# Guide 3: From the GUI to pyfs

Every step of a GUI session, and the key that does it in the workspace

Written for pyflightstream 0.58.2.

Source: the package page "From the GUI to pyfs" [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.

# How to read this deck

- A GUI session is a sequence of steps: load the geometry, mark where its wakes leave it, place frames and set things turning, state the flight condition and the solver, run, export. The deck takes them in that order, one table per stage.
- **In pyfs** says which file the key lives in: a *row key* (a matrix column or the `VAR_NAMES_VALUES` cell), a *setup* key, a *pproc* table, the *reference*, or the *sidecar* `<stem>.boundaries.toml` beside the geometry.
- **DB evidence**: builds covered by the database status, including inherited evidence. Inherited status is not a probe run on that build. Consult the command reference for direct evidence. *none*: no verified coverage; *no command*: writes none.
- **not yet**: no key of its own. The raw route still takes it (last part).

| Stage | Key | Not yet |
|---|---|---|
| Geometry and mesh | 7 | 6 |
| Boundary conditions | 14 | 0 |
| Frames, motion, actuators | 11 | 5 |
| Flight conditions, solver | 24 | 0 |
| The run | 12 | 0 |
| Basic post | 25 | 5 |
| **109 steps** | **93** | **16** |

Source: `docs/gui-to-pyfs.md`, How to read a line.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Contents

Geometry and mesh
Boundary conditions
Frames, motion and actuators
Flight conditions and solver
The run
Basic post
The raw route

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 1

# Geometry and mesh

# Geometry and mesh: what a key takes (1/2)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Open a saved simulation | row key `GEOMETRY` naming a `.fsm`; setup key `load_solver_initialization` loads the solver state it was saved with | 26.100 to 26.124 |
| See the boundaries a saved simulation carries, in its order | sidecar key `boundaries`, which `pyfs-mesh inventory` writes from the file | none |
| Start a new simulation and import a mesh file in its length unit | `pyfs-mesh mark-boundaries <mesh>.obj --apply` imports an OBJ once and saves the `.fsm` that row key `GEOMETRY` names; an STL is first prepared into a saved body or an OBJ; sidecar table `[import]` with its `units`, and sidecar key `boundaries`, the file's surface names in its order, which `pyfs-mesh inventory` writes from an OBJ's groups; setup key `simulation_length_unit` states the simulation's own length unit | 26.101 to 26.124, except one |

Source: `docs/gui-to-pyfs.md`, Geometry and mesh.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Geometry and mesh: what a key takes (2/2)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Scale, mirror, rename, move or turn a surface of the imported mesh | sidecar table `[[import.operations]]`, one per operation | none |
| Turn part of the geometry for one row: a flap, a blade's pitch | row key `ROTATE` | none |
| Move part of the geometry for one row | row key `TRANSLATE` | none |
| Name the configuration you loaded | row key `CONFIGURATION`, a label the script carries as a comment on its first line | none |

Source: `docs/gui-to-pyfs.md`, Geometry and mesh.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Geometry routes and remaining gaps

- Import CAD and mesh it (`IMPORT_CAD`, `CONVERT_CAD_TO_MESH`).
- CCS import is supported through the geometry route. See the website's CCS geometry page for its inputs and supported operations; do not treat every CAD construction command as the same capability.
- Wrap or unite meshes into one closed surface (`WRAPPER_EXECUTE`, `BOOLEAN_UNITE_MESH`).
- Repair a surface: delete, combine, invert, cut by a plane, fill holes (`SURFACE_DELETE`, `SURFACE_COMBINE`, `SURFACE_INVERT`, `SURFACE_CUT_BY_PLANE`, `SURFACE_AUTO_HOLE_FILL`, `DELETE_DEGENERATE_FACES`).
- Copy a surface around an axis or along a line (`SURFACE_CIRCULAR_COPY_PASTE`, `SURFACE_LINEAR_COPY_PASTE`).
- Export the surface mesh (`EXPORT_SURFACE_MESH`).

