Guide 6: Solver setup

The setup file, its keys, and the solver commands behind them

Written for pyflightstream 0.58.2.

Source: the package page "The workspace and the workflow" and the setup code [1].

pyflightstream is the work of Geovana Neves: https://github.com/nevesgeovana/pyflightstream, concept DOI 10.5281/zenodo.21482924.
Copyright (c) 2026 Geovana Neves. Licensed under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/). See LICENSE-AND-AUTHORSHIP.md.

Contents

The setup artifact
Run control
Physics and model
Advanced settings with setup keys
Fourteen new advanced setup keys
Loads keys and body rates
Writing raw entries
Example and checklist

Sources under `src/pyflightstream/`: `workspace/matrix/`; `cases/workflows/`; `script/helpers.py`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 1

The setup artifact

Sources under src/pyflightstream/: cases/__init__.py, SolverSettings; workspace/matrix/; cases/workflows/; commands/.

Where the solver setup lives

- The matrix row's `SET` cell names the setup artifact: `s001` selects `inputs/setups/s001.toml`. The general filename is `sNNN.toml`.
- Top-level TOML keys become `SolverSettings` fields, directly or through the preset alias table. A key that reaches nothing and is not declared recorded-only is refused at plan.
- An unstated optional setting normally emits nothing, leaving the solver's own default. The package also has explicit defaults and derived values, listed on the next frame.
- The run record keeps the `solver_setup` snapshot: effective flag values and provenance, including defaults and unknown values. Unknown does not mean disabled.

Sources under `src/pyflightstream/`: `workspace/inputs.py`, `SetupArtifact`; `workspace/matrix/`, `_solver_from_setup`; `script/solver_setup.py`, `SolverSetup`; `script/helpers.py`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Package defaults and solver defaults

- `iterations = 500` and `convergence = 1e-5` are model defaults and reach solver commands even when omitted from the file.
- Every other `SolverSettings` field defaults to Python `None`. In TOML, omit the key; there is no `None` value.
- An omitted `minimum_cp` still emits `SOLVER_MINIMUM_CP -100` on a build carrying that command. The solver's documented default is `-20`.
- The workflow supplies `INCOMPRESSIBLE` for an unstated model, the freestream velocity for an unstated reference velocity, and `DISABLE` for loading saved solver initialization.
- The snapshot distinguishes explicit, default and unknown values. Its flag inventory is `FLAG_SPECS`; that inventory is larger than the set of keys accepted in a setup file.

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `cases/workflows/`; `script/helpers.py`, `solver_settings`; `script/solver_setup.py`, `FLAG_SPECS`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Recorded-only keys

Key	Why it emits nothing
`solver`, `motion`	Run type and motion come from the row and workflow.
`symmetry_type`	Geometry symmetry belongs to the row's `SYMMETRY`.
`unsteady_delta_theta_deg`	Superseded by the row's `DELTA_THETA`.
`unsteady_N_revolutions`	Superseded by the row's `REVOLUTIONS`.
`set_base_region_trailing_edges`	Boundary-selecting separation model: a recipe's job.
`wake_layers`	No emitter in this package.

- Each warns by name and reason. A file may declare additional names with `recorded_only = ["my_setting"]`; these still warn.
- `slipstream_wake_stabilization` emits a setup toggle; it is not recorded-only.
- Do not use recorded-only to make a misspelled setting appear to work. `mesh_order_list` is refused outright: boundary order belongs to the geometry inventory.

Sources under `src/pyflightstream/`: `workspace/matrix/`, `_PRESET_RECORDED_ONLY`, `_PRESET_RECORDED_ONLY_KEY`, `_solver_from_setup`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 2

Run control

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`; `commands/`.

Iteration and convergence controls

Setup key	Solver command	Type and package default
`iterations` (alias `NITER`)	`SOLVER_SET_ITERATIONS`	Integer ≥ 1; `500`.
`convergence`	`SOLVER_SET_CONVERGENCE`	Float > 0; `1e-5`.
`forced_iterations`	`SOLVER_SET_FORCED_ITERATIONS`	Toggle; `None`.
`convergence_iterations`	`SET_SOLVER_CONVERGENCE_ITERATIONS`	Integer ≥ 1; `None`.

- `iterations` is the solver iteration limit. Convergence is the residual threshold on surface velocity and pressure.
- Forced iterations run the full count regardless of convergence. Convergence iterations is how long the residual must remain below the threshold before convergence is declared.
- A toggle accepts `true`/`false` or the solver words `"ENABLE"`/`"DISABLE"`. Numbers must be finite.

