1 NEAT-DNFs hyperparameter settings

The `neat-dnfs` algorithm exposes a set of hyperparameters that shape (i) the neural field simulation used to evaluate candidate solutions and (ii) the evolutionary dynamics used to search over DNF architectures. We group these parameters into four categories: (1) simulation environment configuration, (2) neural field and kernel parameter spaces, (3) evolutionary algorithm and speciation settings, and (4) mutation operator probability distributions.

Unless otherwise noted, the values below correspond to the current defaults defined in `constants.h`. Several parameters define *ranges* rather than fixed values: these ranges specify the admissible parameter space for random initialisation and step-wise mutation (clamped to bounds).

1.1 Temporal dynamics and numerical stability

The phenotype simulation advances in discrete time using Euler integration with time step Δt (`SimulationConstants::deltaT`). Neural field dynamics are governed by a time constant τ (`NeuralFieldConstants`), which controls how quickly activations respond to inputs. In general, stable Euler integration benefits from a separation of scales $\Delta t \ll \tau$. In the current implementation, τ is evolved within a bounded interval (Table 2), while Δt is fixed (Table 1). If oscillatory artefacts or numerical instability are observed, a robust mitigation is to (i) increase $\tau_{\min}$ and/or (ii) decrease Δt.

The parameter `SimulationConstants::maxSimulationSteps` is used as a simulation budget (number of integration steps) in evaluation routines (e.g., when iteratively stepping the phenotype while stimuli move across the field). It therefore directly trades off computational cost against convergence/stabilisation time.

1.2 Simulation environment configuration

Table 1: Simulation environment parameters (current defaults). These parameters define the discrete-time simulation used to evaluate each evolved phenotype. Spatial parameters set the discretisation of 1D neural fields. Noise and stimulus parameters standardise the stochastic perturbations and input signals presented during evaluation.

Parameter	Value	Role / effect
Temporal dynamics		
Time step Δt SimulationConstants::deltaT	1	Euler integration step passed to the phenotype simulator. Smaller values improve numerical stability but increase runtime.
Max. simulation steps SimulationConstants::maxSimulationSteps	500	Step budget used in evaluation loops (e.g., stimulus motion / settling). Larger values allow longer settling at higher compute cost.
Spatial configuration		
Field size x_{size} DimensionConstants::xSize	100	Number of discrete spatial nodes per 1D DNF. Controls resolution and compute per step.
Spatial step dx DimensionConstants::dx	1.0	Spatial discretization.
Noise model		
Noise amplitude q NoiseConstants::amplitude	0.010	Amplitude for Gaussian normal noise injected into fields. Increases robustness and can help break symmetry.
Stimulus (Gaussian) defaults		
Stimulus width σ_s GaussStimulusConstants::width	5.0	Spatial spread of Gaussian stimulus inputs.
Stimulus amplitude A_s GaussStimulusConstants::amplitude	20.0	Peak magnitude of stimulus input.
Stimulus circularity GaussStimulusConstants::circularity	true	Whether stimulus wraps around field boundaries.
Stimulus normalisation GaussStimulusConstants::normalization	true	Whether stimulus is normalised.

1.3 Neural field and kernel parameter spaces

Field genes encode neural fields and their *self-interaction* kernels. If `FieldGeneConstants::variableParameters` is enabled (currently `true`), new fields initialise their parameters uniformly at random within the allowed bounds, and subsequent mutations perform step-wise updates clamped to these bounds.

Kernel objects (both self-kernels and connection kernels) share the global flags `KernelConstants::circularity` and `KernelConstants::normalization`, which determine boundary handling and whether kernels are normalised.

Table 2: Neural field and kernel parameter spaces (current defaults and bounds). "Step" denotes the additive mutation increment used when a parameter is selected for mutation; mutations are clamped to the corresponding min/max bounds.

