Automation Workflow Brief: OpenFOAM Multi-Region CHT Sweep

Focus: Reproducible workflow, automation, KPI-based comparison, and traceability.

Objective and scope

This project demonstrates a deterministic automation pipeline for a multi-region conjugate heat transfer (CHT) OpenFOAM case so that multiple design variants can be executed and compared without manual case-by-case editing. The workflow is structured to support engineering benchmarking by generating comparable key performance indicators (KPIs), preserving traceability through logs, and producing a global summary suitable for reporting and design tradeoff discussion.

Sweep definition

A two-factor sweep was executed by varying the air inlet velocity and the porous region inlet temperature. The sweep uses $U_{\text{air}} = 5$ and 8 m/s and $T_{\text{porous,in}} = 400$ and 440 K, and it includes one repeatability run at $U = 8$ m/s and $T = 440$ K to confirm deterministic behavior of the pipeline.

Workflow pipeline

The sweep is launched from a single driver script that clones a clean base case, applies parameters, decomposes all regions, runs the solver in parallel, reconstructs results, and performs automated post-processing and global comparison. The pipeline is executed as the following sequence:

Base case → `setParams.sh` → `decomposePar -allRegions` → `mpirun chtMultiRegionSimpleFoam` → `reconstructPar -latestTime` → `plot_case_kpis.py` → `compare_cases.py (CSV + plots)`

The run strategy is hardware-safe and repeatable because each case is executed with 4 MPI ranks per case and the driver limits execution to 2 concurrent cases. Each stage writes a dedicated log file so that failures can be diagnosed quickly and the workflow remains auditable.

Implementation decisions

Parameters are applied using `foamDictionary`, which keeps the base case unchanged and makes every run reproducible. When velocity changes, inlet turbulence quantities are scaled consistently so that the sweep does not accidentally change turbulence assumptions, because turbulent kinetic energy is scaled with $k \propto U^2$ and dissipation is scaled with $\epsilon \propto U^3$, which preserves turbulence intensity and a consistent length-scale assumption across the velocity sweep. KPIs are extracted from OpenFOAM functionObject outputs in the `postProcessing/` directory, and derived metrics are computed consistently as $\Delta T_{\text{air}} = T_{\text{out}} - T_{\text{in}}$ and $\Delta p_{\text{air}} = p_{\text{in}} - p_{\text{out}}$ at the latest time.

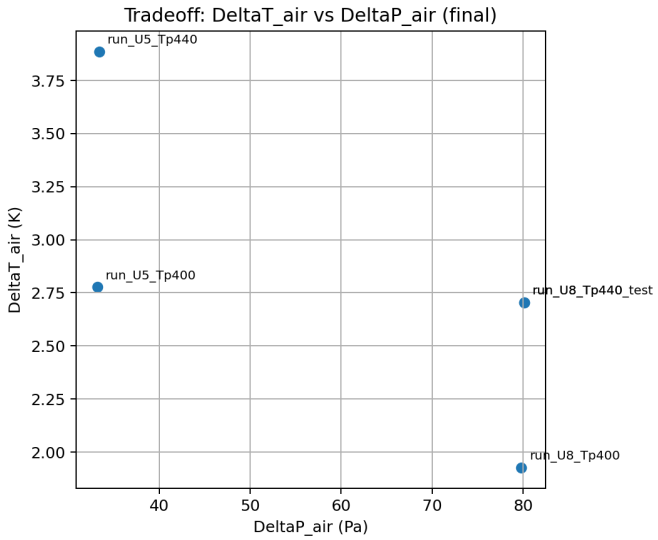
Final KPI snapshot

Case	U_{air} (m/s)	$T_{\text{porous,in}}$ (K)	$T_{\text{out,air}}$ (K)	ΔT_{air} (K)	Δp_{air} (Pa)
run_U5_Tp400	5	400	302.778	2.778	33.2
run_U5_Tp440	5	440	303.886	3.886	33.4
run_U8_Tp400	8	400	301.928	1.928	79.8
run_U8_Tp440	8	440	302.704	2.704	80.1
run_U8_Tp440_test	8	440	302.705	2.705	80.1

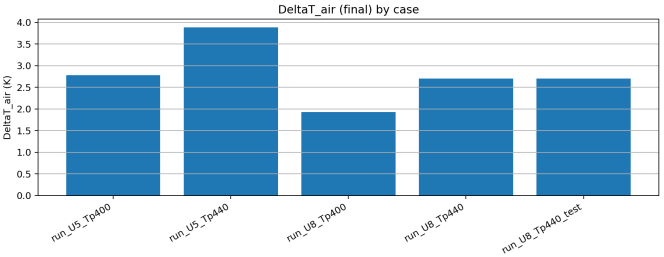
Key takeaways

Increasing air velocity from 5 to 8 m/s increases the pressure drop from about 33 Pa to about 80 Pa, which is the expected direction for flow losses, and it reduces ΔT_{air} because residence time decreases. Increasing porous inlet temperature from 400 K to 440 K increases ΔT_{air} at both velocities, which confirms that the thermal response is controllable and consistent across variants. The repeatability run `run_U8_Tp440_test` matches `run_U8_Tp440` extremely closely, which supports the claim that the pipeline is deterministic and that KPI extraction is stable.

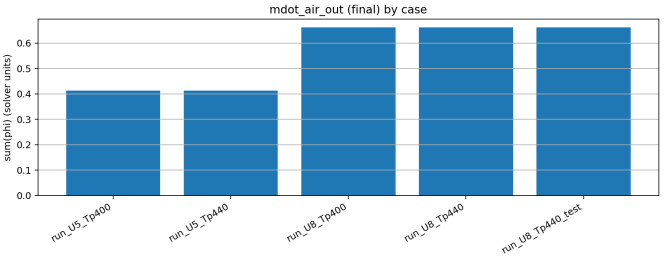
Result Plots



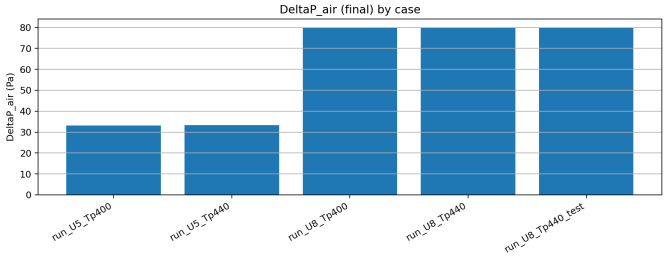
Tradeoff between thermal gain and pressure penalty.



Final temperature rise by case.



Outlet mass-flow proxy (sum of ϕ) by case.



Final pressure drop by case.

Next steps

This workflow can be hardened by adding automatic divergence detection via log parsing, automatic y^+ and mesh-quality reporting per case, and structured metadata export (for example JSON) so that the sweep can act as a regression test suite when workflow settings change.