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `commands/runtime_settings.yaml`; `commands/advanced_settings.yaml`; `script/helpers.py`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Threads and wall-clock limits

- `max_threads`: integer ≥ 1, default `None`; alias `max_parallel_threads`. Parallel core count, emitted as `SET_MAX_PARALLEL_THREADS`. A stated row `NCPUS` overrides the setup value; an unstated row inherits it.
- `timeout_s`: finite float > 0, default `None`. Wall-clock limit for one point's solver process, enforced by the executor, not by a FlightStream command.
- `walltime_margin_s`: finite float > 0, default `None`. Seconds of the row's `WALLTIME` reserved for exports. When unstated, the wall-clock program uses 20 minutes.
- The margin belongs to the setup. It emits no solver command; the wall-clock program written beside the script reads it.

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `cases/workflows/`, `row_ncpus`; `workspace/matrix/`; `commands/runtime_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

NITER: measured on build 26.124

- One short rotor case, 36 physical time steps: with `NITER = 5`, every step used 5 inner iterations, 180 total; `COMPLETED_MAX_ITER`.
- With `NITER = 300` and `convergence_iterations = 3`, the case converged in 6 to 90 inner iterations per step, median 32.5, 1100 total.
- On this measured build and geometry, `NITER` caps each time step. Convergence iterations counts consecutive iterations satisfying the criterion; it is not an iteration cap.

Budget and scope

Worst-case inner work is the number of time steps times `NITER`. This is one build and one geometry, not verification of every solver build.

Release evidence: measurement record, check `niter_limit` [2].

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 3

Physics and model

Sources under src/pyflightstream/: cases/__init__.py, SolverSettings; workspace/matrix/; cases/workflows/; commands/.

Boundary layer and viscous coupling

- `boundary_layer`: string, default `None`; alias `boundary_layer_type`. Emits `SET_BOUNDARY_LAYER_TYPE` with `LAMINAR`, `TRANSITIONAL` or `TURBULENT`. The database gives `TRANSITIONAL` as the solver default.
- Its notes state the default model's chord Reynolds range as 500000 to 1500000, with an explicit laminar or turbulent choice outside that range [3, p.203].
- `viscous_coupling`: toggle, default `None`; emits `SET_SOLVER_VISCOUS_COUPLING`. It couples the boundary layer model to the potential solution. Stall capture also needs a flow-separation model assignment.
- `reynolds_averaged_drag`: toggle, default `None`; alias `reynolds_averaged_drag_forces`. Emits `REYNOLDS_AVERAGED_DRAG_FORCES`, the flat-plate boundary-layer calculation toggle. No solver default is stated.

Sources under `src/pyflightstream/`: `cases/__init__.py`; `workspace/matrix/`; `cases/workflows/`; `script/helpers.py`; `commands/solver_settings.yaml`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Model and initialization

- `solver_model`: string, default `None`; alias `set_solver_model`. The workflow passes it to `INITIALIZE_SOLVER`, using `INCOMPRESSIBLE` if omitted.
- Accepted model arguments: `INCOMPRESSIBLE`, `SUBSONIC_PRANDTL_GLAUERT`, `TRANSONIC_FIELD_PANEL`, `TANGENT_CONE`, `MODIFIED_NEWTONIAN`.
- `wall_collision_avoidance`: toggle, default `None`; alias `proximity_avoidance`. Another argument of `INITIALIZE_SOLVER`; the database says it applies to the first three models above.
- `load_solver_initialization`: toggle, default `None`. Reaches `OPEN`'s `LOAD_SOLVER_INITIALIZATION` argument. The workflow uses `DISABLE` if omitted, so a saved solver state is not loaded implicitly.

Sources under `src/pyflightstream/`: `cases/__init__.py`; `workspace/matrix/`; `cases/workflows/`; `script/helpers.py`, `initialize_solver`; `commands/solver_initialization.yaml`; `commands/file_io.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Stabilization and pressure floor

- `solver_stabilization`: finite float ≥ 0, default `None`; emits `SOLVER_STABILIZATION <value>`. The database describes a dimensionless strength from 0 to 1; the model does not impose that upper bound. No solver default is stated.
- Alternative preset form: `stabilization = "ENABLE"` and `stabilization_strength = 0.25`. A strength without a gate is enabled. Enabling without a strength is refused.
- `stabilization = "DISABLE"` produces no stabilization line. Do not combine the gated pair with `solver_stabilization`. Zero is an emitted numeric value, even though the manual describes zero strength as equivalent to disabling.
- `minimum_cp`: finite float, default `None`; alias `solver_minimum_cp`. It sets the pressure-coefficient floor with `SOLVER_MINIMUM_CP`. Omission uses the library's -100, overriding the solver default -20.

