution with highest final log-likelihood was retained.

Structural Identifiability Analysis

During power analysis debugging, a fundamental structural identifiability issue was discovered that explains the difficulty of individual-level kernel fitting.

0.0.0.3 The problem. The gamma-difference kernel with population parameters $A = 1.5$ and $B = 12.0$ produces a predominantly *inhibitory* response: $K(t) < 0$ for $t > 0.2$ s during LED-ON. This means:

1. Events are *suppressed* during LED-ON relative to LED-OFF.
2. Only ∼20% of events occur during LED-ON, despite LED-ON comprising 33% of the stimulus cycle.
3. A typical track has only ∼2 events in the LED-ON window where τ_1 information is concentrated.

0.0.0.4 Diagnostic evidence. Likelihood surface analysis revealed that the log-likelihood is nearly flat across a wide range of τ_1 values. For a representative track with 11 total events (2 in LED-ON):

```
True tau1 = 0.63s
LL at tau1=0.63: -53.98
LL at tau1=1.50: -53.83 (HIGHER - MLE converges here)
Fitted tau1: 1.87s (3x too high)
```

The MLE finds a spuriously high τ_1 because the likelihood difference between true and incorrect values is smaller than stochastic variation.

0.0.0.5 Why longer tracks do not help. The ratio of informative (LED-ON) to uninformative (LED-OFF) events is determined by the stimulus protocol and kernel shape, not by track duration. Extending recordings from 20 to 80 minutes would increase total events proportionally, but the

~20% informative fraction would remain constant. The Fisher information for τ_1 grows only with informative events in the LED-ON window.

0.0.0.6 Implications. This finding validates the manuscript's conclusion that individual τ_1 estimation is not feasible under the current experimental design. The problem is structural (kernel parameterization + stimulus protocol) rather than merely data sparsity. Resolution would require either:

- Modified experimental design (higher duty cycle, pulse trains), or
- Simplified model (only τ_1 varies by individual; other parameters fixed at population values).

Code Availability

All analysis code is available at https://github.com/GilRaitses/indysim in the `scripts/2025-12-16/phenotyping_f` directory.