Interface Dynamics Simulation (InDy-sim)

Stimulus-Driven Behavioral Modeling of Drosophila Larvae

Gil Raitses

2025-11-06

Abstract

Interface Dynamics Simulation (InDy-sim) investigates behavior as the **interface between perception and adaptation**, combining **agent-based simulation** with **dynamic transition modeling**. This project develops a **stimulus-locked event-hazard model** for **Drosophila larval behavior**, treating **behavioral transitions** as both signal and medium that reveal how **sensory representation** translates into **neural activity** and observable behavior. The model estimates time-varying hazard rates for behavioral events conditioned on **stimulus intensity**, **temporal history**, and **contextual features** including speed and orientation. Using a **design of experiments** framework with **stimulus intensity**, **pulse duration**, and **inter-pulse interval** as factors, the analysis generates **simulated trajectories** that produce **key performance indicators** including mean turn rate, latency to first turn, stop fraction, and spatial distribution metrics. **Confidence intervals** for behavioral metrics provide statistical precision comparable to **Arena simulation replications**, which supports validation of model predictions against empirical larval trajectory data.

1 Introduction

Interface Dynamics Simulation (InDy-sim) investigates behavior as the **interface between perception and adaptation**, treating behavioral transitions as both signal and medium. This framework bridges **computational neuroscience** with **media theory**, modeling how **sensory representation** translates into **neural activity** and ultimately manifests as observable behavior. The behavioral interface reveals how organisms process time-varying stimuli through **hierarchical dynamics** spanning sensory transduction, neural integration, and motor output.

This term project develops a **stimulus-response modeling framework** for **Drosophila larval behavior** using **event-hazard methods** from **survival analysis** and **generalized linear models**. The project uses existing trajectory data from mechanosensation experiments to estimate time-varying hazard rates for behavioral events including turns, stops, and reversals conditioned on **LED stimulus intensity**, **temporal history**, and **contextual features**. The model simulates trajectories under different **experimental conditions** specified through a **design of experiments** framework, which produces behavioral metrics comparable to empirical observations.

The project connects simulation modeling methods from **ECS630** with biological data analysis. **Discrete-event simulation principles** apply to **behavioral systems** where events occur stochastically as functions of external stimuli and internal state. Results use **Arena-style summary statistics** including **across-replications summaries**, **confidence intervals**, and **performance metrics** analogous to manufacturing system analysis.

2 Problem Statement

Research Question

How can **larval behavioral responses** to **time-varying LED stimuli** be modeled using **event-hazard methods** to predict **turn initiation rates**, **behavioral state transitions**, and **trajectory-level statistics**?

Dataset Characteristics

The mechanosensation dataset contains trajectory data with individual larval tracks including position coordinates, speeds, orientations, and behavioral events such as runs, reorientations, and head swings. Stimulus data consists of LED intensity values synchronized to video frames, with stimulus onset times, pulse durations, and inter-pulse intervals. Experimental conditions span multiple experiments with varying stimulus intensities ranging from 0 to 100% LED power, pulse frequencies, and temporal patterns. Sample size includes 40+ larvae per condition with multiple replications per experimental condition.

Modeling Challenge

Behavioral events occur stochastically with rates that depend on stimulus features including current intensity, recent history showing adaptation and habituation, and pulse timing. Rates also depend on contextual features including current speed and orientation relative to stimulus source. Individual heterogeneity contributes through larva-to-larva variation in baseline activity and sensitivity. Traditional approaches use simple rate averages or linear regressions. This project develops a **structured hazard model** that captures temporal dependencies and supports trajectory simulation.

3 Proposed Methodology

Core Model: Stimulus-Locked Event-Hazard GLM

For each behavioral event type $E \in \{\text{turn}, \text{stop}, \text{reverse}\}$, define the time-varying hazard rate (see Equation 1 in Appendix) where $\beta_{0,E}$ represents the baseline log-hazard for event type E, $s(t)$ is the stimulus feature vector including intensity, on/off state, and recent history, κ is the temporal kernel using basis expansion to capture latency and adaptation, $[s \star \kappa](t)$ is the convolution of stimulus with kernel producing stimulus history features, $\mathbf{x}(t)$ represents contextual features including speed and orientation, and $_E$ are the feature coefficients.