***Takeaway*** *A step that is more than a few lines, such as a geometry built from CAD, is done once in the GUI and saved as a* `.fsm`*: GUI once, script everything after.*

Source: `docs/gui-to-pyfs.md`, Geometry and mesh.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 2

# Boundary conditions

# Boundary conditions: what a key takes (1/3)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Mark the trailing edges from a file of edge points | sidecar table `[trailing_edges]` with `file`, and `type` and `tolerance` where not the default | 26.124 |
| Detect the trailing edges over the whole mesh | sidecar table `[trailing_edges]` with `detect = "auto"` | 26.101 to 26.122 |
| Detect the trailing edges of the surfaces you name, above a sweep angle | sidecar table `[trailing_edges]` with `detect = { surfaces = [...], sweep_angle = ... }` | none |
| Mark the wake-termination nodes, over the whole mesh or by surface | sidecar table `[wake_termination]` | none |
| Make the boundaries you name base regions | row key `BASE_REGIONS`, or pproc key `base_regions`, naming the base and not the body that carries it | none |
| Detect the base regions over the whole mesh | sidecar table `[base_regions]` with `detect = "auto"` | none |
| Apply or skip the trailing edges, wake-termination nodes and base regions the geometry declares | setup keys `apply_trailing_edges`, `apply_wake_termination` and `apply_base_regions`; `false` skips the redefinition and never clears saved state | none |

Source: `docs/gui-to-pyfs.md`, Boundary conditions.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Boundary conditions: what a key takes (2/3)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Change one trailing edge's type: relaxed, jet outflow, vortex shedding | setup key `trailing_edge_types`, each trailing-edge index with its type | 26.121 to 26.124 |
| Mark wake-termination nodes by hand, or stop a trailing edge's wake | setup keys `mark_wake_termination_nodes` and `disabled_wake_trailing_edges` | 26.121 to 26.124, except one |
| Shed wakes from the leading edges of a surface | setup key `leading_edge_wake_boundaries` | none |
| Create, remesh, delete or bend a base region, set its pressure, select its faces or mark its trailing or outflow edges | setup key `base_region_operations`, applied in the written order, and setup key `base_region_bending_angle_deg` for the detection angle | none |
| Add an inlet or an outlet | the geometry's `[ports]` identifies the surfaces; setup table `[[ports]]` selects each one's role, and the matrix row supplies its velocity and profile variables. Creating ports on a saved simulation stays refused while its existing port indices are unknown | none |

Source: `docs/gui-to-pyfs.md`, Boundary conditions.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Boundary conditions: what a key takes (3/3)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Delete an existing inlet or outlet | setup keys `delete_inlets` and `delete_outlets`, port indices in the written order | none |
| Set the edge bluntness angles and the vertex merge tolerance | setup keys `geometric_edge_bluntness_angle_deg` and `vertex_merge_tolerance_m`; the trailing-edge bluntness angle through setup table `[[flags]]`, which gives a one-value command a word of your own, or setup table `[[raw]]` | none |

## The trailing edge by file is the default route

Detection is an angle criterion, and a twisted blade's trailing edge is not a crease. The file names exactly the edges; the run checks the solver's count of imported edges against it.

Source: `docs/gui-to-pyfs.md`, **Boundary conditions**.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 3

# Frames, motion and actuators

# Frames, motion and actuators: what a key takes (1/2)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Create a coordinate system | reference table `[[frames]]` | 26.120 to 26.124 |
| Create a rotor's hub and blade frames | reference block `kind = "rotor"`: its hub, `axis` and `families_blades` | 26.120 to 26.124 |
| Turn a rotor: a rotary motion about its axis, at its speed, moving its blades and frames | row key `RPM` or `ADVANCE_RATIO`; the rotor's reference block gives its hub, axis, hand (`rpm_sign`) and blades, and a row stating `RPM_SIGN` is refused | none |
| Hold the blades still and turn the free stream about the shaft | `WORKFLOW qsteady_rotor`: a sector where the row states periodic symmetry, else the whole wheel; `PASSAGE_POSITIONS` for a wheel at incidence or in a custom inflow | none |
| Turn several rotors in one run | row keys `MOTIONS` and `CLOCK_MOTION`: one record per rotor, and the rotor whose turn sets the time step | none |