Sources under src/pyflightstream/: cases/__init__.py; workspace/matrix/, _resolve_stabilization; script/helpers.py; commands/solver_settings.yaml; commands/advanced_settings.yaml.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Boundary families

- `vorticity_drag_families`: list of strings, default `None`; aliases `vorticity_drag_boundaries` and `set_vorticity_drag_boundaries`. Reaches `SET_VORTICITY_DRAG_BOUNDARIES` in analysis.
- `axial_separation_families`: list of strings, default `None`; no alias. Reaches `SET_AXIAL_SEPARATION_BOUNDARIES` in init. Documented through 26.100; a later build is refused at plan.
- Write family names, not indices. Both lists resolve against the opened geometry: absent families are left out; a selection resolving to nothing is refused. An omitted key emits no selection or deletion.
- Without a vorticity selection, the solver uses surface-pressure integration for induced drag. The helper notes advise against a blanket vorticity selection on mixed wing and bluff-body geometry.

Sources under `src/pyflightstream/`: `cases/__init__.py`; `workspace/matrix/`; `cases/workflows/`, `_vorticity_indices`, `_axial_separation_indices`; `script/helpers.py`; `commands/solver_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Reference velocity, precision and symmetry loads

- `reference_velocity_m_per_s`: finite float > 0, default `None`; alias `reference_velocity_mps`. Emits `SOLVER_SET_REF_VELOCITY`, in m/s, for coefficient normalization. Omission makes the workflow state freestream velocity.
- `significant_digits`: integer ≥ 1, default `None`; emits `SET_SIGNIFICANT_DIGITS` in setup. The command database describes output precision from 1 to 8; the model has no upper bound. The model's comment records the solver default as 4 decimals.
- `symmetry_loads`: toggle, default `None`; emits `SET_ANALYSIS_SYMMETRY_LOADS` before solving. It moved to the row's `SYMMETRY_LOADS` column, with the preset still accepted as a fallback.
- A stated row value wins; disagreement with the setup warns. True includes the whole wheel, false the meshed sector. If neither states it, no line is emitted.

Sources under `src/pyflightstream/`: `cases/__init__.py`; `workspace/matrix/`; `cases/workflows/`, `_row_symmetry_loads`; `commands/runtime_settings.yaml`; `commands/simulation_controls.yaml`; `commands/solver_analysis.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 4

Advanced settings with setup keys

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`; `commands/`.

Convergence and pressure floor

All commands here are inline, in phase `init`. Package default: `None` for every key below. Floats must be finite.

Key	Alias	Solver command	Type	Solver default	26.123 / 26.124
`convergence_iterations`	None	`SET_SOLVER_CONVERGENCE_ITERATIONS`	int ≥ 1	Not stated	V / V
`minimum_cp`	`solver_minimum_cp`	`SOLVER_MINIMUM_CP`	float	-20	V / V

V: verified on that build. **D**: documented, not verified on that build.

The first is the count below the residual threshold required for convergence. The second is the pressure-coefficient floor. An omitted minimum Cp emits the library default −100; the solver default is −20.

Manual references in the database: [3, p.344]; [3, p.345].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `cases/workflows/`; `script/helpers.py`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Drag and mesh-induced wake velocity

All commands here are inline, in phase `init`. Package default: `None` for every key below. Floats must be finite.

Key	Alias	Solver command	Type	Solver default	26.123 / 26.124
`reynolds_averaged_drag`	`reynolds_averaged_drag_forces`	`REYNOLDS_AVERAGED_DRAG_FORCES`	toggle	Not stated	V / D
`mesh_induced_wake_velocity`	`induced_wake_velocity`	`SOLVER_SET_MESH_INDUCED_WAKE_VELOCITY`	toggle	Not stated	V / D

V: verified on that build. **D**: documented, not verified on that build.

`reynolds_averaged_drag` switches the flat-plate boundary-layer calculations. For these toggles, the database gives no solver default; omission does not assert `DISABLE`.

Manual references in the database: [3, p.344].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `cases/workflows/`; `script/helpers.py`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Farfield and unsteady pressure

All commands here are inline, in phase `init`. Package default: `None` for every key below. Floats must be finite.

Key	Alias	Solver command	Type	Solver default	26.123 / 26.124
`farfield_layers`	None	`SOLVER_SET_FARFIELD_LAYERS`	int 1 to 5	3	V / D
`unsteady_pressure_and_kutta`	`unsteady_pressure_kutta`	`SOLVER_UNSTEADY_PRESSURE_AND_KUTTA`	toggle	Not stated	V / D

V: verified on that build. **D**: documented, not verified on that build.

The documented farfield layer range is **1 to 5**, solver default **3**; a value above 5 is refused, and every standard setup states **5**. The pressure toggle controls the unsteady Bernoulli and Kutta terms.

Manual references in the database: [3, p.344].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `cases/workflows/`; `script/helpers.py`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Wake termination units

All commands here are inline, in phase `init`. Package default: `None` for every key below. Floats must be finite.