Temporal Kernel Design

The kernel $\kappa(\tau)$ captures latency effects with peak response at delay $\tau_0 \approx 0.5-2$ seconds, adaptation showing decay over longer delays when $\tau > 5$ seconds, and anticipation through pre-stimulus effects if present.

Using raised cosine basis functions (see Equation 2 in Appendix) with knots τ_j spanning $[-2, 20]$ seconds relative to stimulus onset.

Event Likelihood and Estimation

For larva i with observed events at times $\{t_{i,k}\}$, the log-likelihood follows Equation 3 in Appendix. Estimation via **penalized maximum likelihood** with L_2 regularization (see Equation 4 in Appendix).

Model Validation

Model validation uses three methods. The time-rescaled KS test transforms observed event times $t_k \rightarrow \int_0^{t_k} \lambda_E(t)dt$, which under the correct model should form Poisson process increments. Predictive log-likelihood evaluates holdout larvae for each experimental condition. Peri-stimulus time histograms compare model-predicted versus observed event rates around stimulus onsets.

Trajectory Simulation

Given a fitted hazard model, the simulation follows five steps. Initialization samples starting position, orientation, and speed from empirical distributions. Event generation samples the next event time t^* from $\lambda_E(t)$ using a thinning algorithm. Event execution handles turns by sampling new orientation, stops by pausing, and reversals by flipping direction. State update integrates position using forward Euler integration with current speed and orientation. The process repeats until reaching maximum simulation time.

Speed Dynamics

Run speeds follow observed empirical distributions (typically log-normal). During runs, integrate using Equation 5 in Appendix where $v(t)$ is current speed and $\theta(t)$ is current heading.

Design of Experiments

Factors and Levels

Factor	Levels	Description	Units
Stimulus Intensity	3 (PWM 250, 500, 1000)	Threshold, moderate, and high stimulus levels	PWM percentage
Pulse Duration	5 (10s, 15s, 20s, 25s, 30s)	Short to extended stimulus presentations	Seconds
Inter-Pulse Interval	3 (5s, 10s, 20s)	Rapid, moderate, and spaced stimulus timing	Seconds

Response Variables (KPIs)

KPI	Description	Units
Turn Rate	Reorientations per minute	reorientations/min
Latency to First Turn	Time from stimulus onset to first turn	seconds
Stop Fraction	Proportion of time spent stopped	dimensionless
Pause Rate	Pauses per minute	pauses/min
Path Tortuosity	Net displacement / path length	dimensionless
Spatial Dispersal	Mean distance from starting position	pixels
Mean Spine Curve Energy	Average curvature energy along body axis	dimensionless

Experimental Design

Full factorial design: $3 \times 5 \times 3 = 45$ conditions. For each condition, the simulation runs 30 replications where each replication represents one larva. The process records events and computes KPIs per replication, then generates **AcrossReplicationsSummary.csv** with means and confidence intervals.

Run Length Determination

The replication length follows Lab04 CI methodology. Target precision uses 10% relative half-width for mean turn rate, which means when the mean equals 5 turns per minute the confidence interval width stays below 0.5 turns per minute. A pilot study runs 10 replications to estimate variance $\hat{\sigma}^2$. Required replications follow Equation 6 in Appendix where E represents the desired half-width. When $n > 30$ the simulation uses that calculated value, while values $n \leq 30$ default to 30 replications as the minimum standard.

4 Analysis Plan

Data Preprocessing

Input Data Sources

The analysis uses trajectory data from MAGAT experiments containing position coordinates, speeds, orientations, and behavioral events synchronized with LED stimulus timing. Data includes multiple experimental conditions with varying stimulus intensities, pulse durations, and inter-pulse intervals, providing sufficient sample size for model fitting and validation.

