Configuration Schema (SFV–dSB Bounce)

This file documents the YAML configuration keys used by the repository. Keys are grouped by top-level sections. Each entry lists **type**, **default** (if any), **units**, **allowed values**, and **notes**.

> All floats are in dimensionless reduced-Planck units unless otherwise stated. A tilde in the paper (\~) corresponds to these dimensionless quantities in code/config.

---

1) `meta`

key	type	default	notes
`name`	string	required	Short run label used in folder names/log headers.
`description`	string	""	Optional free text.
`seed`	int	42	Seed passed to any randomized initialization (if used).

---

2) `model`

key	type	default	units	notes
`units`	string	"reduced Planck"	—	Informational only (logged).
`v`	float	required	M_Pl	Brane symmetry-breaking scale and the rescaling unit.
`vPhi`	float	required	M_Pl	Bulk symmetry-breaking scale. Often written as `v_Φ`.
`lambda`	float	required	—	Brane quartic λ .
`lambdaPhi`	float	required	—	Bulk quartic λ_Φ .
`g_portal`	float	required	—	Portal coupling g .
`mu2`	float or string	"derived"	—	Tachyonic mass parameter for the brane sector. If a number is given, it overrides any derived value.
`bias_rule`	string	"epsilon = -1.01 * DeltaV / vPhi^2"	—	Documentation string only; the code computes ϵ accordingly if `epsilon` is not set explicitly.

key	type	default	units	notes
`epsilon`	float	computed	—	Bias term for the bulk field potential. If omitted, computed from `bias_rule`.

Notes: - If both `epsilon` and `bias_rule` are provided, `epsilon` takes precedence and is logged as "explicit". - The repository code records a fully resolved copy of all parameters in `runs/.../params_resolved.json`.

---

3) `branch_selection`

key	type	default	allowed	notes
`false_vacuum_branch`	string	"negative_phi"	{`negative_phi`, `positive_phi`}	Chooses which ϕ branch is treated as the false vacuum.
`enforce_branch`	bool	true	—	When true, the initial path and boundary checks keep $\phi \leq 0$ (or ≥ 0) as required.
`offset_phi0`	float	1e-6	—	Small nudge used to avoid numerical sign flips at the start of the path.

---

4) `solver`

key	type	default	units	notes
`engine`	string	"scipy.solve_bvp"	—	Informational; current implementation uses SciPy BVP with analytic Jacobian.
`tol`	float	1e-5	—	Nonlinear BVP tolerance.
`max_nodes`	int	500000	—	Hard guardrail for adaptive mesh size (collocation nodes).
`rmax`	float	100.0	—	Finite box size for the asymptotic boundary. Increase for thick walls.
`analytic_jacobian`	bool	true	—	Use analytic Jacobian for speed/stability.
`shooting_refine`	bool	false	—	Optional pre-refinement of seed via 1D shooting (if supported).
`continuation.parameter`	string	"g_portal"	—	Parameter to vary during continuation.
`continuation.schedule`	list[float]	[0.0,0.2,0.4,0.8,1.2,1.6,2.0]	—	Sequence of parameter values including the target.

key	type	default	units	notes
`continuation.step_halving`	bool	true	—	On failure, insert midpoints between scheduled values until solve succeeds or `min_step` is reached.
`continuation.min_step`	float	0.01	—	Minimum spacing when halving steps.
`stability_checks.mesh_factor`	float	2.0	—	Factor for mesh doubling check (optional).
`stability_checks.tol_grid`	list[float]	[1e-5,2e-5]	—	Tolerance sweep for robustness (optional).

5) `initial_guess`

key	type	default	units	notes
`R0`	float	12.0	—	Seed radius for the wall ($\tanh$ -like ansatz).
`w`	float	3.0	—	Seed wall width.
`profile`	string	"tanh-wall"	{`tanh-wall`, `gaussian`, `interp`}	Selects the functional form of the initial profile.
`interp_file`	string	""	—	If `profile: interp`, path to CSV with columns `r,Phi,phi`.

6) `outputs`

key	type	default	notes
`outdir`	string	`./runs/benchmark_${DATE}/`	Output directory. `${DATE}` expands to timestamp.
`save_plots`	bool	true	Save PNG figures listed under `figures`.
`save_csv`	bool	true	Write `results.csv` summarizing the run.
`save_logs`	bool	true	Write `solver_log.txt` (iterations, residuals, diagnostics).

7) `figures`

key	type	default	notes
`profile_png`	string	`profile_phiPhi.png`	Φ(r) and ϕ(r) profiles.
`action_density_png`	string	`action_density.png`	Radial action density plot.
`fieldspace_png`	string	`fieldspace_contours_path.png`	Field-space contours with path overlay.

8) `scan` (for `bounceScan_v3.py`)

key	type	default	notes
`vary`	string	required	Parameter to scan: e.g., `rho`, `lambda`, `lambdaPhi`, `g_portal`, `vPhi`.
`grid.start`	float	required	Start of range.

key	type	default	notes
`grid.stop`	float	required	End of range (inclusive handling defined by script).
`grid.step`	float	required	Step size.
`per_point.solver_tol`	float	1e-5	Tolerance per scan point.
`per_point.max_time_s`	int	900	Soft timeout per point (optional).
`per_point.retries`	int	2	Retry count with refined mesh/step.
`record.write_csv`	string	`runs/scans/scan_results.csv`	Output CSV path.
`record.fields`	list[string]	["param","converged","SE","R0","w","nodes","tol"]	Columns to include in CSV.
`record_per_point`	bool	false	If true, also write per-point plots/logs in subfolders.