Key	Alias	Solver command	Type	Solver default	26.123 / 26.124
`wake_termination_steps`	None	`SET_WAKE_TERMINATION_TIME_STEPS`	int	Not stated	D / D
`wake_termination_revolutions`	`unsteady_N_revolutions_wake`	`SET_WAKE_TERMINATION_TIME_STEPS`	float	Not stated	D / D

V: verified on that build. **D**: documented, not verified on that build.

The command takes integer time steps; revolutions use the rotor clock. The alternative length form defaults to 4 rotor radii (row-clock slide).

Manual references in the database: [3, p.344].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `cases/workflows/`; `script/helpers.py`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Wake induction and extra relaxation

All commands here are inline, in phase `init`. Package default: `None` for every key below. Floats must be finite.

Key	Alias	Solver command	Type	Solver default	26.123 / 26.124
`wake_on_wake_induction`	None	`SET_WAKE_ON_WAKE_INDUCTION`	toggle	Not stated	V / D
`additional_wake_relaxation`	`additional_wake_relaxation_iteration`	`ADDITIONAL_WAKE_RELAXATION_ITERATION`	toggle	Not stated	V / D

V: verified on that build. **D**: documented, not verified on that build.

The first switches wake-on-wake induced velocity computation. The second requests one additional wake relaxation iteration. Neither command has a stated solver default.

Manual references in the database: [3, pp.344-345]; [3, p.345].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `cases/workflows/`; `script/helpers.py`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Wake termination and the row clock

- `wake_termination_length` states rotor radii (positive). An unstated rotor termination uses 4 radii. Axial convection speed converts that length to time steps; it is a model assumption, not a measured native wake speed.
- `wake_termination_thrust_n` supplies the momentum-theory convection estimate; `wake_termination_revolutions_cap` bounds the length conversion.
- `wake_termination_steps` supplies the integer directly. For `wake_termination_revolutions`, the workflow computes `round(revolutions * steps_per_revolution)` and converts to an integer.
- Negative steps or revolutions count backwards from the run's end. State one of length, steps and revolutions; the rotor resolver refuses combinations.
- Revolutions require a rotor speed and a clock. A steady row or a nonrotating unsteady row cannot convert them; use the run type's supported clock and the steps form where appropriate.
- The database describes the command as the number of time steps after which a wake vortex filament edge that has lost all its strength is removed. It states no default.

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `cases/workflows/`; `script/helpers.py`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 5

Fourteen new advanced setup keys

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`; `commands/`.

Advanced setup keys (1/4)

Setup key	Solver command
laminar_separation	LAMINAR_SEPARATION
kutta_joukowski_lift	KUTTA_JOUKOWSKI_LIFT_FORCES
aeroelastic_rbf_type	AEROELASTIC_RBF_TYPE
print_rotor_induced_velocities	PRINT_ROTOR_INDUCED_VELOCITIES

- A stated key emits its command; an omitted key emits nothing.
- Unknown keys, unsupported values and unavailable commands are refused, naming the build.
- The command tables below retain the per-build status. A new key does not establish licensed verification.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Advanced setup keys (2/4)

Setup key	Solver command
`adaptive_field_grid_refinement`	`SET_ADAPTIVE_FIELD_GRID_REFINEMENT`
`rotor_induced_velocity_blending`	`ROTOR_INDUCED_VELOCITY_BLENDING`
`wake_numerical_relaxation`	`SET_WAKE_NUMERICAL_RELAXATION`
`wake_relaxation`	`SET_WAKE_RELAXATION`

- A stated key emits its command; an omitted key emits nothing.
- Unknown keys, unsupported values and unavailable commands are refused, naming the build.
- The command tables below retain the per-build status. A new key does not establish licensed verification.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Advanced setup keys (3/4)

Setup key	Solver command
wake_decay_constant_per_m	SET_WAKE_DECAY_CONSTANT
wake_streamwise_agglomeration	SET_WAKE_STREAMWISE_AGGLOMERATION
jet_wake_decay_normalized_length	SET_JET_WAKE_DECAY_NORMALIZED_LENGTH
jet_wake_filaments_grid_induction	SET_JET_WAKE_FILAMENTS_GRID_INDUCTION

- A stated key emits its command; an omitted key emits nothing.
- Unknown keys, unsupported values and unavailable commands are refused, naming the build.
- The command tables below retain the per-build status. A new key does not establish licensed verification.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Advanced setup keys (4/4)

Setup key	Solver command
`adverse_gradient_boundary_layer`	`SOLVER_SET_ADVERSE_GRADIENT_BOUNDARY_LAYER`
`vortex_ring_normalization`	`SOLVER_VORTEX_RING_NORMALIZATION`

- A stated key emits its command; an omitted key emits nothing.
- Unknown keys, unsupported values and unavailable commands are refused, naming the build.
- The command tables below retain the per-build status. A new key does not establish licensed verification.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Setup toggles for separation and lift

These commands have setup keys. Use the keys on a supported build; `[[raw]]` remains available. Phase: `init`.

