WINGBENCH: BENCHMARKING HOW AI AGENTS SEARCH FOR BETTER DESIGNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Design search often uses cheap learned predictions to choose which candidates deserve expensive tests, but these tests can fail, especially on the unusual designs that optimization favors. We introduce WingBench, a multifidelity benchmark in which optimizers and AI agents design transonic wings with computational fluid dynamics (CFD) inside the search. They can query a learned aerodynamic surrogate and spend a fixed budget of sequential CFD evaluations, each returning a score or a failure type. WingBench grades the design each method recommends, reporting separately whether it receives a valid score and how good that score is. Using WingBench, we contribute a study of five baseline optimizers and a controlled comparison of three AI agent methods, which produced 6,629 CFD evaluation records. Feedback helps baseline optimizers deliver valid designs when it changes their later proposals, but the differential-evolution variant does so with lower aerodynamic quality. An open-ended research agent is competitive with the strongest baseline optimizers over 32 CFD queries, and in early search, our reflection and planning implementations do not consistently improve on it. We release the data, code, agent prompts, and analysis scripts for reproducibility.

1 INTRODUCTION

Designing an aircraft, material, drug, or energy system often means searching a large space of possibilities while having reliable evidence for only a small number of designs. High-fidelity numerical models and physical experiments can take hours or days, so researchers commonly use cheaper learned models to screen many candidates before committing scarce laboratory or computing resources. These approximations make search practical, but they are least reliable for the unusual and extreme designs that optimization tends to favor. The central decision is therefore not simply which candidate looks best, but which candidate should be tested next and which candidates are credible enough to recommend for laboratory or other costly validation.

AI agents could help make these choices by combining prior knowledge, computational tools, and feedback from experiments. Recent systems have used experimental results to guide further computational research (Lu et al., 2026), and others have proposed nanobody candidates and biomedical hypotheses that researchers then tested in the laboratory (Swanson et al., 2025; Gottweis et al., 2026). Whether this kind of reasoning leads to better design choices is less clear. In controlled comparisons, classical optimizers have outperformed language-model agents, although a hybrid of the two did better than either (Ferreira et al., 2026). These agents have also sometimes failed to respond to the results of their own experiments (Gupta et al., 2025). Work on language-model agents suggests giving them explicit structure for using feedback, such as written reflections that carry lessons into later attempts (Shinn et al., 2023) or plans that are revised after each new result (Liu et al., 2024). One of our central questions is whether explicit reflection or planning improves design search over open-ended research when expensive tests can fail.

We introduce WingBench, a multifidelity benchmark for studying how optimization methods and agents use cheap predictions and limited, failure-prone tests to find designs. Transonic wing design provides a concrete setting. A candidate wing is judged by its lift-to-drag ratio L/D under computational fluid dynamics (CFD), a high-fidelity numerical model. In WingBench, testing one candidate wing with CFD uses 30.4 processor-core hours on average. AeroTransformer (Yang et al., 2026a), a learned aerodynamic model, estimates the same quantities in about a second. However,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

AeroTransformer's predictions can still be optimistic for wings that fail to mesh or converge under CFD. Optimizers must learn both where L/D is high and where the high-fidelity process fails.

Existing benchmarks for aerodynamic design do not ask optimizers to learn this second skill during search. Some run CFD inside optimization from a given starting design, with gradients (Lyu et al., 2015; Felten et al., 2025) or without them (Plaban et al., 2026). Others, such as ShapeBench (Fazliani et al., 2026), search with learned surrogates and use CFD only to verify final designs. Yet optimization naturally pushes toward unusual designs, where the surrogate is least reliable and CFD is most likely to fail. WingBench therefore makes the search itself multifidelity, integrating CFD evaluations inside under a fixed budget and returning either a score or a recorded failure type to the optimizer. A WingBench episode is one independent optimization run. The benchmark allows up to 4,096 predictions from the surrogate and 32 sequential CFD evaluations per episode. After specific CFD evaluation checkpoints, the optimizer must recommend the design it would return to a user. A private grade then evaluates whether that recommendation receives a valid score and, if so, how good it is. We call these outcomes *delivery* and *conditional quality*. It separates a method that returns exceptional wings in half its runs from one that returns mediocre wings every time.

We use WingBench for two complementary studies. The first study establishes baseline performance with five optimizers. These include three CMA-ES variants that isolate different uses of CFD feedback, a feedback-guided differential-evolution method, and a Bayesian optimization baseline implemented with SMAC3 (Lindauer et al., 2022). The second study tests whether explicit reflection or planning improves on open-ended agentic research. We compared these three agent methods after four CFD queries and extended open-ended research to 32 queries to compare against the baseline optimizers. At 32 queries, open-ended research is competitive with the strongest baseline optimizers. At four queries, the three methods deliver similarly, while open-ended research leads in aerodynamic quality at nominal load and no method leads elsewhere. Neither reflection nor planning consistently improves on open-ended research across both outcomes in the tested implementations.

The paper makes four contributions.

1. **A multifidelity benchmark with CFD in the search.** WingBench combines a cheap learned surrogate with sequential CFD evaluations that can fail with a recorded failure type. It grades each recommendation privately and reports delivery and conditional quality separately.
2. **A baseline study.** Five baseline optimizers show that feedback raises delivery when it changes later proposals, and that the proposal generator still sets the quality of what is delivered.
3. **An optimization-agent study.** Three agent optimizers test whether explicit reflection or planning improves on open-ended research.
4. **Released data and code.** The release holds all 6,629 evaluation records with failure types, recommendation scores, benchmark and baseline code, prompts, and analysis scripts.

2 WingBench

2.1 Design space and objective

WingBench uses the wing parameterization from SuperWing (Yang et al., 2026b). Each wing is described by 38 input variables, including 20 class-shape-transformation coefficients that control the upper and lower airfoil profiles (Kulfan, 2008) and 18 variables for planform and spanwise shape. We standardize each coordinate with the released data and use the box $[-3.5, 3.5]^{38}$ as our design space. Ninety-three percent of dataset designs lie inside this box, while the remaining volume still permits substantial extrapolation. This is compact compared with the 153 degrees of freedom that Yang et al. (2026a) use for the Common Research Model wing (Vassberg et al., 2008). The objective is L/D at a target lift coefficient C_L. Because the required lift is the operating constraint, angle of attack is adjusted until computed lift is within 0.02 of the target. We use $C_L \in \{0.15, 0.41, 0.66\}$, the rounded lower quartile, median, and upper quartile of successful SuperWing conditions. We call these low, nominal, and high load. Each wing is evaluated at Mach 0.7853, 0.8195, and 0.8530, the lower quartile, median, and upper quartile of Mach number among successful SuperWing conditions, with Reynolds number 2×10^7. The score is the geometric mean of the three L/D values. A design delivers only if all three conditions return valid results, $L/D \in (0, 40]$, and angle of attack lies in

$[-10°, 25°]$. The upper limit of 40 leaves headroom above both the largest successful single-condition L/D in the dataset, 29.4, and the largest three-condition score observed in our studies, 34.0.

2.2 High-fidelity evaluator and surrogate

We use ATsurf-L, the large surface-flow variant of AeroTransformer (Yang et al., 2026a). It predicts surface quantities and integrates them to estimate lift and drag. In our three-worker CPU deployment, one surrogate evaluation scores one wing in about a second, including all three Mach conditions and the angle-of-attack adjustment needed to reach the required lift. The high-fidelity evaluator checks candidate wings using ADflow (Mader et al., 2020), the SuperWing mesh procedure, and common solver settings. Each CFD job uses 16 cores and 64 GB of memory on one AWS m7a.4xlarge machine. Meshing has a 30-minute limit and each of three flow conditions a two-hour limit, both in wall-clock time. A high-fidelity evaluation can nevertheless fail to produce a usable score. The geometry may be rejected as unmeshable, meshing may stop without a valid mesh, or the flow solver may stop before satisfying the convergence and lift requirements. We call these outcomes nondeliveries and record their types rather than assign them invented objective values. Across both studies, 4,009 of the 6,629 evaluations delivered no score. The mesh generator rejected 557 geometries and stopped without a mesh in 2,173 evaluations, a flow condition reached its two-hour limit in 782, a solver process stopped with an error in 493 (24 had both), and 28 ended in other ways. The 6,576 timed evaluations used 200,019 core-hours, 30.4 per evaluation on average (Appendix E). The failure types give an optimizer useful evidence about the proposed geometry and where the evaluation process encounters difficulties. Figure 1 shows visible shape differences between some meshable and unmeshable wings.