Source: `docs/gui-to-pyfs.md`, Frames, motion and actuators.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Frames, motion and actuators: what a key takes (2/2)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Choose the direction a rotor's relaxed wake sheds | row key `ROTOR_SHEDDING`, refused whatever its value: no workflow command applies the direction, so a row stating it is refused rather than run with a wake it did not ask for | none |
| Turn the free stream: a roll, pitch or yaw rate | row keys `roll_rate`, `pitch_rate` and `yaw_rate` in the `FLIGHT_CONDITION` cell; reference table `[body_axes]` | none |
| Create an actuator disc: its axis, radius, speed and swirl | reference block `kind = "actuator"`; row keys `ACTUATOR` and `ACTUATOR_RPM` | 26.121 to 26.124, except one |
| Load the disc by its thrust | row key `ACTUATOR_THRUST` | none |
| Load the disc by a radial profile | row key `PROFILE`, a file of `inputs/profiles/` | none |
| Rename or delete a disc | setup key `actuator_operations`, each naming one disc by its exact current name | 26.120 to 26.124 |

Source: `docs/gui-to-pyfs.md`, Frames, motion and actuators.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Frames, motion and actuators: not yet

- The actuator wake type is available through the actuator declaration. Disabling a disc uses the raw command route where no workspace key is documented.
- Duplicate, mirror or delete a coordinate system (`DUPLICATE_COORDINATE_SYSTEM`, `MIRROR_COORDINATE_SYSTEM`, `DELETE_COORDINATE_SYSTEM`).
- A motion of six degrees of freedom, with its mass, forces and gravity (`CREATE_NEW_MOTION_6DOF`, `SET_MOTION_MASS_PROPERTIES`, `CREATE_NEW_6DOF_EXTERNAL_FORCE`, `SET_MOTION_GRAVITY`).
- A prescribed motion: a velocity, an acceleration, or a table in time (`CREATE_NEW_MOTION_EUCLIDEAN`, `SET_MOTION_VELOCITY`, `SET_MOTION_ACCELERATION`, `CREATE_NEW_MOTION_CUSTOM`, `SET_MOTION_CUSTOM_TABLE`).
- Start a motion after the run has started (`SET_MOTION_START_TIME`).

## The rates' sense is measured (26.124)

Roll and yaw take the sign of their body axis: `roll_rate:40` writes `ROTATION <frame> X -6.667` [2].

Source: `docs/gui-to-pyfs.md`, Frames, motion and actuators.

Part 4

# Flight conditions and solver

# Flight conditions and solver: what a key takes (1/6)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Set the free stream: its speed, angle of attack and sideslip | row keys `ALPHA` and `BETA` in the `FLIGHT_CONDITION` cell, whose `MACH` or `TASmps` gives the speed; row key `VELOCITY` from Python alone | 26.120 to 26.124, except one |
| Set a custom free stream: import a velocity field over the YZ plane from a file | row key `FREESTREAM`, the stem of a file of `inputs/freestreams/`, a `.txt` in the STRUCTURED form or a `.dat` in the UNSTRUCTURED form, in m and m/s; the field is the flow's direction, so the row's angle of attack and sideslip are 0 | none |
| Sweep the angle of attack or the sideslip | `ALPHA:sweep` or `BETA:sweep` in the `FLIGHT_CONDITION` cell, with the `SWEEP_VALUES` column: one script; every point starts cold, clearing the solution before it, and row key `COLD_START` set to `false` keeps the earlier warm start from the last point | 26.120 to 26.124 |
| Set the fluid: density, pressure, temperature, viscosity, speed of sound | the `FLIGHT_CONDITION` cell, or the setup's `[flight_condition]` table | 26.101 to 26.124 |

Source: `docs/gui-to-pyfs.md`, Flight conditions and solver.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Flight conditions and solver: what a key takes (2/6)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Set the reference area, length and velocity | reference lengths; setup key `reference_velocity_m_per_s` | 26.101 to 26.124 |
| Give the free stream as a Mach number, set the reference Mach number or the speed of sound, or switch the reference velocity off | setup keys `freestream_input`, `reference_mach` and `disable_reference_velocity`; setup key `sonic_velocity_m_per_s` is refused on every build, since no build records `SONIC_VELOCITY` and the sound speed follows from temperature and specific-heat ratio | 26.120 to 26.124, except two |
| State the units of a custom free-stream file | row key `FREESTREAM_UNITS`: `SI` for metres and metres per second, which the package scales once into the simulation's unit, or `NATIVE` for a file already in it | none |
| Choose steady or unsteady, the time step and the number of steps | the `WORKFLOW` column: `steady`, `unsteady_rotor` or `qsteady_rotor`; row keys `DELTA_TIME` and `TIME_ITERATIONS`, or `DELTA_THETA` and `REVOLUTIONS` on a rotor row | 26.100 to 26.124 |
| Set the iterations and the convergence threshold | setup keys `iterations`, `convergence`, `forced_iterations` and `convergence_iterations` | 26.120 to 26.124 |

Source: `docs/gui-to-pyfs.md`, Flight conditions and solver.