Feature Extraction

The analysis extracts behavioral features from trajectory data including position coordinates, speed, heading angles, and event timing. Features are computed at 50ms time bins and aligned with stimulus timing to capture temporal relationships between stimulus presentation and behavioral responses. The feature set includes stimulus history features derived from temporal kernel convolution, contextual features such as current speed and orientation, and event indicators marking turn starts, stop starts, and reversals.

Model Fitting Pipeline

The model fitting process identifies behavioral events from trajectory data, constructs feature matrices combining stimulus history and contextual variables, and fits GLM models using penalized maximum likelihood estimation. Cross-validation assesses model generalization using leave-one-larva-out validation. Model validation evaluates temporal fidelity through time-rescaled KS tests and compares predicted versus observed event rates using peri-stimulus time histograms.

Simulation and DOE Execution

Simulation Engine

The trajectory simulator generates larval paths by sampling event times from fitted hazard models and integrating position using empirical speed distributions. The simulator handles behavioral state transitions including turns, stops, and reversals, producing complete trajectories under specified stimulus schedules.

DOE Execution

The full factorial design executes 45 experimental conditions with 30 replications per condition. Each replication simulates a complete larval trajectory under the condition's stimulus parameters, computes KPIs, and aggregates results into summary statistics with confidence intervals following Arena-style output format.

Statistical Analysis

Performance Metrics Calculation

For each KPI and condition, the analysis computes sample means, standard deviations, 95% confidence intervals using t-distributions, and minimum/maximum values across replications. These statistics provide point estimates and uncertainty quantification for behavioral metrics under each experimental condition (see Equations 7–9 in Appendix).

Model Validation Metrics

Model validation uses four metrics. Holdout log-likelihood compares results to a null model with constant hazard. The KS test p-value evaluates whether time-rescaled event times follow a uniform distribution. PSTH correlation compares model-predicted versus observed peri-stimulus histograms. Trajectory-level statistics compare simulated versus empirical distributions of KPIs.

5 Expected Deliverables

Report Components

The report includes six components. Model specification covers mathematical formulation, kernel design, and estimation procedure. Data description presents dataset characteristics, preprocessing steps, and feature engineering. Model fitting results show fitted kernels, coefficients, and validation metrics. DOE results include AcrossReplicationsSummary tables, main effects, and interaction plots. Simulation validation compares simulated versus empirical KPIs. Confidence intervals explain CI construction methodology and run length justification.

Supporting Files

Supporting files include four categories. Model code consists of Python scripts for fitting (`fit_hazard_model.py`), simulation (`simulate_trajectories.py`), and DOE execution (`run_doe.py`). Data exports produce Arena-style CSV files including `AcrossReplicationsSummary.csv`, `ContinuousTimeStatsByRep.csv`, and `DiscreteTimeStatsByRep.csv`. The DOE table stores 45 conditions and factor levels in `doe_table.csv`. Configuration uses `config.json` for model parameters, simulation settings, and CI targets.

Format Requirements

Format requirements include four elements. The report renders to PDF with style matching Lab01 and Lab02 formatting using Avenir Next body text, Didot headings, and technical terminology. Figures include kernel plots, PSTHs, KPI comparisons, and interaction plots. Tables include the DOE table, AcrossReplicationsSummary, and model coefficients.

6 Timeline and Milestones

Week 1: November 6-13 (Data, Model Development, and Fitting)

Week 1 covers data preparation, model development, and initial model fitting. The week loads and inspects trajectory and stimulus data, implements the feature extraction pipeline, and computes empirical KPIs for baseline conditions. It implements temporal kernel basis functions, builds the GLM fitting pipeline, and fits a baseline model with constant hazard. The week fits stimulus-locked hazard models for turns, stops, and reversals, performs cross-validation and hyperparameter tuning, and validates models using KS tests and PSTH comparisons. Deliverables include a data exploration notebook, baseline statistics, fitted models, and a validation report.