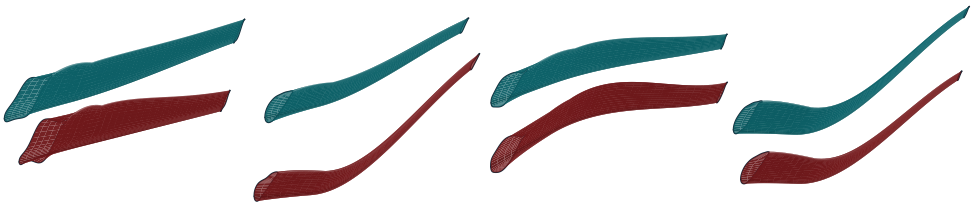

Figure 1: Teal wings (top) yielded CFD scores, and red wings (bottom) were rejected as unmeshable.

The surrogate does not solve this problem by itself. It was trained on converged data and frequently predicts strong scores for regions in which the CFD pipeline is unreliable. Furthermore, standard failure classifiers trained on source records reach AUC 0.65–0.71 in cross-validation but only 0.41–0.75 on optimizer proposals, depending on task and model. Appendix C gives details.

2.3 Episodes, budgets, and private recommendations

An episode is an independent search, and information is not shared across episodes. An optimizer may spend at most 4,096 cheap evaluations and 32 sequential online CFD evaluations. It submits one design, observes the accepted result, and then chooses again. After selected CFD query counts, the optimizer nominates one design. Its CFD performance is assessed outside the search, and the assessment provides no additional feedback to the optimizer. The benchmark supports checkpoints $q \in \{0, 1, 2, 4, 8, 16, 32\}$. We report three quantities. Recommendation delivery is the fraction of private recommendations with a valid score. Conditional quality is their L/D among delivered recommendations. Online delivery is the fraction of requested experiments that return a score. Appendix F states the problem in symbols and Appendix G gives the full rules.

3 Related work

Choosing which candidates to evaluate matters when high-fidelity simulations are costly. Adjoint methods make gradient-based shape optimization practical (Martins, 2022), and benchmarks such as the Common Research Model wing case (Lyu et al., 2015) and the EngiBench airfoil problem (Felten et al., 2025) run CFD inside that optimization. They start from a given design and report a failed simulation as an error. WingBench instead treats CFD as a black box with a fixed budget of 32 evaluations that can fail. Surrogate-assisted search explores more designs with cheap models (Forrester & Keane, 2009; Jin, 2011), and multifidelity Bayesian optimization adds lower-cost

approximations (Kandasamy et al., 2017; Takeno et al., 2020), even when simulations can fail (He et al., 2021). Work on hidden constraints, which a simulation reveals only by failing (Le Digabel & Wild, 2024), learns where the objective can be computed (Lee et al., 2011; Gelbart et al., 2014; Sacher et al., 2018; Bachoc et al., 2020; Chakrabarty et al., 2021), including in CFD and aircraft design (Tran et al., 2019; He et al., 2020; Tfaily et al., 2024), and benchmark problems with such failures exist outside CFD (Stenger & Abel, 2022; Bussemaker et al., 2024). ShapeBench (Fazliani et al., 2026) is the closest work to ours. It uses the same SuperWing parameterization and also runs LLM-driven optimizers, one with a persistent reflection memory, but they search with surrogates and CFD, if available, only verifies the final designs. WingBench instead integrates CFD evaluations directly into the search, returning each evaluation's score or failure type to the optimizer. Furthermore, it privately grades the design each optimizer explicitly recommends, not just the best design it found during search. Agent benchmarks make similar distinctions, reporting the best submission apart from the best attempt (Nathani et al., 2025; Chan et al., 2025; Huang et al., 2024b) or valid execution apart from success (Chen et al., 2025; Somasekharan et al., 2026), but checking an attempt there is cheap and can be repeated within a run. A WingBench grade costs hours of CFD and can fail, so each recommendation is graded once, privately, at a fixed budget. Together, these let WingBench test whether an optimizer turns limited, failure-prone CFD feedback into a design worth recommending.

The optimizer can also be an AI agent. Systems that conduct computational research (Lu et al., 2026; Yamada et al., 2025; Schmidgall et al., 2025), run laboratory chemistry (Boiko et al., 2023; Bran et al., 2024), discover algorithms (Romera-Paredes et al., 2024; Novikov et al., 2025), or propose scientific designs (Swanson et al., 2025) show how language models can participate in research beyond predicting an outcome. Language-model agents can also run CFD, using solver errors to repair their simulation setups (Yue et al., 2026), and language-model optimizers can estimate the chance that an evaluation fails (Meindl et al., 2026). For design search, this raises the question of how an agent should organize its reasoning. Reflexion addresses this by turning feedback into written reflections that are retained to guide subsequent attempts (Shinn et al., 2023). Reason for Future, Act for Now (RAFA) emphasizes planning ahead from the interaction history, taking the first planned action, and revising the plan after observing its outcome (Liu et al., 2024). Evidence on when such structure helps is mixed. Written self-critique improves results mainly when reliable external feedback is available (Huang et al., 2024a; Kamoi et al., 2024), and searching over candidate plans pays off only when those plans can be judged accurately (Chen et al., 2024). Most of this evidence comes from games, question answering, mathematics, and coding, where feedback is cheap and an attempt is easy to repeat. It is less clear whether such structure helps when each piece of feedback is a costly simulation that may fail. WingBench creates such a setting by combining a learned aerodynamic surrogate with sequential CFD evaluations, giving multifidelity feedback on an engineering task. We test whether reflection or planning improves on open-ended research.

4 METHODS

We study two ways of organizing sequential design search. The baseline optimizers use numerical update rules, while the agents can choose their own analyses and code.

4.1 BASELINE OPTIMIZERS

The five baseline methods all receive 32 online CFD evaluations. Four also use 4,096 surrogate evaluations. Table 1 summarizes the controlled differences. At each checkpoint, every method recommends its best delivered online design once it has one. Before that, it recommends an untested design chosen by its own selection rule. The four surrogate-assisted methods also start every episode from the same 62 CFD results recorded during benchmark development. Those that learn from feedback use them as prior data, none of the 62 is tested again, and the agents did not receive them.

The first three use CMA-ES (Hansen & Ostermeier, 2001; Hansen, 2016) as their proposal generator. They share a 64-design archive policy. The *no-feedback* method completes the surrogate search and evaluates the 32 best distinct archive designs in descending surrogate-score order. The *selection-only* method fits simple delivery and value-correction models to choose which archived design to evaluate next, but cannot generate a new design. The *feedback-guided CMA-ES* method alternates eight surrogate generations with one CFD evaluation. The resulting feedback updates beliefs that guide both selection and the future proposal distribution. The *feedback-guided differential evolution* method

Table 1: Baseline optimizers.

method	surrogate calls	CFD calls	feedback picks next	changes proposals	generator
CMA-ES, no feedback	4,096	32	no	no	CMA-ES
CMA-ES, selection only	4,096	32	yes	no	CMA-ES
Feedback-guided CMA-ES	4,096	32	yes	yes	CMA-ES
Feedback-guided DE	4,096	32	yes	yes	DE
SMAC3	0	32	yes	yes	GP BO