# Flight conditions and solver: what a key takes (3/6)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Set the number of processors | row key `NCPUS`, or setup key `max_threads` | 26.120 to 26.124 |
| Choose the boundary layer and its coupling | setup keys `boundary_layer` and `viscous_coupling` | 26.120 to 26.124 |
| Switch the viscous coupling on at a time step of an unsteady run | setup key `unsteady_viscous_coupling_iteration`, refused on 26.124, which does not answer the command | none |
| Initialise the solver: the compressibility model, wall-collision avoidance, a mirror plane or periodic copies | setup keys `solver_model` and `wall_collision_avoidance`; row keys `SYMMETRY` and `PERIODIC_COPIES`; setup key `legacy_solver_model`, the older model command, refused on every build: only 25.000 documents `SET_SOLVER_MODEL` and no workflow writes that edition's `INITIALIZE_SOLVER` | 26.101 to 26.124, except one |
| Remove a saved initialization, mark proximal boundaries, set the automatic trailing-edge and wake-node physics, or delete transition trips | setup keys `remove_initialization`, `proximal_boundaries`, `physics_auto_trailing_edges`, `physics_auto_wake_nodes` and `delete_transition_trips` | 26.121 to 26.124, except two |

Source: `docs/gui-to-pyfs.md`, Flight conditions and solver.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Flight conditions and solver: what a key takes (4/6)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Advanced settings: far-field layers, the mesh-induced wake velocity, wake-on-wake induction, a further wake relaxation | setup keys `farfield_layers`, `mesh_induced_wake_velocity`, `wake_on_wake_induction` and `additional_wake_relaxation` | 26.120, 26.122, 26.123 |
| Advanced settings: the minimum pressure coefficient, stabilisation, unsteady pressure and Kutta condition, Reynolds-averaged drag | setup keys `minimum_cp`, `solver_stabilization`, `unsteady_pressure_and_kutta` and `reynolds_averaged_drag` | 26.120, 26.122, 26.123 |
| Advanced settings: wake relaxation, decay and agglomeration, and jet wakes | setup keys `wake_relaxation`, `wake_numerical_relaxation`, `wake_decay_constant_per_m`, `wake_streamwise_agglomeration`, `jet_wake_decay_normalized_length` and `jet_wake_filaments_grid_induction` | none |
| Advanced settings: where the wake ends | setup key `wake_termination_revolutions` or `wake_termination_steps` | none |

Source: `docs/gui-to-pyfs.md`, Flight conditions and solver.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Flight conditions and solver: what a key takes (5/6)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Advanced settings: the lift and separation models | setup keys `kutta_joukowski_lift`, `vorticity_lift_model`, `laminar_separation`, `adverse_gradient_boundary_layer`, `vortex_ring_normalization` and `axial_separation_families` | none |
| Advanced settings: rotor-induced velocities, field-grid refinement, the aeroelastic interpolation | setup keys `print_rotor_induced_velocities`, `rotor_induced_velocity_blending`, `adaptive_field_grid_refinement` and `aeroelastic_rbf_type` | none |
| Create or delete a separation model: bulk, airfoil, axial vortex, cylindrical, Stratford | setup keys `bulk_separation`, `airfoil_separation`, `axial_vortex_separation`, `cylindrical_bulk_separation`, `stratford_bulk_separation` and `delete_separations` | none |
| Set the Valarezo and crossflow separation criteria | setup keys `valarezo_criterion`, `valarezo_separation_boundaries`, `crossflow_separation_boundaries`, `crossflow_separation_diameter`, `crossflow_separation_mean_diameter` and `crossflow_separation_axisymmetric` | none |