Parameter	Range / value	Role / effect
Neural field parameters (`NeuralFieldConstants`)		
Time constant τ	[1.0, 200.0]	Controls temporal responsiveness of a field (larger $\tau \Rightarrow$ slower dynamics).
τ mutation step	15.0	Additive step for τ mutations (scaled by random sign).
Resting level h (baseline)	$[-15.0, -1.0]$	Baseline activation (more negative $\Rightarrow$ harder to activate).
h mutation step	0.5	Additive step for resting-level mutations (scaled by random sign).
Activation function	Sigmoid(0.0, 5.0)	Output nonlinearity used by the neural field element (threshold and slope parameters).
{Kernel global flags} (`KernelConstants`)		
Kernel circularity		Periodic boundary conditions for kernel convolution.
Kernel normalisation		Normalise kernel profile (affects amplitude interpretation and stability).
Gaussian kernel bounds (`GaussKernelConstants`)		
Width σ	[1.0, 10.0]	Spatial spread of interaction.
Width mutation step	0.50	Additive step for σ mutations.
Amplitude a	[3.0, 30.0]	Interaction strength (self-kernels use positive amplitudes; connection kernels may be signed).
Amplitude mutation step	0.50	Additive step for a mutations.
Global amplitude a_{glob}	$[-0.2, 0.0]$	Uniform inhibitory component (offset).
Global amplitude mutation step	0.05	Additive step for a_{glob} mutations.
Mexican-hat kernel bounds (`MexicanHatKernelConstants`)		
Excitatory width σ_{exc}	[5.0, 30.0]	Spread of local excitation.
Exc.\width mutation step	0.50	Additive step for σ_{exc} mutations.
Inhibitory width σ_{inh}	[5.0, 35.0]	Spread of lateral inhibition.
Inh.\width mutation step	0.50	Additive step for σ_{inh} mutations.
Excitatory amplitude a_{exc}	[15.0, 25.0]	Strength of excitation component.
Exc.\amplitude mutation step	0.50	Additive step for a_{exc} mutations.
Inhibitory amplitude a_{inh}	[1.0, 35.0]	Strength of inhibition component.
Inh.\amplitude mutation step	0.50	Additive step for a_{inh} mutations.
Global amplitude a_{glob}	$[-0.2, 0.0]$	Uniform inhibitory component (offset).
Global amplitude mutation step	0.05	Additive step for a_{glob} mutations.

1.4 Evolutionary algorithm, population management, and speciation

Table 3: Evolutionary algorithm and speciation settings (current defaults). Population pruning removes the worst-performing fraction within each species before reproduction. Stagnation thresholds trigger resets/elimination policies when progress stalls. Speciation uses a NEAT-style compatibility distance with coefficients shown below.

Parameter	Value	Role / effect
Population management (`PopulationConstants`)		
Parallel evolution	true	Enables parallel evaluation/evolution where supported (affects runtime, not search objective).
Prune ratio	0.8	Within each species, remove the worst 80% of members before producing offspring (keeps top 20%).
Elitism	true	Preserves champions of sufficiently large species into the next generation unchanged.
Stagnation criteria (`PopulationConstants`)		
Population stagnation threshold	10	Generations without global improvement before applying population-level reset policy.
Species stagnation threshold	7	Generations without improvement before species is considered stagnant/removed.
Speciation (compatibility distance) (`CompatibilityCoefficients`)		
Compatibility threshold δ_c	3.5	Maximum normalised genetic distance for membership in a species.
Excess genes weight c_1	1.0	Contribution of excess genes (topology mismatch at genome ends).
Disjoint genes weight c_2	0.5	Contribution of disjoint genes (non-aligned genes within genomes).
Connection-difference weight c_3	1.5	Scales the (summed) parameter differences of matching connection genes.
Amplitude diff. coefficient	0.05	Scales amplitude differences when computing connection-gene parameter distance.
Width diff. coefficient	0.05	Scales width differences when computing connection-gene parameter distance.

1.5 Mutation operator probabilities

Mutation in `neat-dnfs` is applied at multiple levels. *Genome-level* probabilities are independent Bernoulli trials executed each time `Genome::mutate()` is called; therefore, their sum is not required to equal 1. In contrast, several categorical choices (e.g., selecting which *class* of field-gene mutation to apply) are implemented as multinomial selections and must sum to 1.

Table 4: Mutation probability distributions (current defaults). Genome-level probabilities are applied independently per mutation call. Several sub-distributions are either (i) multinomial selections (must sum to 1) or (ii) per-parameter Bernoulli trials (do not need to sum to 1).