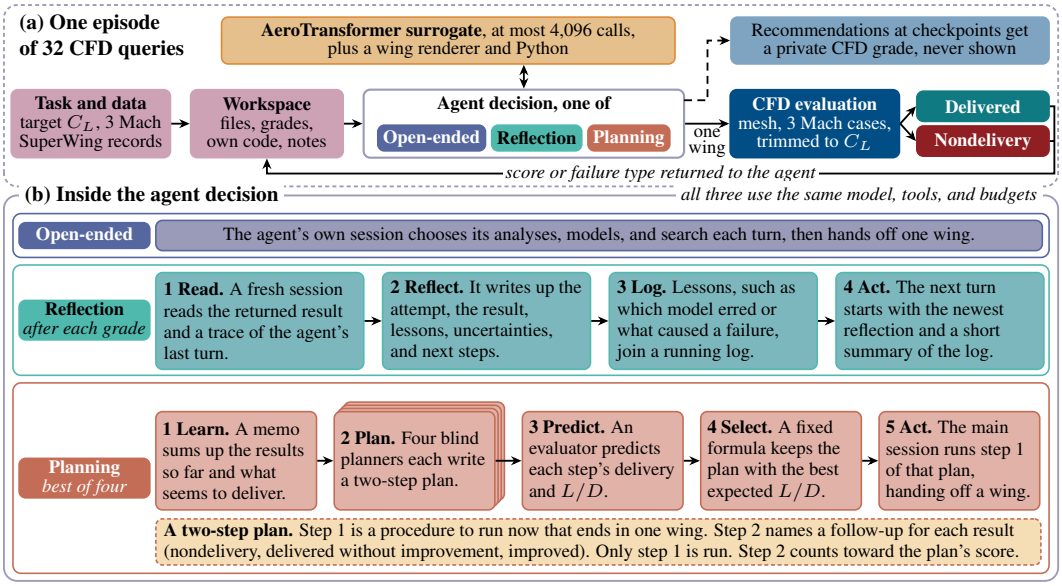

Figure 2: Agent episodes and decision procedures. (a) In each episode, the agent works in its own workspace, may query the AeroTransformer surrogate, and hands off one wing for each CFD query. The score or failure type is returned, and recommendations nominated at checkpoints are graded without feedback. (b) The three methods differ only in how each decision is made.

uses differential evolution (Storn & Price, 1997). We used DE/rand/1/bin with mutation factor 0.8 and crossover probability 0.9. It otherwise shares the feedback design. The *SMAC3* baseline (Lindauer et al., 2022) uses version 2.4.0 for CFD-only Gaussian-process Bayesian optimization with 8 Sobol initial points and 24 adaptive queries, with a common penalty for each nondelivery. Appendix F gives the belief model's equations and every setting.

4.2 Optimization agents

The agent study compares three methods under the same task specification, tools, budgets, isolated workspace, DeepSeek V4.1 Flash model, and maximum reasoning-effort setting. The agent can inspect the supplied SuperWing data, visualize wing geometries, write and run analysis code, fit predictive models to the data available within its episode, and run numerical optimization to propose new designs. Figure 2 outlines the steps that every agent episode follows.

Open-ended research. The agent receives the optimization task and available tools without an imposed planning or reflection template. It may choose its own analysis, code, and decision process.

Explicit reflection. Following the verbal-feedback approach of Reflexion (Shinn et al., 2023), the same agent writes a reflection after every newly returned online grade. The next decision receives the newest full reflection and a bounded cumulative summary of at most 4,000 characters. Reflection supplements the numerical history rather than replacing it.

Best-of-four contingent planning. A learner first summarizes the observed environment. Four blind, independent planners are then asked to produce complete two-step contingent plans. A common value estimator scores the plans by expected best delivered L/D, the first action of the selected plan is executed, and the process replans after the returned grade. At most three planning cycles are used

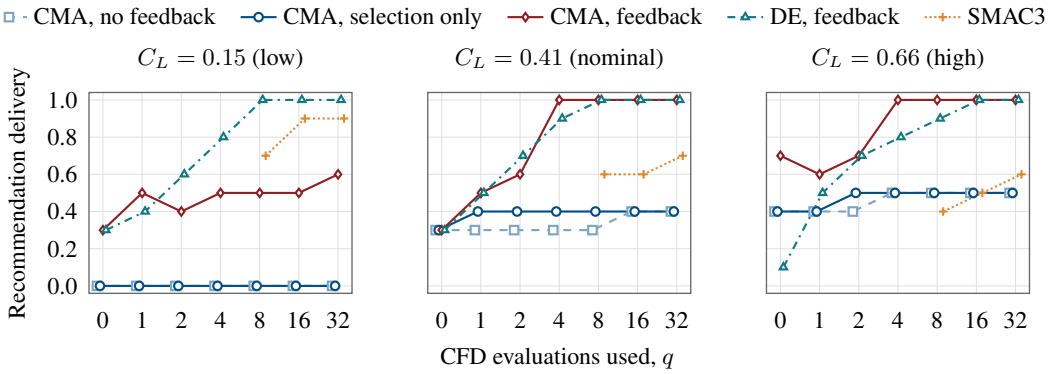

Figure 3: Recommendation delivery for the five baseline optimizers. A method keeps its best delivered design once it has one. Before that, it recommends untested designs, so a curve can fall. SMAC3 is shown from query 8 onward, omitting the early checkpoints within its Sobol initialization.

per decision. We call this RAFA-motivated because it learns, plans, acts once, and replans, but its finite horizon and discrete best-of-four selection differ from RAFA (Liu et al., 2024). Appendix M reproduces every prompt and task document.

5 Experimental design and analysis

The *baseline study* contains 150 episodes spanning five methods, three tasks, and ten seeds. Its 4,800 online evaluations and 493 additional recommendation grades total 5,293 CFD evaluations. For delivery, seed-level method differences are averaged over the three tasks. For conditional quality, each seed contributes an average over the tasks where both methods deliver. Seeds without such a pair are excluded from that contrast. We used exact sign-flip tests and 10,000 seed-block bootstrap resamples. Holm correction covers the six planned delivery and paired-quality tests.

The *agent comparison* contains 90 episodes spanning three methods and three tasks, with the same ten seed labels for every method. We compared their recommendations after one, two, and four CFD queries. These runs contain 360 online evaluations and 136 additional private recommendation evaluations. We continued the open-ended agent's 30 episodes to 32 queries for comparison with the baselines, choosing it for its highest conditional quality on two of the three tasks after four queries. The extension added 840 online evaluations, bringing both studies to 6,629 evaluation records. For the agent comparison, we report delivery and conditional mean L/D for each task. At query 4, we estimated differences between open-ended research and each alternative. Because a seed label does not pair agent episodes, the two methods' episodes are resampled independently, 10,000 times, and quality differences compare delivered recommendations only. Delivery rates carry exact binomial intervals and agent delivery differences Newcombe score intervals (Appendix F). Appendix K lists every result table, Appendix H the released records, and Appendix I the protocol.

6 Results and discussion

6.1 Baseline optimizer performance

At 32 queries, feedback-guided CMA-ES combines reliable delivery with strong aerodynamic quality at nominal and high load. It delivers in all ten episodes on both tasks, with conditional mean L/D of 33.04 and 31.64. The archive-only CMA-ES methods find slightly stronger wings among their successful runs, averaging about 33.2 and 32.1, but each delivers in only four nominal-load and five high-load episodes (Figure 3). Both archive-only methods fail to deliver at low load.

The three CMA-ES variants suggest that feedback is most useful when it changes future proposals. Reordering an existing archive cannot escape shared meshing difficulties or surrogate errors, whereas feedback-guided CMA-ES can redirect its search. Its higher online delivery rates show that adaptation changes which designs are tested, not just which recommendation is retained (Table 2). The variants also allocate cheap evaluations differently, completing them up front or interleaving them with CFD.

Table 2: Online CFD delivery rate in the baseline study.

method	$C_L = 0.15$	$C_L = 0.41$	$C_L = 0.66$
CMA-ES, no feedback	0.00	0.26	0.37
CMA-ES, selection only	0.00	0.27	0.35
Feedback-guided CMA-ES	0.04	0.77	0.77
Feedback-guided DE	0.29	0.62	0.62
SMAC3	0.36	0.28	0.15

Table 3: Open-ended research over a longer search. Best L/D is the mean episode maximum over delivered online designs (nine low-load episodes have one at query 4, ten in every other cell).

C_L	online delivery through query 32	best L/D at 4	at 16	at 32
0.15	0.62	17.18	17.87	18.23
0.41	0.70	31.09	32.71	33.04
0.66	0.62	27.83	29.81	30.24

Their comparison therefore tests whole search strategies, not failure modeling alone. Feedback-guided DE also delivers in every nominal- and high-load episode, but mean quality is lower at 31.09 and 27.05. At low load, DE delivers in all ten episodes with mean L/D of 17.52, compared with six deliveries and 16.76 for feedback-guided CMA-ES. Since both use the same feedback model, the generator remains important. DE may preserve useful alternatives where CMA-ES concentrates too narrowly, while CMA-ES may refine productive regions more effectively on other tasks. SMAC3 delivers in 9, 7, and 6 episodes at low, nominal, and high load, but its conditional mean L/D is much lower at 10.56, 17.00, and 12.82. Unlike the other baselines, it learns from expensive feedback without a pretrained aerodynamic model. With only 32 evaluations in 38 dimensions, with many failing, its resources are limited. This is a plausible explanation for its low quality.