Command	Argument form	Solver default	26.123 / 26.124
`LAMINAR_SEPARATION`	`ENABLE` or `DISABLE`	Not stated	D / D
`KUTTA_JOUKOWSKI_LIFT_FORCES`	`ENABLE` or `DISABLE`	Not stated	D / D

D: documented, not verified on that build.

Laminar separation is a boundary-layer separation switch. The lift command selects bound-circulation lift instead of surface-pressure integration; the database does not identify every affected output quantity.

Manual references: [3, p.345]; [3, p.344].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`, `_settings`, `_raw_commands`; `script/solver_setup.py`, `FLAG_SPECS`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Setup output and field-grid toggles

These commands have setup keys. Use the keys on a supported build; `[[raw]]` remains available. Phase: `init`.

Command	Argument form	Solver default	26.123 / 26.124
`PRINT_ROTOR_INDUCED_VELOCITIES`	`ENABLE` or `DISABLE`	Not stated	D / D
`SET_ADAPTIVE_FIELD_GRID_REFINEMENT`	`ENABLE` or `DISABLE`	Not stated	D / D

D: documented, not verified on that build.

The print switch writes rotor-induced velocities to the log at every unsteady time step. The refinement note marks the field-grid option as transonic only; no subsonic behavior is established.

Manual references: [3, p.345].

Sources under src/pyflightstream/: cases/__init__.py, SolverSettings; workspace/matrix/; cases/workflows/, _settings, _raw_commands; script/solver_setup.py, FLAG_SPECS; commands/advanced_settings.yaml.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Setup wake factors

These commands have setup keys. Use the keys on a supported build; `[[raw]]` remains available. Phase: `init`.

Command	Argument form	Solver default	26.123 / 26.124
`ROTOR_INDUCED_VELOCITY_BLENDING`	`<float>`	0.25	D / D
`SET_WAKE_NUMERICAL_RELAXATION`	`<float>`	0.15	D / D
`SET_JET_WAKE_DECAY_NORMALIZED_LENGTH`	`<float>`	100.0	D / D

D: documented, not verified on that build.

The two factors are documented from 0 to 1. Jet decay length is in jet-wake diameters, with a documented minimum of 1.0: it is the distance to one tenth of the initial strength. These notes do not enforce the bounds.

Manual references: [3, p.345]; [3, p.346].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`, `_settings`, `_raw_commands`; `script/solver_setup.py`, `FLAG_SPECS`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Setup wake decay and jet induction

These commands have setup keys. Use the keys on a supported build; `[[raw]]` remains available. Phase: `init`.

Command	Argument form	Solver default	26.123 / 26.124
`SET_WAKE_DECAY_CONSTANT`	`<float>`	Not stated	D / D
`SET_JET_WAKE_FILAMENTS_GRID_INDUCTION`	`ENABLE` or `DISABLE`	Not stated	Refused / refused

D: documented, not verified on that build.

The decay constant is in inverse metres, derived in the database from `19.1/L`, with `L` in metres. No fixed default is recorded. Jet-filament grid induction has evidence only on 26.101 and 26.120; it is refused on 26.123 and 26.124.

Manual references: [4, p.346]; [3, p.346].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`, `_settings`, `_raw_commands`; `script/solver_setup.py`, `FLAG_SPECS`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Build-limited wake commands

These commands have setup keys. Use the keys on a supported build; `[[raw]]` remains available. Phase: `init`.

Command	Argument form	Solver default	26.123 / 26.124
`SET_WAKE_RELAXATION`	`ENABLE` or `DISABLE`	Not stated	Refused / refused
`SET_WAKE_STREAMWISE_AGGLOMERATION`	`ENABLE` or `DISABLE`	Not stated	Refused / refused

D: documented, not verified on that build.

Both are documented on 25.000, 25.100 and 26.000. The database records their removal from later documentation. Neither is available to emit on 26.123 or 26.124; `[[raw]]` does not bypass the build check.

Manual references: [5, p.319]; [5, p.318].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`, `_settings`, `_raw_commands`; `script/solver_setup.py`, `FLAG_SPECS`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Build-limited boundary-layer and vortex commands

These commands have setup keys. Use the keys on a supported build; `[[raw]]` remains available. Phase: `init`.

Command	Argument form	Solver default	26.123 / 26.124
`SOLVER_SET_ADVERSE_GRADIENT_BOUNDARY_LAYER`	`ENABLE` or `DISABLE`	Not stated	Refused / refused
`SOLVER_VORTEX_RING_NORMALIZATION`	`ENABLE` or `DISABLE`	Not stated	Refused / refused

D: documented, not verified on that build.