Week 2: November 13-20 (Simulation, DOE, and Report)

Week 2 covers simulation development, DOE execution, and report completion. The week implements the trajectory simulator, tests on simple stimulus schedules, and validates that simulated KPIs match empirical values for known conditions. It runs the full factorial design with 45 conditions and 30 replications per condition, generates AcrossReplicationsSummary CSVs, computes confidence intervals, and analyzes main effects and interactions. The week writes the report with all sections, generates figures and tables, and performs final model validation against holdout data. Deliverables include a working simulator, complete DOE results and summary tables, and the final report PDF with supporting code and data.

7 Risk Assessment and Mitigation

Technical Risks

Temporal alignment between stimulus and behavior data could be inaccurate. Mitigation uses validated synchronization methods through MAGAT `addTonToff`, implements unit tests for alignment, and visually inspects PSTHs for timing errors.

The model could overfit to training data with a complex kernel. Mitigation uses cross-validation, applies L_2 regularization, limits kernel complexity by reducing basis functions, and compares AIC/BIC across models.

Individual heterogeneity could violate the exchangeability assumption. Mitigation includes larva-specific random effects when needed, stratifies analysis by experimental batch, and uses hierarchical models when time permits.

The model could fail to capture complex spatial dynamics. Mitigation validates that spatial distributions match empirical data and includes position-dependent features when needed.

Data Risks

Some experimental conditions might have insufficient data. Mitigation pools similar conditions, uses data augmentation through bootstrap resampling, and reports sample sizes per condition.

Trajectory segments could be missing or corrupted. Mitigation implements robust data validation, excludes problematic trajectories, and reports exclusion criteria in methods.

Timeline Risks

Model fitting could take longer than expected. Mitigation starts with a simplified model using fewer features, uses existing GLM libraries, and parallelizes over larvae when needed.

Simulation could be computationally expensive for 1,350 total runs across 45 conditions with 30 replications each. Mitigation optimizes simulation code, runs overnight batches, uses cloud computing when needed, and reduces replications if CI targets are still met.

8 Success Criteria

Model Performance Targets

Model performance targets include three criteria. Predictive accuracy requires holdout log-likelihood improvement of at least 20% over the null model. Temporal fidelity requires a KS test p-value greater than 0.05, which indicates the model is not rejected. Kernel interpretability requires the fitted kernel to show biologically plausible latency with peak response between 0.5 and 2 seconds.

Simulation Validation Targets

Simulation validation targets include three criteria. KPI accuracy requires simulated mean turn rates to fall within the 95% CI of empirical values for baseline conditions. Distribution match requires simulated KPI distributions to pass a two-sample KS test versus empirical distributions with p-value greater than 0.05. DOE consistency requires main effects to show expected signs such as higher intensity producing higher turn rate.

Deliverable Quality Targets

Deliverable quality targets include three criteria. Report completeness requires all sections present with clear figures and tables and reproducible methods. Code quality requires well-documented scripts, configuration files, and unit tests where feasible. Data format requires CSVs to match Arena output format and remain readable by standard tools.

9 Conclusion

This project applies **simulation modeling methods** from **ECS630** to **biological behavioral data**, developing a **stimulus-response model** that simulates larval trajectories under different experimental conditions. The **event-hazard framework** models stochastic behavioral events, while the **DOE methodology** explores stimulus parameter space systematically. Results use **Arena-style statistics** including **confidence intervals** and **across-replications summaries**, which show how simulation principles extend from manufacturing systems to biological processes.

10 Appendix

Equations

Core Model Equations

Equation 1: Hazard Rate Model

The time-varying hazard rate for behavioral event type E combines baseline log-hazard, stimulus history through temporal kernel convolution, and contextual features. This exponential link function ensures positive hazard rates while allowing linear combination of features in log-space.

$$\lambda_E(t) = \exp\left\{\beta_{0,E} + \phi_E^\top [s \star \kappa](t) + \mathbf{x}(t)^\top{}_E\right\} \tag{1}$$

Equation 2: Temporal Kernel Basis Expansion

The temporal kernel uses raised cosine basis functions centered at knots τ_j spanning the analysis window. Each basis function has local support $[\tau_j - \Delta\tau, \tau_j + \Delta\tau]$ and captures stimulus-response dynamics at different time delays relative to stimulus onset.