The final-recommendation contrasts quantify the reliability-quality tradeoff. Relative to both archive-only CMA-ES variants, feedback-guided CMA-ES improves delivery by 0.57 averaged across tasks (95% bootstrap interval [0.37, 0.77], Holm $p = 0.039$). Its paired conditional-quality differences are about -0.35, with bootstrap intervals of $[-0.79, -0.01]$ and $[-0.80, -0.01]$, but seven pairs leave the planned sign-flip tests underpowered (Holm $p = 0.28$). Relative to the selection-only method, feedback-guided DE improves delivery by 0.70 [0.57, 0.83] (Holm $p = 0.012$) but loses 3.50 L/D [2.28, 4.94] among paired delivered recommendations (Holm $p = 0.047$). At nominal load, feedback-guided CMA-ES largely resolves the tradeoff with complete delivery at similar quality. This leaves less room for improvement at query 32, although differences at other loads rule out uniform saturation. At query 4, the nominal-load archive-only methods already average about 33.2 but deliver in three or four episodes, while feedback-guided CMA-ES delivers in all ten at 28.73.

6.2 Agent performance over 32 queries

At 32 queries, open-ended research is competitive with the strongest baselines but does not consistently outperform them. For the baselines, recommendation outcomes at query 32 match their best delivered online designs. Every delivered recommendation is that design, and no episode without an online delivery delivers its recommendation. We compare open-ended research on the same basis. Its low-load mean of 18.23 exceeds DE's 17.52, while its nominal-load mean of 33.04 is essentially identical to feedback-guided CMA-ES (33.038 and 33.040). At high load, its 30.24 exceeds DE's 27.05 but remains below CMA-ES's 31.64. All ten agent episodes find a delivered online design on every task. The extension requested no further recommendations, so we also score each episode's last nomination as its standing recommendation. It is a delivered wing in 28 of 30 episodes, nine each at low and high load and ten at nominal load, with mean L/D of 18.21, 33.03, and 30.43, and the best delivered wing in 20 (Appendix B). Table 3 and Figure 4 show how this performance develops. The scores are episode maxima among delivered online designs, not private recommendation assessments. Ten of the 960 online evaluations failed in the grading service before CFD began and were charged to the agent against the released rules (Appendix B). Regraded after the study, the high-load wing was unmeshable, three of the other four delivered, one above its episode's best online score, and the fourth reached the time limit at one of its three conditions. Appendix J shows the best delivered wing of every method and task, and Appendix L the outcome composition of the online evaluations.

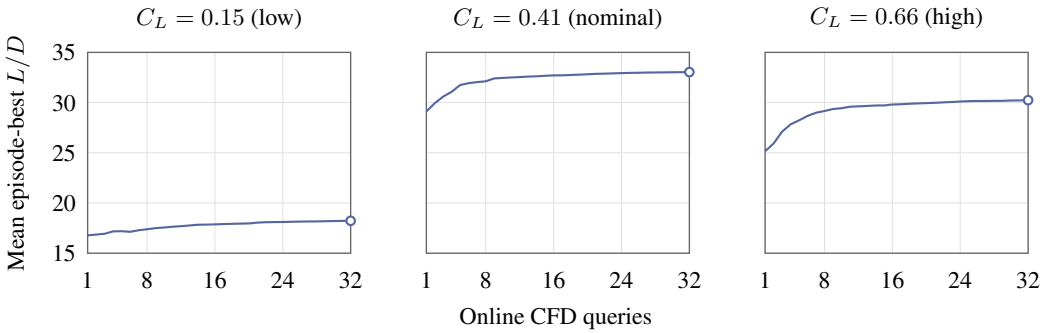

Figure 4: Mean episode-best L/D during the open-ended search. Each curve conditions on episodes with at least one delivered design. A retained best design cannot deteriorate within an episode, but the population contributing to the mean can change when another episode first delivers.

Table 4: Agent recommendations after four CFD queries. Mean L/D includes only delivered recommendations, so the quality columns can describe different subsets of episodes.

method	$C_L = 0.15$		$C_L = 0.41$		$C_L = 0.66$	
	delivered	mean L/D	delivered	mean L/D	delivered	mean L/D
Open-ended research	9/10	17.15	10/10	31.09	10/10	27.83
Planning	10/10	17.41	9/10	29.31	9/10	27.59
Reflection	10/10	17.33	10/10	29.83	10/10	26.79

Better designs need not come with higher delivery rates. At high load, online delivery falls from 0.85 over the first four queries to 0.625 over the first 16 queries, even as mean episode-best L/D improves. This is consistent with exploring less reliable regions while retaining strong designs.

The largest three-condition score across the studies is 34.0, compared with a maximum single-condition L/D of 29.4 in the SuperWing source records. This exceeds the dataset's recorded range but is not a matched estimate of improvement over its best wing. SuperWing samples operating conditions, whereas WingBench trims lift and aggregates three Mach conditions.

6.3 Agent recommendations at four queries

The early checkpoint allows all three agent methods to be compared. Reflection delivers 30 of 30 recommendations, open-ended research 29, and planning 28 (Table 4). Planning has the highest low-load conditional mean L/D, while open-ended research leads at nominal and high load. No method leads across all tasks and both outcomes (Figure 5).

The quality comparisons reinforce this task dependence. At nominal load, the delivered recommendations of open-ended research exceed planning's by 1.77 L/D with a descriptive 95% bootstrap interval of [0.32, 3.21]. At low load, the difference is -0.26 with interval $[-0.85, 0.31]$, so planning's higher mean is not distinguishable from chance. The intervals against reflection include zero on all three tasks. Full comparisons are given in Appendix B. The whole agent study used 71.5 million input and 29.1 million output tokens and cost about \$118, under \$0.25 per CFD query for every method, with open-ended research averaging 9 minutes of model time per query, planning 25, reflection 31. Relative to the baselines at this checkpoint, open-ended research attains higher nominal- and high-load quality than feedback-guided CMA-ES while matching its complete delivery. These are intermediate results within the 32-query task, not a test of strategies built to stop after four evaluations.

6.4 Agent strategies and failure modes

The traces show agents fitting supplementary models, writing geometry checks, and developing numerical search procedures across the three methods and tasks. A nominal-load reflection episode combined a local aerodynamic model and condition-success classifier with CMA-ES. Its CFD quality rose from 28.79 to 31.14 across four delivered evaluations, but the supplementary aerodynamic model became about 12% optimistic on later candidates while AeroTransformer stayed within about half a percent. Extra modeling could also encourage excessive caution. A high-load reflection episode initially rejected a wing family whose local ensemble prediction was 22.91. When tested later, the

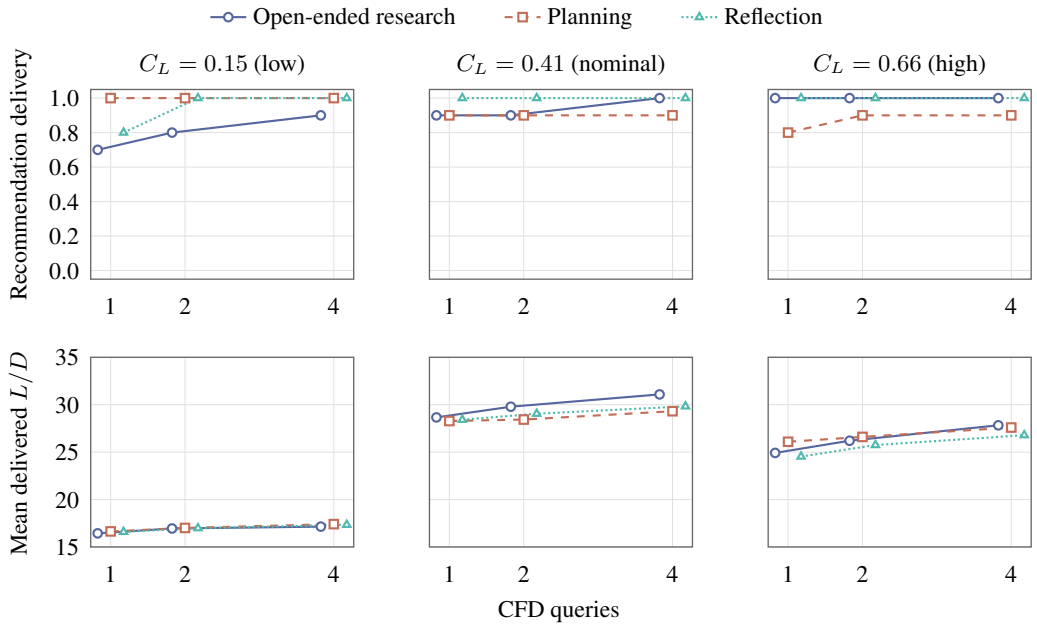

Figure 5: Agent recommendation delivery and conditional mean L/D after one, two, and four CFD queries. Quality is averaged over the delivered subset at each checkpoint, which can change.