Both are documented on 25.000, 25.100 and 26.000 and refused on 26.123 and 26.124. The database gives no definition of the vortex normalization beyond the switch, so no numerical mechanism is asserted here.

Manual references: [5, p.318].

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`, `_settings`, `_raw_commands`; `script/solver_setup.py`, `FLAG_SPECS`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Setup aeroelastic RBF choice

`AEROELASTIC_RBF_TYPE <rbf_type>` is an inline `init` command. The setup key is `aeroelastic_rbf_type`. Its argument is one of `WENDLAND_C2`, `GAUSSIAN`, `THIN_PLATE_SPLINE`, `MULTI_QUADRATIC` or `INV_MULTI_QUADRATIC`.

Solver default: **not stated**. Status: **documented** on both 26.123 and 26.124 [3, pp.345-346]. The manual also describes numbered choices, but the command database's accepted grammar uses the names above. Use a quoted name as the setup value.

What the choice is

A radial basis function interpolates the structure's displacements, known at its nodes, onto every aerodynamic panel [9, 10, 11]. The [REF-44], [REF-45], [REF-46] multiquadric [12] has global support; the Wendland C2 [13] is compact. A [REF-47], [REF-48] coupled FSI row states its own choice: the multiquadric for a beam line of structural nodes (Guide 8).

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`, `_settings`, `_raw_commands`; `script/solver_setup.py`, `FLAG_SPECS`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 6

Loads keys and body rates

Sources under src/pyflightstream/: cases/__init__.py, SolverSettings; workspace/matrix/; cases/workflows/; commands/.

Loads analysis keys on steady rows

Setup key	Solver command	Value
`analysis_families`	`SET_SOLVER_ANALYSIS_BOUNDARIES`	family names; every other boundary leaves the loads. Resolved like `vorticity_drag_families`
`load_units`	`SET_LOADS_AND_MOMENTS_UNITS`	`COEFFICIENTS`, `NEWTONS`, `KILO-NEWTONS`, `POUND-FORCE`, `KILOGRAM-FORCE`
`inviscid_loads`	`SET_INVISCID_LOADS`	toggle: the loads without their viscous part

- Stated after `START_SOLVER` on every point. The solver's own spellings (`set_loads_and_moments_units`, ...) are aliases.
- A row of an unsteady run type stating one is **refused at plan**.
- A point whose loads table is not in `COEFFICIENTS` writes **no product row**; its export is still collected and hashed, and the post names the unit.

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`, `_PRESET_ALIASES`; `cases/workflows/`; `commands/solver_analysis.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Two keys 26.124 does not answer

Setup key	Solver command	On 26.124
`vorticity_lift_model`	`SET_VORTICITY_LIFT_MODEL`	removed: refused at plan
`unsteady_viscous_coupling_iteration`	`SET_UNSTEADY_VISCOUS_COUPLING_ITERATION`	removed: refused at plan

- Both are stated before `INITIALIZE_SOLVER`. The lift model is a toggle on every run type; the coupling step is an integer ≥ 1, unsteady run types only, and emits on 25.100 and 26.000 alone.
- 26.124 answers both names as *Unrecognized command*, word for word as an unknown name [2]. A `[[raw]] SET_VORTICITY_LIFT_MODEL` on a 26.124 row is refused too.
- The lift model beside `kutta_joukowski_lift = true` plans with a warning.

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `commands/advanced_settings.yaml`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Body rates: a cell of the row, not the setup

`roll_rate`, `pitch_rate` and `yaw_rate` are keys of `FLIGHT_CONDITION`, in deg/s, one non-zero rate per row; the reference's `[body_axes]` must use roll=X, pitch=Y, yaw=Z. Permutations are refused: RPT-052 and RPT-060 did not measure their signs.

```
MACH:0.2, REmi:5.5, ALPHA:0, roll_rate:sweep | -40,0,40
```

Rate	Positive is	Emitted for +40 deg/s	Measured on 26.124
`roll_rate`	right wing down	`ROTATION <frame> X -6.667`	rolling moment opposes the roll [2]
`pitch_rate`	nose up	`ROTATION <frame> Y 6.667`	pitching moment nose down [2]
`yaw_rate`	nose right	`ROTATION <frame> Z -6.667`	advancing left wing gains lift [2]

Older rows: reversed sign

Roll and yaw were emitted with the opposite sign and solved at $-p$ or $-r$. Redo such points with `--force-rerun`; `--resume` skips them.

Sources under `src/pyflightstream/`: `cases/workflows/`, `FREESTREAM_ROTATION_SIGN`; `workspace/matrix/`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 7

Writing raw entries

Sources under src/pyflightstream/: cases/__init__.py, SolverSettings; workspace/matrix/; cases/workflows/; commands/.