Mutation type	**Probability**	**Role / effect**
Genome-level mutation triggers (`GenomeMutationConstants`; independent trials)		
Toggle connection gene	0.01	With probability 1%, enable/disable a randomly selected connection gene.
Add field gene	0.0005	With probability 0.05%, add a new field gene.
Mutate field genes (any)	0.8	With probability 80%, apply field-gene mutation operators.
Add connection gene	0.15	With probability 15%, add a new inter-field connection gene.
Mutate connection genes (any)	0.8	With probability 80%, apply connection-gene mutation operators.
Per-field-gene mutation prob.	0.8	When mutating fields, each field gene mutates with probability 80%.
Per-connection-gene mutation prob.	0.8	When mutating connections, each connection gene mutates with probability 80%.
Kernel-type selection (initialisation; must sum to 1)		
Field-gene: Gaussian / Mexican-hat	0.8 / 0.2	Probability of selecting kernel type for new field genes.
Connection-gene: Gaussian / Mexican-hat	0.8 / 0.2	Probability of selecting kernel type for new connection genes.
Field-gene mutation routing (`FieldGeneConstants`; must sum to 1)		
Mutate kernel parameters	0.70	Select kernel-parameter mutation operator.
Mutate neural field parameters	0.20	Select neural-field parameter mutation operator.
Mutate kernel type	0.10	Switch kernel class (Gaussian ↔ Mexican-hat).
Field-gene kernel-parameter mutations (Bernoulli per parameter; do not sum to 1)		
Gaussian: amplitude / width / global amp.	0.80 / 0.60 / 0.20	Independently mutate selected Gaussian-kernel parameters using the step sizes in Table 2.
Mex.-hat: $a_{\text{exc}}/a_{\text{inh}}$ /$\sigma_{\text{exc}}/\sigma_{\text{inh}}/a_{\text{glob}}$	0.80 / 0.80 / 0.60 / 0.60 / 0.20	Independently mutate selected Mexican-hat parameters using the step sizes in Table 2.
Field-gene neural-field mutation routing (must sum to 1)		
Step-wise parameter mutation	0.90	Mutate τ and/or h via additive steps (clamped to bounds).
Random reinitialisation	0.10	Resample τ and h uniformly from their allowed ranges.
Field-gene neural-field parameter trials (Bernoulli per parameter; do not sum to 1)		
Mutate τ / mutate h	0.50 / 0.80	Independently decide whether to apply a τ and/or h step update.
Connection-gene mutation routing (`ConnectionGeneConstants`; must sum to 1)		
Mutate kernel parameters	0.70	Select kernel-parameter mutation operator.
Mutate kernel type	0.05	Switch kernel class (Gaussian ↔ Mexican-hat).
Mutate connection signal	0.25	Mutate between excitatory/inhibitory connections.
Connection-gene kernel-parameter mutations (Bernoulli per parameter; do not sum to 1)		
Gaussian: amplitude / width / global amp.	0.80 / 0.60 / 0.20	Independently mutate selected Gaussian-kernel parameters.
Mex.-hat: $a_{\text{exc}}/a_{\text{inh}}$ /$\sigma_{\text{exc}}/\sigma_{\text{inh}}/a_{\text{glob}}$	0.80 / 0.80 / 0.60 / 0.60 / 0.20	Independently mutate selected Mexican-hat parameters.

1.6 Initialisation constraints and run-time options

Table 5: Additional constants that affect initial topology constraints and (optionally) logging/validation behaviour. These settings do not change the optimisation objective but can affect runtime and debugging.

Parameter	Value	Role / effect
Min. initial input genes (`SolutionConstants::minInitialInputGenes`)	1	Ensures at least one input field gene exists in the initial genome.
Min. initial output genes (`SolutionConstants::minInitialOutputGenes`)	1	Ensures at least one output field gene exists in the initial genome.
Variable field parameters (`FieldGeneConstants::variableParameters`)	true	If true, new fields initialise τ, h, and kernel parameters randomly within bounds (rather than fixed defaults).
Population validation flags (`PopulationConstants::validate*`)	all false	Optional consistency checks for debugging (can increase overhead when enabled).
Logging flags (`PopulationConstants::log*`)	overview true	Controls per-run logging verbosity.
Saving flags (`PopulationConstants::save*`)	all true	Controls which artefacts are written to disk (overview, champions, best solutions, etc.).