wing delivered at 24.89, close to AeroTransformer's 24.85. These cases show why constructing another predictor does not necessarily improve the search. Agents also used failure diagnostics to change candidate selection. After two meshing failures, a high-load planning executor reconstructed wing surfaces and developed a surface-triangle quality measure. Using that measure produced a fourth wing that delivered at 28.55, slightly above the first wing's 28.52. This was a substantive attempt to recover delivery, but not a large improvement.

Some procedures resemble the baselines, such as a nominal-load open-ended episode that ran CMA-ES with corrections for surrogate errors. Agents can construct and revise these methods, but added flexibility helps only when it changes which experiments are worth performing. With at most 32 CFD evaluations and fewer usable scores, validating new models or rules is difficult. This is one reason elaborate analyses need not beat a strong numerical optimizer, even at the early checkpoint.

7 Conclusion

WingBench turns a hard engineering question into a measurable one. Every optimizer, numerical or agentic, faces the same 32 costly CFD evaluations that can fail, and the benchmark grades the wing a method would hand to a user, not on the best result it happened to see. The baseline study shows that this framing rewards a specific skill. Feedback that changes later proposals raised delivery by 0.57 across tasks at a small cost in quality, while reranking a fixed archive could not escape shared failures. The agent study shows that a general research agent, working from documents and tools rather than a hand-built update rule, matches the quality of the strongest baseline at nominal load, delivers in nine low-load episodes where that baseline delivered in six, and beats the most reliable baseline's quality at low and high load, for under a quarter of a dollar of model usage per CFD query. Everything needed to repeat or extend these results is released, the 6,629 graded evaluations with their failure types, the rules, the runner and verifier, the prompts, and the analysis. The two endpoints make the reliability of a method visible next to its ambition, and the failure types turn every nondelivery into evidence that an optimizer can use. The benchmark side is complete as well. A new optimizer plugs into the same episode machine, its trace is replayed independently, and its result lands next to the six reference methods in the closed or the open variant of three tracks. The benchmark rewards methods that treat a failed evaluation as information, and the release gives them a fixed place to prove it. The reference results set a bar that a stronger method should clear on all three tasks at once, and every record needed to check such a claim is in the release, down to the single evaluation.

AI USE STATEMENT

Generative AI tools assisted with writing and editing the manuscript, with implementing and debugging code for the analysis, the experiment pipeline, and the agent harness, and with organizing the CFD records. The authors conceived the research, designed the benchmark and the experiments, oversaw their execution, checked every reported number against the accepted CFD records, and take responsibility for all analyses, claims, and text.

REPRODUCIBILITY STATEMENT

Sections 2, 4, and 5 describe the benchmark, optimization methods, experimental budgets, and statistical analysis. Appendix B gives the agent comparisons, while Appendices C, D, and E provide the failure-classifier analysis, sensitivity checks, and CFD outcomes. The accompanying materials, an anonymized repository at `https://anonymous.4open.science/r/wingbench`, support three levels of reproduction. First, every table and number in the paper, except the counts taken from the agents' execution traces, can be recomputed from the released records with the analysis scripts, which need no CFD. Second, every evaluated wing but five is released with its 38 coordinates, and every evaluation with a result document, 6,241 of the 6,629, with its per-condition lift, drag, angle of attack, and convergence values, so a design can be regraded, and the CFD evaluator can be rebuilt from a container recipe, a fetch script for the SuperWing simulation files, and the public surrogate weights, all pinned by fingerprint. Third, the benchmark folder holds a runner that lets a new optimizer play episodes under the released episode machine, a verifier that replays its trace, and a scorer that compares it with the six reference methods. The runner was tested with recorded outcomes, and its CFD path was run once in a rebuilt environment on a fresh machine of the reference type, where it reproduced a recorded evaluation exactly, to the last digit of lift, drag, and angle of attack at every condition (Appendix E). The SuperWing run records are redistributed with the authors' permission. The code that ran jobs on our cloud and cluster machines, the agent session controller, and the full agent traces are not included, and the recorded results were graded on the machine type the rules name.

REFERENCES

François Bachoc, Céline Helbert, and Victor Picheny. Gaussian process optimization with failures: classification and convergence proof. *Journal of Global Optimization*, 78(3):483–506, 2020.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024.

Jasper Bussemaker, Paul Saves, Nathalie Bartoli, Thierry Lefebvre, and Björn Nagel. Surrogate-based optimization of system architectures subject to hidden constraints. In *AIAA Aviation Forum and Ascend 2024*, AIAA Paper 2024-4401, 2024.

Ankush Chakrabarty, Scott A. Bortoff, and Christopher R. Laughman. Simulation failure robust Bayesian optimization for estimating black-box model parameters. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1533–1538, 2021.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025.

Ziru Chen, Michael White, Ray Mooney, Ali Payani, Yu Su, and Huan Sun. When is tree search useful for LLM planning? It depends on the discriminator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13659–13678, 2024.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel

Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025.

Shaghayegh Fazliani, Krissh Chawla, Jack Guo, Yiren Shen, Matthias Ihme, and Madeleine Udell. ShapeBench: A scalable benchmark and diagnostic suite for standardized evaluation in aerodynamic shape optimization. *arXiv preprint arXiv:2605.20763*, 2026.

Florian Felten, Gabriel Apaza, Gerhard Bräunlich, Cashen Diniz, Xuliang Dong, Arthur Drake, Milad Habibi, Nathaniel Hoffman, Matthew Keeler, Soheyl Massoudi, Francis VanGessel, and Mark Fuge. EngiBench: A framework for data-driven engineering design research. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. Can LLMs beat classical hyperparameter optimization algorithms? A study on autoresearch. *arXiv preprint arXiv:2603.24647*, 2026.

Alexander I. J. Forrester and Andy J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1–3):50–79, 2009.

Michael A. Gelbart, Jasper Snoek, and Ryan P. Adams. Bayesian optimization with unknown constraints. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pp. 250–259, 2014.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R. D. Costa, José R. Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Accelerating scientific discovery with Co-Scientist. *Nature*, 655(8122):487–496, 2026. doi: 10.1038/s41586-026-10644-y.

Rushil Gupta, Jason Hartford, and Bang Liu. LLMs for Bayesian optimization in scientific domains: Are we there yet? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 15482–15510, 2025.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

Youwei He, Jinju Sun, Peng Song, and Xuesong Wang. Dual Kriging assisted efficient global optimization of expensive problems with evaluation failures. *Aerospace Science and Technology*, 105:106006, 2020.

Youwei He, Jinju Sun, Peng Song, and Xuesong Wang. Variable-fidelity expected improvement based efficient global optimization of expensive problems in presence of simulation failures and its parallelization. *Aerospace Science and Technology*, 111:106572, 2021.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309, 2024.

Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1799–1808. PMLR, 2017.

Brenda M. Kulfan. Universal parametric geometry representation method. *Journal of Aircraft*, 45(1): 142–158, 2008.

Sébastien Le Digabel and Stefan M. Wild. A taxonomy of constraints in black-box simulation-based optimization. *Optimization and Engineering*, 25(2):1125–1143, 2024.