The raw table and its checks

- Each `[[raw]]` table contains `command`, the complete one-line command with arguments, and `before`, its insertion phase.
- Phases: `geometry`, `setup`, `init`, `exec`, `analysis`, `export`, `control`. The line goes before the first command of the named phase; `control` places it at the script's head.
- At plan, the emitter checks that the command is available on the row's build, that its grammar and argument count and types match, and that the script stays in phase order. A command from a later phase cannot be placed before an earlier one.
- One-line commands only. Keyword blocks, payloads and parameter blocks are refused by this route. A helper keyword alone does not make a top-level setup key valid.
- Lines retain their written order. The run record's `raw_commands` keeps `command`, `before`, `setup` and `source`; the provenance document carries them too.

Sources under `src/pyflightstream/`: `cases/__init__.py`, `RawCommand`; `cases/workflows/`, `_raw_commands`; `workspace/inputs.py`; also repository `docs/workspace-and-workflows.md`, raw commands section.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Setup settings and the pproc surface window

In the setup, a dedicated key replaces the raw separation command:

```
laminar_separation = true
```

In the separate pproc artifact, request the surface average:

```
[time_averaging]
last_iters = 36
```

- With 144 time steps, the package resolves the window to 109 to 144. Exactly one of `last_iters` or `last_revs` is required.
- Surface averaging requires build 26.122 or later. Verified on 26.122 [2]: the bounds are time steps, inclusive, 1-based.
- Existing `[[raw]]` entries still work. Use one route per setting; the pproc window also records the surface's provenance.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

What belongs beside the settings

- A raw line is fixed for every row citing this setup. Row-specific `RAW` entries extend the preset: at a given phase, preset lines come first and row lines follow.
- `[[flags]]` declares a command whose value the row supplies. It reaches only `control`, `geometry` and `setup`. The advanced `init` commands in this deck therefore need `[[raw]]` when they have no top-level setup key.
- `[flight_condition]` can supply fluid pins; it is consumed separately from `SolverSettings`. It does not create new solver keys.
- Geometry `[aliases]` and `[[frames]]` moved to the reference artifact. Post-processing tables belong to the `PPROC` artifact. Their old setup location is refused.
- Do not state a field and its alias together. The resolver refuses two spellings of the same setting, even when the values agree.

Sources under src/pyflightstream/: workspace/inputs.py, SetupArtifact; workspace/matrix/, _solver_from_setup; cases/__init__.py, CustomFlag; cases/workflows/.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 8

Example and checklist

Sources under src/pyflightstream/: cases/__init__.py, SolverSettings; workspace/matrix/; cases/workflows/; commands/.

Complete setup example: settings

`inputs/setups/s901.toml`, selected by `SET = s901`. An illustrative unsteady setup for 26.123 or 26.124, continued next frame.

```
iterations = 500
convergence = 1e-5
forced_iterations = false
convergence_iterations = 5
max_threads = 8
timeout_s = 3600.0
walltime_margin_s = 1200.0
boundary_layer = "TURBULENT"
viscous_coupling = true
solver_model = "INCOMPRESSIBLE"
wall_collision_avoidance = false
solver_stabilization = 0.25
minimum_cp = -100.0
reynolds_averaged_drag = false
reference_velocity_m_per_s = 40.0
significant_digits = 7
```

Sources under `src/pyflightstream/`: `cases/__init__.py`, `SolverSettings`; `workspace/matrix/`; `cases/workflows/`; `script/helpers.py`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Complete setup example: wake settings

Continue the same file. Use geometry carrying the `Wing` family; these are example choices.

```
load_solver_initialization = false
symmetry_loads = false
vorticity_drag_families = ["Wing"]
mesh_induced_wake_velocity = true
farfield_layers = 5          # the recommended value; above 5 is refused
unsteady_pressure_and_kutta = true
wake_termination_steps = 72
wake_on_wake_induction = true
additional_wake_relaxation = false
laminar_separation = true
```

Sources under src/pyflightstream/: cases/__init__.py, SolverSettings; workspace/matrix/; cases/workflows/; commands/advanced_settings.yaml; commands/unsteady_solver.yaml.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Before planning the row

1. Check the `SET` id and the file under `inputs/setups/`. Keep settings above `[[raw]]` tables so TOML does not put them inside the last raw entry.
2. Check each key, alias, type and bound. Toggle strings are quoted; TOML booleans are not. Omit optional keys for `None`.
3. Check the row's build and workflow. Set the physical clock on the row; choose one wake-termination unit. Family names must resolve on that geometry.
4. Check row overrides for `NCPUS` and `SYMMETRY_LOADS`. Read recorded-only warnings; they mean no solver command was emitted.
5. Run `pyfs-matrix plan`. Inspect the emitted commands, including the minimum-Cp default and each raw line, before running the solver. `pyfs-matrix inspect-setups` shows each resolved value and where it came from, and `plan --setup-standards --setup-guidelines` writes the `s9XX` presets and their guide.
6. After the run, inspect `solver_setup` and `raw_commands` in the run record. A documented command is not a measured result on that build.