Special case: If `vary: rho`, the script interprets `rho = vPhi / v` with `v` fixed by `model.v`.

---

9) `grid` / `normalize_to_false_vacuum` / `overlay` / `style` (for field-space plot)

These appear in `config_fieldspace_plot.yaml`.

section.key	type	default	notes
`grid.Phi_range`	list[expr or float, expr or float]	[-2.0*vPhi, 2.0*vPhi]	Allowed to reference `vPhi` using a simple expression parser.
`grid.phi_range`	list[expr or float, expr or float]	[-1.5*phi_FV, 1.5*phi_FV]	`phi_FV` is inferred from the model (negative branch magnitude).
`grid.n_Phi`	int	300	Number of Phi grid points.
`grid.n_phi`	int	300	Number of phi grid points.
`normalize_to_false_vacuum`	bool	true	Subtracts V_{false} so contours are relative.
`overlay.show_path`	bool	true	Draw the bounce path if available.
`overlay.mark_TV_FV`	bool	true	Label the vacua.
`style.contour_levels`	int	25	Number of contour levels.
`style.label_inline`	bool	true	Inline contour labels when supported.
`output.png`	string	`fieldspace_contours_path.png`	Output filename.

Defaults Summary (for quick reference)

```yaml
meta:
  seed: 42
model:
  units: "reduced Planck"
branch_selection:
  false_vacuum_branch: negative_phi
  enforce_branch: true
  offset_phi0: 1e-6
solver:
  engine: scipy.solve_bvp
  tol: 1.0e-5
  max_nodes: 500000
  rmax: 100.0
  analytic_jacobian: true
  continuation:
    parameter: g_portal
    schedule: [0.0, 0.2, 0.4, 0.8, 1.2, 1.6, 2.0]
    step_halving: true
    min_step: 0.01
initial_guess:
  R0: 12.0
  w: 3.0
  profile: tanh-wall
outputs:
  outdir: ./runs/benchmark_${DATE}/
  save_plots: true
  save_csv: true
  save_logs: true
figures:
  profile_png: profile_phiPhi.png
  action_density_png: action_density.png
  fieldspace_png: fieldspace_contours_path.png
```

Examples

A) Benchmark (SE ≈ 1424)

```yaml
meta:
  name: sfv_dsb_benchmark_negphi
  description: Negative-phi FV branch; g_portal=2.0; target SE~1424
model:
```

```
  v: 9.0e-5
  vPhi: 9.0e-5
  lambda: 1.3e-4
  lambdaPhi: 1.0e-1
  g_portal: 2.0
  bias_rule: "epsilon = -1.01 * DeltaV / vPhi^2"
branch_selection:
  false_vacuum_branch: negative_phi
  enforce_branch: true
solver:
  tol: 1.0e-5
  max_nodes: 500000
  rmax: 100.0
  continuation:
    parameter: g_portal
    schedule: [0.0, 0.2, 0.4, 0.8, 1.2, 1.6, 2.0]
initial_guess:
  R0: 11.9
  w: 3.0
outputs:
  outdir: ./runs/benchmark_${DATE}/
figures:
  profile_png: profile_phiPhi.png
  action_density_png: action_density.png
  fieldspace_png: fieldspace_contours_path.png
```

B) Scan in ρ = vPhi/v

```
meta:
  name: rho_scan_negphi
model:
  v: 9.0e-5
  lambda: 1.3e-4
  lambdaPhi: 1.0e-1
  g_portal: 2.0
scan:
  vary: rho
  grid: {start: 0.4, stop: 2.5, step: 0.05}
  per_point:
    solver_tol: 1.0e-5
    max_time_s: 900
    retries: 2
record:
  write_csv: runs/scans/rho_scan_results.csv
  fields: ["rho","converged","SE","R0","w","nodes","tol"]
```

C) Field-space contour plot

```
grid:
  Phi_range: [-2.0*vPhi, 2.0*vPhi]
  phi_range: [-1.5*phi_FV, 1.5*phi_FV]
  n_Phi: 300
  n_phi: 300
normalize_to_false_vacuum: true
overlay:
  show_path: true
  mark_TV_FV: true
style:
  contour_levels: 25
  label_inline: true
output:
  png: fieldspace_contours_path.png
```

Logging and Provenance

- At start of each run, scripts record: Python, NumPy, SciPy, OS, and git commit (if available).
- `solver_log.txt` contains iteration tables (residuals, max boundary residual, nodes added).
- `params_resolved.json` captures the fully expanded numeric parameter set actually used.

Backward/Forward Compatibility

- New keys should be optional with sensible defaults.
- Unknown keys are ignored but echoed in logs to aid debugging.
- If a key becomes required, the script should emit a clear error with a suggested default.