Source: `docs/gui-to-pyfs.md`, Flight conditions and solver.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Flight conditions and solver: what a key takes (6/6)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Mark thin or viscous-excluded boundaries, or set a surface roughness | setup keys `thin_boundaries`, `viscous_excluded` and `surface_roughness` | 26.121 to 26.124, except two |
| Couple the blade's structure to the aerodynamics (FSI) | row key `FSI`, naming a structural configuration of `inputs/fsi/`, and fourteen row keys `FSI_<PROPERTY>_FACTOR`, each scaling one structural property; allowed on `steady`, `qsteady_rotor` sector and `unsteady` rows, refused on `unsteady_rotor` rows (Guide 8) | none |

## The custom field is the flow's direction

On 26.124 the angle of attack does not turn a custom field [2]: incidence goes into `vy` and `vz`, and a nonzero `ALPHA` or `BETA` beside `FREESTREAM` is refused at plan.

Source: `docs/gui-to-pyfs.md`, Flight conditions and solver.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 5

# The run

# The run: what a key takes (1/2)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Start the solver | `pyfs-matrix plan`, then `pyfs-matrix run` | 26.101 to 26.124 |
| Close FlightStream when the script ends | every run type | 26.100 to 26.124 |
| Choose the build | the `FS_BUILD` column | none |
| Run on this machine | `pyfs-matrix run`; on Linux with an HPC profile, `pyfs-matrix run --local` | none |
| Submit to a cluster | `pyfs-matrix run` on Linux with an HPC profile under `inputs/hpc/`, then `pyfs-matrix collect`; row keys `NCPUS` and `WALLTIME` | none |
| Stop an unsteady run before its wall clock does, so it can be continued | row key `WALLTIME`, with its unit; setup key `walltime_margin_s` | none |
| Continue an unsteady run where it stopped | row key `RESTART`: `{FINISH_PENDING}`, `{ADDITIONAL_ITERS=n}` or `{ADDITIONAL_REVS=n}` | 26.100 to 26.124 |

Source: `docs/gui-to-pyfs.md`, The run.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# The run: what a key takes (2/2)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Limit one point's wall clock on this machine | setup key `timeout_s`, which pyfs enforces around the solver process | none |
| Run only the points not run yet | `pyfs-matrix run --resume` | none |
| Redo a point already run | `pyfs-matrix run --force-rerun <point>` | none |
| Run several points of a polar, or several polars, in one solver session | `pyfs-matrix plan --polar-sweep` (one job per polar) or `--batch N` (N jobs), then `pyfs-matrix run` with the same option, then `pyfs-matrix collect`; the plan names each polar it leaves out | 26.124 |
| Skip or refuse a family the mesh does not carry | `--ignore-missing-families` of `plan` and `run`, which reaches each case as row key `IGNORE_MISSING_FAMILIES` | none |

***Takeaway*** *Every run step listed here has a workspace key.*

Source: `docs/gui-to-pyfs.md`, The run.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 6

# Basic post

# Basic post: what a key takes (1/5)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Export the loads table | pproc table `[exports]`: `loads`, always on | 26.101 to 26.124 |
| Save the solved simulation | pproc table `[exports]`: `simulation`, always on | 26.100 to 26.124 |
| Place the moment reference point and take the loads about it | reference table `[moment_point]`, which creates the frame `MRP` | 26.121 to 26.124 |
| Report the loads of the meshed half or sector, or of the whole | row key `SYMMETRY_LOADS`, or setup key `symmetry_loads` | 26.120, 26.122, 26.123 |
| Take the induced drag from the vorticity of the surfaces you name | setup key `vorticity_drag_families`; setup key `clear_vorticity_drag_boundaries` clears the selection after a steady solve | 26.121 to 26.124, except one |
| Loads in newtons rather than coefficients, inviscid loads, or boundaries left out of the loads (steady rows) | setup keys `load_units`, `inviscid_loads` and `analysis_families` | 26.120, 26.122, 26.123, except one |