Herbert K. H. Lee, Robert B. Gramacy, Crystal Linkletter, and Genetha A. Gray. Optimization subject to hidden constraints via statistical emulation. *Pacific Journal of Optimization*, 7(3):467–478, 2011.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: A principled architecture for autonomous LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 31186–31261. PMLR, 2024.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651(8107):914–919, 2026. doi: 10.1038/s41586-026-10265-5.

Zhoujie Lyu, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins. Aerodynamic shape optimization investigations of the Common Research Model wing benchmark. *AIAA Journal*, 53(4):968–985, 2015.

Charles A. Mader, Gaetan K. W. Kenway, Anil Yildirim, and Joaquim R. R. A. Martins. ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. *Journal of Aerospace Information Systems*, 17(9):508–527, 2020.

Joaquim R. R. A. Martins. Aerodynamic design optimization: Challenges and perspectives. *Computers & Fluids*, 239:105391, 2022.

Jamison Meindl, Yunsheng Tian, Tony Cui, Veronika Thost, Zhang-Wei Hong, Jie Chen, Wojciech Matusik, and Mina Konaković Luković. SemanticOpt: Towards LLM-based semantic black-box optimization. *arXiv preprint arXiv:2510.25404v3*, 2026.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Mikhail Plekhanov, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A new framework and benchmark for advancing AI research agents. In *Second Conference on Language Modeling*, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Punya Plaban, Peter Bachman, and Ashwin Renganathan. Derivative-free optimization is competitive for aerodynamic design optimization in moderate dimensions. *Structural and Multidisciplinary Optimization*, 69(6):154, 2026.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Matthieu Sacher, Régis Duvigneau, Olivier Le Maître, Mathieu Durand, Élisa Berrini, Frédéric Hauville, and Jacques-André Astolfi. A classification approach to efficient global optimization in presence of non-computable domains. *Structural and Multidisciplinary Optimization*, 58(4): 1537–1557, 2018.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent Laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023.

James E. Smith and Robert L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

Nithin Somasekharan, Ling Yue, Yadi Cao, Weichao Li, Patrick Emami, Pochinapeddi Sai Bhargav, Anurag Acharya, Xingyu Xie, and Shaowu Pan. CFDLLMBench: A benchmark suite for evaluating large language models in computational fluid dynamics. *Journal of Data-centric Machine Learning Research*, 3(13):1–40, 2026.

David Stenger and Dirk Abel. Benchmark of Bayesian optimization and metaheuristics for control engineering tuning problems with crash constraints. *arXiv preprint arXiv:2211.02571*, 2022.

Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

Kyle Swanson, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou. The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature*, 646(8085):716–723, 2025. doi: 10.1038/s41586-025-09442-9.

Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian optimization with max-value entropy search and its parallelization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9334–9345. PMLR, 2020.

Ali Tfaily, Youssef Diouane, Nathalie Bartoli, and Michael Kokkolaras. Bayesian optimization with hidden constraints for aircraft design. *Structural and Multidisciplinary Optimization*, 67(7):123, 2024. doi: 10.1007/s00158-024-03833-8.

Anh Tran, Jing Sun, John M. Furlan, Krishnan V. Pagalthivarthi, Robert J. Visintainer, and Yan Wang. pBO-2GP-3B: A batch parallel known/unknown constrained Bayesian optimization with feasibility classification and its applications in computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 347:827–852, 2019.

John C. Vassberg, Mark DeHaan, Melissa Rivers, and Richard A. Wahls. Development of a Common Research Model for applied CFD validation studies. In *26th AIAA Applied Aerodynamics Conference*, AIAA Paper 2008-6919, 2008.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Yunjia Yang, Babak Gholami, Caglar Gurbuz, Mohammad Rashed, and Nils Thuerey. Towards a foundation-model paradigm for aerodynamic prediction in three-dimensional design. *arXiv preprint arXiv:2604.18062*, 2026a.

Yunjia Yang, Weishao Tang, Mengxin Liu, Nils Thuerey, Yufei Zhang, and Haixin Chen. SuperWing: A comprehensive transonic wing dataset for data-driven aerodynamic design. *Scientific Data*, 13: 1113, 2026b. doi: 10.1038/s41597-026-07769-0.

Ling Yue, Nithin Somasekharan, Tingwen Zhang, Yadi Cao, Zhangze Chen, Shimin Di, and Shaowu Pan. Foam-Agent: A large language model-based multi-agent framework for automating computational fluid dynamics workflows. *Computer Methods in Applied Mechanics and Engineering*, 461: 119271, 2026.

A LIMITATIONS

WingBench focuses on transonic wing design under a fixed simulation protocol and computational budget. Ten seeds per method and task provide limited power to detect small performance differences. Delivery rates also depend on the computing environment: time limits account for 31% of baseline nondeliveries, with their impact bounded in Appendix E. Planning results capture the combined contributions of the selected plans and the executor's reasoning. Future work could extend the benchmark to other design domains, increase the number of independent runs, and evaluate additional search strategies across different computational budgets.

B AGENT COMPARISONS

All agents used DeepSeek V4.1 Flash on a dedicated deployment, through an OpenAI-compatible chat interface with reasoning effort at its maximum. The requests set no temperature or other sampling option, so the provider's defaults applied. Each model response could use up to 262,144 output tokens within the model's context of 1,048,576 tokens, and each turn could make up to 64 rounds of tool calls. The agent episodes ran in September 2026. For the agents, a seed is a label. The same ten labels match episodes across methods, and they do not set the model's sampling.

The planning value rule is fixed. For a candidate first action, the evaluator predicts the probability p that the evaluation completes and, given completion, the 10th, 50th, and 90th percentiles of its L/D. The rule turns this into four scenarios, incomplete with weight $1 - p$ and complete with weights $0.3p$, $0.4p$, and $0.3p$ at the three percentiles, the Pearson–Tukey three-point approximation. With β_t the best complete L/D measured after step t of the plan, zero if none, and b_0 that best before the plan, the value of a two-step plan is $\mathbb{E}[(\beta_1 + \beta_2)/2]$ over the scenario tree, where the second step's predictions are conditional on whether the first step was incomplete, complete without improving on b_0, or improved on it. Plans are ranked by this value, then by the first step's completion probability.

The agent recommendation data contain 270 assessments at queries 1, 2, and 4, across three methods, three tasks, and ten seeds. Each assessment identifies its episode, checkpoint, geometry, delivery outcome, and conditional L/D. Table 5 compares open-ended research with each alternative at query 4. Because a seed label does not pair agent episodes, the two methods' ten episodes are resampled independently. A quality difference is the difference between the two conditional means in Table 4, and its interval comes from the delivered recommendations of each resample. Delivery differences carry Newcombe score intervals, because a bootstrap of two arms that both deliver every recommendation has no spread. The intervals describe uncertainty in these individual comparisons and do not account for multiple comparisons or the choice of which agent to continue.

The data also contain 90 recommendations made before the first online query. Before any CFD feedback, 64 of the 90 delivered, including 18 of 30 for open-ended research and 23 of 30 for both reflection and planning. The four surrogate-assisted baselines delivered 7 to 13 of 30 at this checkpoint. Among the delivered agent recommendations, mean L/D is 16.5 to 16.8 at low load, 28.2 to 29.1 at nominal load, and 24.4 to 26.1 at high load, below every value at query 4 in Table 4. At this point, agents have only the surrogate and the SuperWing records to guide them.

The 32-query continuation asked for no further checkpoint recommendations, but agents could record one at any time, and all 30 open-ended episodes did. Taking the last recommendation recorded by 32 results as the standing recommendation, 28 of 30 name a wing that delivered online, 2 name a wing that failed, and 20 name the best delivered wing. The delivered ones average 18.21, 33.03, and 30.43 L/D at low, nominal, and high load, within 0.2 of the best online values in Table 3. At 8 and 16 results, 19 and 14 of the standing recommendations were untested wings with unknown delivery.