$$\kappa(\tau) = \sum_{j=1}^{J} w_j \cos^2\left(\frac{\pi(\tau - \tau_j)}{2\Delta\tau}\right) \mathbf{1}_{[\tau_j - \Delta\tau, \tau_j + \Delta\tau]}(\tau) \tag{2}$$

Equation 3: Event Likelihood

The log-likelihood sums log-hazard contributions from observed events across all larvae and subtracts the integrated hazard over each larva's observation period. This formulation follows standard point process likelihood theory for inhomogeneous Poisson processes.

$$\ell() = \sum_i \sum_k \log \lambda_E(t_{i,k}) - \int_0^{T_i} \lambda_E(t)dt \tag{3}$$

Equation 4: Penalized Maximum Likelihood Estimation

Parameter estimation maximizes the log-likelihood subject to L_2 regularization penalty. The regularization parameter λ_{reg} controls the trade-off between model fit and complexity, preventing overfitting to training data.

$$\hat{} = \arg\max\left\{\ell() - \lambda_{\text{reg}}\|\|_2^2\right\} \tag{4}$$

Equation 5: Position Integration

During run phases, position updates using forward Euler integration with current speed $v(t)$ and heading angle $\theta(t)$. The integration step size Δt balances numerical accuracy with computational efficiency.

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + v(t)\Delta t \cdot [\cos(\theta(t)), \sin(\theta(t))]^\top \tag{5}$$

Equation 6: Replication Count Determination

The required number of replications n depends on the desired confidence interval half-width E, estimated variance $\hat{\sigma}^2$, and significance level α. This formula ensures statistical precision targets are met while minimizing computational cost.

$$n \geq \left(\frac{z_{\alpha/2}\hat{\sigma}}{E} \right)^2 \tag{6}$$

Statistical Metrics

Equation 7: Sample Mean

The sample mean provides an unbiased estimate of the population mean for each KPI across replications within a condition. This statistic serves as the primary point estimate for comparison across experimental conditions.

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{7}$$

Equation 8: Sample Standard Deviation

The sample standard deviation quantifies variability across replications using Bessel's correction ($n - 1$ in the denominator) to provide an unbiased estimate of population variance. This measure informs confidence interval width and statistical power.

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2} \tag{8}$$

Equation 9: 95% Confidence Interval

The confidence interval uses the t-distribution with $n - 1$ degrees of freedom to account for uncertainty in variance estimation with finite samples. The interval provides a range of plausible values for the population mean at 95% confidence level.

$$\bar{X} \pm t_{0.025, n-1} \frac{s}{\sqrt{n}} \tag{9}$$

Code Blocks

Feature Extraction

Code ??: H5 Data Extraction Command

The extraction script processes raw H5 files from MAGAT experiments, extracting trajectory positions, LED stimulus data, and metadata. The script outputs event records, full trajectory data aligned with stimulus timing, and summary statistics in CSV and JSON formats for downstream analysis.

```
{#code-h5-extract}
python3 scripts/engineer_dataset_from_h5.py \
    --h5-dir /Users/gilraitses/mechanosensation/h5tests \
    --output-dir data/engineered \
    --experiment-id GMR61_202509051201
```

Code ??: Feature Engineering Pipeline

The feature engineering pipeline processes each larval trajectory by extracting position coordinates, computing derived features including speed and heading, detecting behavioral events such as turns, and aligning trajectory data with stimulus timing. The pipeline creates time bins and generates stimulus kernel features for model fitting.

```
{#code-feature-engineering}
# For each larva trajectory from H5:
for each track in h5_file['tracks']:
    # Extract positions (x, y) and compute derived features
    positions = track['positions']  # N×2 array
    speed = compute_speed(positions, frame_rate=20)
    heading = compute_heading(positions)
    heading_change = detect_turns(heading, threshold=30°)

    # Time-align with stimulus data
    stimulus_df = extract_stimulus_timing(h5_file['led_data'])
    aligned_df = align_trajectory_with_stimulus(trajectory_df, stimulus_df)

    # Create time bins (50ms)
    bin_width = 0.05
    binned = aggregate_to_bins(aligned_df, bin_width)

    # Create stimulus kernel features (in fit_hazard_model.py)
    for each bin:
        stimulus_history = extract_stimulus_window(time_window=[-2, 20])
        kernel_features = apply_temporal_kernel(stimulus_history, basis_functions)
        feature_vector = [kernel_features, speed, heading]
        event_occurred = (binned['is_turn'] == True)
```