Source: `docs/gui-to-pyfs.md`, Basic post.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Basic post: what a key takes (2/5)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Print more significant digits | setup key `significant_digits` | none |
| Export the surface solution as Tecplot, VTK or CSV, with the VTK variables you choose | pproc table `[exports]`: `tecplot` (on unless switched off), `vtk` and `csv` (off unless switched on); pproc key `vtk_variables`. The package preserves VTK cell values; pproc key `singularity_strength` (off unless set to true) adds same-step native nodal `Singularity_strength` after a unique geometry/topology match | 26.120 to 26.124, except one |
| Export the force distribution, panel by panel | pproc table `[exports]`: `force_distributions`, off unless switched on | 26.120 to 26.124 |
| Average the surface solution over the last steps of an unsteady run | pproc table `[time_averaging]`: the row exports its surface at every step of the window and `pyfs-matrix post` averages them, cell by cell | none |

Source: `docs/gui-to-pyfs.md`, Basic post.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Basic post: what a key takes (3/5)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Read boundary-layer integrals along configured cuts | pproc `[products]`: `boundary_layer_integrals = true`; original VTK cell values at recorded section cuts. A fixed-frame native comparison exists on 26.124; other conventions need their own evidence | none |
| Write sampled velocity fields as VTK, Tecplot or reusable inflow | pproc `[[probes]]`: `field_formats` and `reusable_inflow`; volume sections use the same probe sampling route. Evidence is specific to the export, the build and the unit; the guide states it | none |
| Cut surface sections and export their Cp and sectional loads | pproc table `[sections]` | 26.120 to 26.124, except one |
| Sample a flow-field plane and export velocity | pproc table `[volume_section]`; steady probes or unsteady/rotor fluid plots become package-written point fields; evidence is specific to the export, the build and the unit | 26.124, except one |

Source: `docs/gui-to-pyfs.md`, Basic post.

# Basic post: what a key takes (4/5)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Probe the flow at points and along lines (steady rows) | pproc table `[[probes]]` | 26.101 to 26.124, except one |
| Probe the flow through an unsteady run | pproc table `[[probes]]`, sampled as fluid plots | 26.124 |
| Plot the forces through an unsteady run | pproc table `[plots]` | 26.124 |
| Save residual, load and final section Cp plots | pproc table `[exports]`: `plot_residuals`, `plot_loads` and `plot_sections_cp`; section Cp is exported once after the march, never at every STEP | 26.124 |
| Export the solver log | pproc table `[exports]`: `log`; row key `EXPORT_LOG`, which the HPC profile's `[log]` table writes, and row key `LOG_OUTPUT` from Python alone | 26.100 to 26.124 |
| Export at every step of an unsteady run, from a step on | row keys `EXPORT_UNSTEADY_AFTER_ITER` and `EXPORT_UNSTEADY_AFTER_REV` | none |
| Sum the loads of groups of surfaces into polars | pproc tables `[groups]` and `[products]` | none |

Source: `docs/gui-to-pyfs.md`, Basic post.

# Basic post: what a key takes (5/5)

| In the GUI | In pyfs | DB evidence |
|---|---|---|
| Average an unsteady run over its last steps, per blade and at each azimuth | row keys `LAST_ITERS_AVG` or `LAST_REVS_AVG`, and `BLADES`; pproc tables `[phase_locked]`, `[equations]`, `[glossary]` and `[names]`, and pproc key `blade_pattern` | none |
| Post-process a saved simulation again, without solving it again | row key `ADDITIONAL_PPROC`, naming a second pproc of sections, sectional loads and surface exports, which `pyfs-matrix post --additional-pproc` extracts from each recorded point's final `.fsm` with no solve | 26.124, except one |
| Export a boundary-layer velocity profile | Separate pproc `[products]` flag `boundary_layer_velocity_profile`; currently refused by name before execution. Integral products are not a substitute. The command has no positive unattended-build evidence: it opens a modal dialog on 26.122 and is broken on 26.124 | none |
| Probe a surface property through an unsteady run | pproc table `[[surface_probes]]`, one per named point, parameter and frame | none |

Source: `docs/gui-to-pyfs.md`, Basic post.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# Basic post: additional routes and limits

- Trace streamlines, on the surface or through the field (`GENERATE_ALL_SURFACE_STREAMLINES`, `NEW_OFF_BODY_STREAMLINE`, `EXPORT_ALL_SURFACE_STREAMLINES`, `EXPORT_ALL_OFF_BODY_STREAMLINES`).
- Save a picture of the scene, coloured by a variable (`SET_SCENE_CONTOUR`, `SAVE_SCENE_AS_IMAGE`).
- Export the surface pressures as structural loads (`EXPORT_SOLVER_ANALYSIS_PLOAD_BDF`, verified 26.120 to 26.124).
- Import probe points through `points_file` in a pproc probe entry; the workflow emits `PROBE_POINTS_IMPORT`.
- Animate a surface through an unsteady run (`UNSTEADY_SOLVER_ANIMATION`).