Ten of the extension's 960 online evaluations, in five episodes and covering five distinct wings, ended before CFD began, with the grader's exit code and a reason code saying that the grading job exited without a result. Nine fell within twelve minutes on one day, and one agent resubmitted the same wing five times into that outage. They were recorded as inconclusive outcomes and consumed the agent's queries, which the released rules do not allow. Excluding them raises online delivery in Table 3 from 0.62 to 0.63 at low load and from 0.70 to 0.71 at nominal load, leaves 0.62 at high load, and changes no episode maximum. We regraded the five wings after the study on fresh machines of the reference type. The high-load wing was rejected as unmeshable in three minutes, so that outcome

was the wing's. Three of the other four delivered, at 18.39, 30.59, and 33.97, and the fourth was valid at two conditions and hit the two-hour limit at the third. The low-load wing of episode A070 exceeded its episode's best online score of 18.35. The linked repository's regrade table lists them.

The model sessions of the agent study used 71.5 million input tokens, 29.1 million output tokens, and 6.2 billion cached input tokens, and cost about $118 at the provider's list prices, or under $0.25 per CFD query for every method. Per CFD query, open-ended research averaged 9 minutes of model time and $0.08, planning 25 minutes and $0.15, and reflection 31 minutes and $0.23. The released tables list every session and every recorded recommendation.

Table 5: Open-ended research minus each comparator after four CFD queries, with the two methods' episodes resampled independently. Quality differences compare delivered recommendations, counted in the delivered column for each method. Positive differences favor open-ended research. Quality intervals use 10,000 bootstrap resamples and delivery intervals are Newcombe's.

comparator	C_L	Δ delivery	95% CI	delivered	$\Delta L/D$	95% CI
Planning	0.15	-0.10	[-0.40, 0.19]	9/10	-0.26	[-0.85, 0.31]
Planning	0.41	0.10	[-0.19, 0.40]	10/9	1.77	[0.32, 3.21]
Planning	0.66	0.10	[-0.19, 0.40]	10/9	0.24	[-1.76, 2.15]
Reflection	0.15	-0.10	[-0.40, 0.19]	9/10	-0.19	[-0.67, 0.31]
Reflection	0.41	0.00	[-0.28, 0.28]	10/10	1.26	[-0.03, 2.55]
Reflection	0.66	0.00	[-0.28, 0.28]	10/10	1.04	[-0.33, 2.38]

Table 6 gives each reported mean its own interval, an exact binomial interval for the delivery rate and a bootstrap interval for the conditional mean from 10,000 resamples of the ten episodes of each method and task. At nominal load, the delivered mean of open-ended research, 31.09, has interval [30.04, 32.11], and planning's 29.31 has [28.28, 30.30]. Among the baselines, the archive-only methods' delivered means rest on four or five episodes, and SMAC3's rest on scores that vary widely.

Table 6: 95% intervals for the recommendation delivery rate (exact binomial) and for the mean L/D of delivered recommendations (percentile bootstrap from 10,000 resamples of each method's ten episodes per task). Agents at query 4, baselines at query 32. A dash marks a cell with no delivery.

method	C_L	delivered	95% CI	mean L/D	95% CI
Agents after four queries					
Open-ended research	0.15	9/10	[0.56, 1.00]	17.15	[16.73, 17.56]
	0.41	10/10	[0.69, 1.00]	31.09	[30.04, 32.11]
	0.66	10/10	[0.69, 1.00]	27.83	[26.58, 29.02]
Planning	0.15	10/10	[0.69, 1.00]	17.41	[16.99, 17.80]
	0.41	9/10	[0.56, 1.00]	29.31	[28.28, 30.30]
	0.66	9/10	[0.56, 1.00]	27.59	[26.03, 29.20]
Reflection	0.15	10/10	[0.69, 1.00]	17.33	[17.09, 17.58]
	0.41	10/10	[0.69, 1.00]	29.83	[29.05, 30.58]
	0.66	10/10	[0.69, 1.00]	26.79	[26.15, 27.41]
Baselines after 32 queries					
CMA-ES, no feedback	0.15	0/10	[0.00, 0.31]	–	–
	0.41	4/10	[0.12, 0.74]	33.22	[32.85, 33.63]
	0.66	5/10	[0.19, 0.81]	32.12	[32.00, 32.23]
CMA-ES, selection only	0.15	0/10	[0.00, 0.31]	–	–
	0.41	4/10	[0.12, 0.74]	33.22	[32.87, 33.63]
	0.66	5/10	[0.19, 0.81]	32.12	[31.99, 32.23]
Feedback-guided CMA-ES	0.15	6/10	[0.26, 0.88]	16.76	[15.60, 18.01]
	0.41	10/10	[0.69, 1.00]	33.04	[32.68, 33.39]
	0.66	10/10	[0.69, 1.00]	31.64	[31.25, 31.98]
Feedback-guided DE	0.15	10/10	[0.69, 1.00]	17.52	[17.04, 17.97]
	0.41	10/10	[0.69, 1.00]	31.09	[30.61, 31.53]
	0.66	10/10	[0.69, 1.00]	27.05	[25.83, 28.04]
SMAC3	0.15	9/10	[0.56, 1.00]	10.56	[7.52, 13.63]
	0.41	7/10	[0.35, 0.93]	17.00	[13.77, 20.12]
	0.66	6/10	[0.26, 0.88]	12.82	[8.74, 16.93]

C Offline failure-classifier details

The source records provide outcomes for 4,544 designs, including the 4,239 released geometries and attempted designs with no valid condition. We standardized the 38 variables with the benchmark normalization and predicted whether a design fails any SuperWing condition. Logistic regression, 15-nearest-neighbors, and histogram gradient boosting reach five-fold AUC 0.69, 0.65, and 0.71. Training on the central 85% and testing on the extreme 15% reduces these to 0.62, 0.52, and 0.63.

We then applied the source-trained classifiers to the 3,840 designs proposed by the four surrogate-assisted baseline methods and labeled them with their actual WingBench outcomes. AUC is 0.41, 0.52, and 0.50 at low load and 0.51–0.75 at nominal and high load. These values are per task because online nondelivery differs sharply across tasks. The experiment does not prove that failure prediction is impossible. It shows that the tested source models do not remove the need for online learning.

D SENSITIVITY TO THE INCONCLUSIVE-OUTCOME WEIGHT

The two feedback-guided baseline methods give inconclusive CFD outcomes weight 0.5 in their delivery model. Without changing any completed search, we refitted the belief for the 103 checkpoint recommendations that could depend on this weight. Using weight 1 changes 21 recommendations, and using weight 0 changes 33. At query 32, only four of 60 episodes lack a delivered online design and so use this merit-based path, and two recommendations change between weights 0.5 and 1. Both alternative designs and both designs they replace fail at the mesh stage. Consequently no reported final endpoint changes. This is a recommendation sensitivity check, not a full rerun, since changing the weight during search could change the proposal trajectory.

E CFD OUTCOMES

The two studies contain 6,629 graded CFD requests. Of these, 2,620 returned a score and 4,009 did not. Table 7 accounts for the online evaluations and private recommendation evaluations in both studies. These are counts of graded requests, including accepted results reused for repeated designs, rather than counts of distinct solver executions. They cover at least 5,205 distinct wings. The 6,576 requests with recorded run times used 200,019 processor-core hours of CFD in total.

Table 7: CFD score delivery across both studies. Private evaluations score recommendations separately from the online search when a prior online grade cannot be reused.

study	evaluation	delivered a score	no score	total
Five baseline optimizers	online	1,648	3,152	4,800
	private	62	431	493
Three agent methods	online	806	394	1,200
	private	104	32	136
Both studies	all	2,620	4,009	6,629

The baseline study contributes 5,293 graded requests, including 3,583 nondeliveries. Table 8 assigns each baseline outcome to its first failing stage. Of 954 evaluations that build a mesh and attempt all three conditions but do not deliver, 375 produce a valid result at two of three conditions. This group differs slightly from the 955 time-limit and solver-error outcomes in Table 8. It adds five other errors that attempted all three conditions and leaves out six outcomes that have no condition records. A nondelivery is therefore often a near miss, not a geometry from which the pipeline obtains nothing.

Table 8: CFD outcomes in the five-method baseline study. Private counts include only recommendations requiring a separate grade. Counts are evaluations, not distinct wings.

outcome	online	checkpoint	total
delivered a score	1,648	62	1,710
mesh generator rejected geometry	438	65	503
mesh generator stopped without a mesh	1,825	282	2,107
flow condition reached time limit	552	47	599
solver process stopped with an error	319	37	356
other errors	18	0	18
all nondeliveries	3,152	431	3,583
evaluations	4,800	493	5,293

The agent study contributes 1,336 graded requests. Its 136 private grades include 90 before the first online query and 46 at later recommendation checkpoints. Another 224 recommendation assessments reused online grades and add no new request to this count. Table 9 reports the recorded agent outcome types for both online and private grades. A partial outcome has usable results for only some of the required flow conditions and does not receive a complete wing score.