Simulation Engine

Code ??: Trajectory Simulator Class

The simulator class implements discrete-event simulation of larval trajectories using fitted hazard models. The simulation initializes state variables, samples event times from hazard rates using a thinning algorithm, executes behavioral events, and integrates position using forward Euler method until reaching maximum simulation time.

```
{#code-simulator}
class LarvalTrajectorySimulator:
    def __init__(self, hazard_model, speed_distribution, arena_bounds):
        self.hazard_model = hazard_model
        self.speed_dist = speed_distribution
        self.arena = arena_bounds

    def simulate(self, stimulus_schedule, max_time, random_seed):
        # Initialize state
        position = sample_starting_position()
```

```
        orientation = sample_starting_orientation()
        speed = self.speed_dist.sample()

        events = []
        t = 0

        while t < max_time:
            # Compute current hazard
            lambda_t = self.hazard_model.predict(
                stimulus=stimulus_schedule(t),
                speed=speed,
                orientation=orientation
            )

            # Sample next event time (thinning algorithm)
            next_event_time = sample_from_hazard(lambda_t, current_time=t)

            # Execute event
            if next_event_time < max_time:
                event_type = sample_event_type(lambda_t)  # turn, stop, reverse
                events.append((next_event_time, event_type))

                # Update state based on event
                if event_type == 'turn':
                    orientation = sample_new_orientation(orientation)
                elif event_type == 'stop':
                    speed = 0
                elif event_type == 'reverse':
                    orientation = flip_orientation(orientation)

            # Update position
            dt = min(next_event_time - t, 0.1)  # 100ms integration step
            position += speed * dt * [cos(orientation), sin(orientation)]
            t = next_event_time

        return Trajectory(position_history, events)
```

DOE Execution

Code ??: Design of Experiments Execution Loop

The DOE execution script iterates through all 45 factorial conditions, running 30 replications per condition. For each replication, the script creates a stimulus schedule from condition parameters, simulates a trajectory, computes KPIs, and aggregates results into Arena-style CSV format with confidence intervals.

```
{#code-doe-execution}
results = []
for condition in doe_table:
    for replication in range(30):
        # Set stimulus schedule from condition
        stimulus = create_stimulus_schedule(
            intensity=condition.intensity,
            pulse_duration=condition.pulse_duration,
            inter_pulse_interval=condition.inter_pulse_interval
```

```
        )

        # Simulate trajectory
        traj = simulator.simulate(stimulus, max_time=300, random_seed=replication)

        # Compute KPIs
        kpis = compute_kpis(traj)
        results.append({
            'condition': condition.id,
            'replication': replication,
            **kpis
        })

# Export to Arena-style CSV
export_to_arena_format(results, 'AcrossReplicationsSummary.csv')
```

Data Sources

Primary H5 files contain complete experiments with tracks and LED data (tier1, 16 MB), full tier2 exports with contours (83 MB), and tier3 exports with FID data (701 KB). Backup CSV data provides trajectory and spatial analysis outputs. Stimulus data contains frame-level LED values. Experimental metadata consists of experiment IDs embedded in H5 metadata or trajectory CSVs.

Model Implementation References

Model implementation draws on survival analysis methods from Collett (2015) *Modelling Survival Data in Medical Research*, point process GLM approaches from Berman & Turner (1992) "Approximating point process likelihoods with GLIM", and temporal kernel methods from Pillow et al. (2008) "Spatio-temporal correlations and visual signalling".

Software Tools

Software tools include Python with **`scikit-learn`**, **`statsmodels`**, **`numpy`**, and **`pandas`** for modeling, R for statistical analysis, and MATLAB using the MAGAT API for data loading when needed.