Sources under `src/pyflightstream/`: `workspace/inputs.py`; `workspace/matrix/`; `cases/workflows/`; `script/solver_setup.py`; **also repository** `docs/workspace-and-workflows.md`.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

The physics behind the keys, and where to read it

Keys	The physics, and its origin
The solve itself	a panel method: source and doublet singularities on the surface, one boundary condition per panel [6, 7]
`solver_model`	incompressible potential flow [7]; the linearised subsonic correction of Prandtl and Glauert [8]; the tangent-cone and modified Newtonian pressure laws of supersonic and hypersonic flow [14]
`unsteady_pressure_and_kutta`, trailing edges	the Kutta condition at a sharp trailing edge, and its unsteady form [7, 19]
Wake relaxation, termination, decay	a force-free wake moved with the local velocity; a time-stepping wake that grows one row per step [7]
`boundary_layer`, `viscous_coupling`	the boundary layer and its transition [15], coupled to the inviscid solution [16]
Separation models and criteria	turbulent separation by a pressure-gradient criterion [17]; a maximum-lift criterion [18]
`aeroelastic_rbf_type`	radial basis function interpolation of a structure's displacements [9, 10, 11, 12, 13]
`SYMMETRY`, periodic copies	the method of images for a mirror plane, and copies turned about an axis [7]

Copyright (c) 2026 Geovana Neves, CC BY 4.0

References

[1] Geovana Neves, *pyflightstream documentation*, page "The workspace and the workflow". `https://nevesgeovana.github.io/pyflightstream/latest/workspace-and-workflows/`

[2] Geovana Neves, *pyflightstream: measurement records of the solver's behaviour*, 2026. `https://github.com/nevesgeovana/pyflightstream`, concept DOI `10.5281/zenodo.21482924`.

[3] Altair Engineering, *FlightStream User Manual*, version 26.12, 2026.

[4] Altair Engineering, *FlightStream User Manual*, version 26.12 (build 26121), 2026.

[5] Altair Engineering, *FlightStream User Guide*, version 26.0, 2025.

REF-28 J. L. Hess, A. M. O. Smith, "Calculation of potential flow about arbitrary bodies", *Progress in Aerospace Sciences*, 8, pp. 1-138, 1967.

REF-29 J. Katz, A. Plotkin, *Low-Speed Aerodynamics*, 2nd ed., Cambridge University Press, Cambridge, 2001.

REF-31 H. Glauert, "The effect of compressibility on the lift of an aerofoil", *Proceedings of the Royal Society of London, Series A*, 118(779), pp. 113-119, 1928.

REF-44 A. Beckert, H. Wendland, "Multivariate interpolation for fluid-structure-interaction problems using radial basis functions", *Aerospace Science and Technology*, 5(2), pp. 125-134, 2001.

REF-45 A. de Boer, M. S. van der Schoot, H. Bijl, "Mesh deformation based on radial basis function interpolation", *Computers and Structures*, 85(11-14), pp. 784-795, 2007.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

References (continued)

REF-46 T. C. S. Rendall, C. B. Allen, "Unified fluid-structure interpolation and mesh motion using radial basis functions", *International Journal for Numerical Methods in Engineering*, 74(10), pp. 1519-1559, 2008.

REF-47 R. L. Hardy, "Multiquadric equations of topography and other irregular surfaces", *Journal of Geophysical Research*, 76(8), pp. 1905-1915, 1971.

REF-48 H. Wendland, "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree", *Advances in Computational Mathematics*, 4(1), pp. 389-396, 1995.

REF-49 J. D. Anderson, *Modern Compressible Flow: With Historical Perspective*, 3rd ed., McGraw-Hill, New York, 2003.

REF-39 H. Schlichting, K. Gersten, *Boundary-Layer Theory*, 9th ed., Springer, Berlin, 2017.

REF-40 M. Drela, M. B. Giles, "Viscous-inviscid analysis of transonic and low Reynolds number airfoils", *AIAA Journal*, 25(10), pp. 1347-1355, 1987.

REF-41 B. S. Stratford, "The prediction of separation of the turbulent boundary layer", *Journal of Fluid Mechanics*, 5(1), pp. 1-16, 1959.

REF-17 W. O. Valarezo, V. D. Chin, "Method for the prediction of wing maximum lift", *Journal of Aircraft*, 31(1), pp. 103-109, 1994.

REF-30 W. M. Kutta, "Auftriebskräfte in strömenden Flüssigkeiten", *Illustrierte Aeronautische Mitteilungen*, 6, pp. 133-135, 1902.

Copyright (c) 2026 Geovana Neves, CC BY 4.0