*except one*: one command of the line has no probe run on those builds; the page names it.

Source: `docs/gui-to-pyfs.md`, **Basic post**.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 7

# The raw route

# A step with no key: three ways a row still takes it

**A `[[raw]]` line in the setup**: every row naming the setup emits it.

```
[[raw]]
command = "SET_TRAILING_EDGE_TYPE 3 RELAXED"
before = "init"
```

**A `RAW` record in the row's own cell**, for that row alone:

```
RAW: {COMMAND: SET_TRAILING_EDGE_TYPE 3 RELAXED
      / BEFORE: init}
```

**A custom flag**, for a command of one value; each row states `base_bending: 12.5`:

```
[[flags]]
name = "base_bending"
command = "SET_BASE_REGION_BENDING_ANGLE"
```

- Each line passes the checks a curated line passes: the command exists on the row's build, its arguments have the database's types, and it keeps the script's phase order. A line that fails is refused at `plan`, before any seat is spent.
- A raw line names what it acts on **by index**, a fact about one mesh file: keep it to the case it is written for.
- More than a few lines, such as a geometry from CAD: do it once in the GUI and save a `.fsm`. A `LEGACY` row runs a recipe of your own, in Python.

Source: `docs/gui-to-pyfs.md`, The raw route.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# The whole session, in the order we work

1. **Geometry**: a `.fsm` prepared once in the GUI, or an OBJ in metres with its sidecar: names, unit, trailing edge by file (26.124).
2. **The study**: one row per position in `matriz.fs`, citing the reference, the setup (`farfield_layers = 5`) and the pproc.
3. `pyfs-matrix plan`: no seat spent; every refusal happens here.
4. `pyfs-matrix run`, then `collect` on the cluster; `--resume` for a matrix that grew, `--force-rerun` for a row that was wrong.
5. `pyfs-matrix post`: products under `post/`, and `post.log` read first.
6. **More later, with no solve**: `ADDITIONAL_PPROC` and `post --additional-pproc`, from each point's saved `.fsm` (26.124).

***Takeaway*** *The GUI is for what has no key yet. Everything with a key is a line in a file that the next person can read, plan and run again.*

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# The physics behind the steps, and where to read it

| Step | The physics, and its origin |
|---|---|
| The surface mesh and its solve | a panel method: singularities on the surface panels, one boundary condition per panel [3, 4] |
| Trailing edges and wake nodes | the Kutta condition fixes the circulation at a sharp trailing edge, and the wake carries it downstream [4, 5] |
| Wake relaxation, wake termination | the wake is force-free: it is moved with the local velocity until it stops changing [4] |
| The compressibility model | the linearised subsonic correction of Prandtl and Glauert [6] |
| The boundary layer and its coupling | a viscous layer solved on the inviscid pressure and fed back to it [7, 8] |
| The separation models and criteria | turbulent separation by a pressure-gradient criterion, and a maximum-lift criterion [9, 10] |
| An actuator disc | the momentum theory of a propeller: a pressure jump and a swirl across a disc [11] |
| A body rate | the free stream seen from axes that turn with the aircraft [12] |

Copyright (c) 2026 Geovana Neves, CC BY 4.0

# References


[1] Geovana Neves, *pyflightstream documentation*, page "From the GUI to pyfs". `https://nevesgeovana.github.io/pyflightstream/latest/gui-to-pyfs/`

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

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

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


Copyright (c) 2026 Geovana Neves, CC BY 4.0

# References (continued)


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-42 H. Glauert, *The Elements of Aerofoil and Airscrew Theory*, 2nd ed., Cambridge University Press, Cambridge, 1947.

REF-43 B. Etkin, L. D. Reid, *Dynamics of Flight: Stability and Control*, 3rd ed., Wiley, New York, 1996.


Copyright (c) 2026 Geovana Neves, CC BY 4.0