Both tables come from the same grader. Table 9 uses the four outcome types that the feedback-guided baselines weight in their delivery model, and agents receive each result together with the files the evaluation produced. Table 8 groups the grader's reason codes by the first failing stage. In baseline records with both labels, every mesh rejection is a conclusive failure and every mesh stop undetermined. Of the 955 time-limit and solver-error outcomes, the 641 with a type label are partial in 389 cases and undetermined in 252. The other 314 are SMAC3 records, which carry no type label.

Table 9: Recorded CFD outcome types in the three-method agent study. A pass delivers a complete score. The other outcome types do not. Private counts are the 136 separate grades.

outcome	online	private	total
pass	806	104	910
partial	212	6	218
undetermined	137	17	154
conclusive failure	45	9	54
evaluations	1,200	136	1,336

Mesh rejection in the baseline study is a geometry failure under the chosen meshing procedure. Stopping without a mesh is recorded as inconclusive because the flow solver never tests the wing. Most of these stops are crashes, not time limits. Of the 2,100 mesh stops with a recorded attempt, 1,582 ended in a process error after a median of 125 seconds, 1,526 of them with a segmentation fault, and 518 reached the 30-minute limit. A flow timeout consumes the full condition budget without producing a valid result. Outcomes set by a time limit, these 518 mesh stops and the 599 flow timeouts, make up 31% of baseline nondeliveries and could change on faster hardware. If every clock-limited nondelivery were instead counted as unknown, baseline online delivery would lie between 0.34 and 0.54 overall, between 0.53 and 0.66 for feedback-guided CMA-ES, 0.51 and 0.67 for feedback-guided DE, 0.21 and 0.43 for no-feedback CMA-ES, 0.21 and 0.41 for selection-only CMA-ES, and 0.26 and 0.55 for SMAC3. Mesh stops are concentrated at low load (1,212 of 2,107) and solver-process errors at high load (276 of 356). The records also show how repeatable outcomes are. 809 task and wing pairs were evaluated more than once in separate runs, with different mesh times. Delivery agreed in 800 of them (98.9%). All nine disagreements were high-load time limits or solver-process errors. The raw release preserves condition-level evidence and keeps infrastructure failures separate from these scientific outcomes.

The rebuilt environment was tested once. On 2026-09-25 we built the CFD evaluator from the released container recipe on a new machine of the reference type. The container built from the recipe, the fetch script matched all four pinned SuperWing files, and the local grading adapter regraded a recorded nominal-load wing. It reproduced the recorded score of 33.4018 exactly, with lift, drag, and angle of attack identical to the last digit at all three conditions, in 7,973 seconds. Only the runtime fingerprint differed, as documented, because it binds the original machines. The regrades of the five outage wings in Appendix B used the same rebuilt evaluator on machines of the same type.

F Formal problem statement and method equations

This appendix states the benchmark and the reference methods in symbols. It adds no new claim. Every quantity is the one the released code computes, and Table 10 lists the settings.

F.1 Design space, evaluator, and score

A design is a vector $z \in \mathcal{B} = [-3.5, 3.5]^{38}$ of standardized coordinates. Its physical parameters are $x = \mu + \sigma \odot z$, where μ and σ hold the per-coordinate mean and standard deviation of the SuperWing designs and $\odot$ is the coordinatewise product. A task is a target lift coefficient $C_L^* \in \{0.15, 0.41, 0.66\}$. Every design is evaluated at three flow conditions $c \in \{1, 2, 3\}$ with Mach numbers 0.7853, 0.8195, and 0.8530 and Reynolds number 2×10^7, as in Section 2.

For one condition the evaluator meshes the wing, adjusts the angle of attack α until the computed lift coefficient matches the target, and returns the lift-to-drag ratio at that angle,

$$r_c(x) = \frac{C_L(x, c, \alpha_c^*)}{C_D(x, c, \alpha_c^*)}, \qquad \left|C_L(x, c, \alpha_c^*) - C_L^*\right| \le 0.02. \tag{1}$$

The condition is valid only if a mesh exists, the solver process exits normally, the solver's lift oscillation measure is below 5×10^{-4}, the trimmed angle lies in $[-10^\circ, 25^\circ]$, and $0 < r_c(x) \le 40$. Otherwise the condition returns a failure type instead of a number. The score is the geometric mean over the three conditions,

$$y(x) = \Big(\prod_{c=1}^{3} r_c(x) \Big)^{1/3}, \tag{2}$$

and it exists only when all three conditions are valid. We write $d(x) \in \{0, 1\}$ for this event, delivery. When $d(x) = 0$ the evaluation returns one of three outcome types. A conclusive failure means the geometry was rejected or a condition was conclusively invalid. Partial means some conditions returned values and none was invalid. Undetermined means no condition returned one. In the released records every conclusive failure is a rejected geometry.

The surrogate $s(x)$ is the same geometric mean computed from AeroTransformer predictions with the same trim procedure. One surrogate evaluation takes about a second and returns a number for every finite x, including designs with $d(x) = 0$.

F.2 An episode

An episode is one search on one task. Let $\mathcal{H}_k$ denote everything the optimizer has observed after k online evaluations, that is, its surrogate queries, the designs it submitted, and for each submitted design either the score or the failure type. A policy π maps $\mathcal{H}_k$ either to a batch of at most 16 surrogate queries or to the next online design x_{k+1}. The budgets are

$$\text{surrogate evaluations} \le 4096, \qquad k \le 32, \tag{3}$$

and online evaluations are strictly sequential, so x_{k+1} may depend on $d(x_k)$ and $y(x_k)$. At each checkpoint $q \in \mathcal{Q} = \{0, 1, 2, 4, 8, 16, 32\}$ the policy also names one recommendation $\rho_q \in \mathcal{B}$, the design it would hand to a user. The recommendation is graded by the same evaluator, the grade is never shown to the policy, and it costs no budget. If ρ_q was already evaluated online, the grade reuses that result. A design submitted twice in one episode returns its recorded result and still costs a query.

F.3 Endpoints

For one method and one task with episodes $e = 1, \dots, n$, where $n = 10$ seeds, write $d_q^e = d(\rho_q^e)$ and $y_q^e = y(\rho_q^e)$. Then

$$D_q = \frac{1}{n} \sum_{e=1}^{n} d_q^e, \qquad Q_q = \frac{\sum_e d_q^e \, y_q^e}{\sum_e d_q^e}, \qquad O = \frac{1}{32\,n} \sum_{e=1}^{n} \sum_{k=1}^{32} d(x_k^e). \tag{4}$$

D_q is recommendation delivery, Q_q is conditional quality, defined only when some recommendation delivers, and O is online delivery. The paper reports D_q and Q_q at $q = 32$ for the baselines and at $q = 4$ for the agents. The delivery averaged across tasks in Section 5 is the mean of D_{32} over tasks.

F.4 The belief model of the surrogate-assisted baselines

The four surrogate-assisted baselines share one belief model, fitted anew in every episode from the 62 warm-start results and the episode's own online outcomes. Its feature map is

$$\phi(x) = \big(1,\ z_1/3.5,\ \dots,\ z_{38}/3.5,\ \log s(x) - \log 30\big) \in \mathbb{R}^{40}. \tag{5}$$

The delivery model is a weighted logistic regression with an ℓ_2 penalty,

$$\begin{aligned} p(x) &= \mathrm{clip}\big(\sigma(w^\top \phi(x)),\ 0.05,\ 0.95\big), \\ w &= \arg\min_{w} \sum_i \omega_i \, \ell\big(d_i, \sigma(w^\top \phi(x_i))\big) + 4\,\|w_{1:39}\|^2 + 0.1\,w_0^2, \end{aligned} \tag{6}$$

where ℓ is the logistic loss, the label d_i is 1 for a pass and 0 otherwise, and the weight ω_i is 1 for a pass or a conclusive failure and 0.5 for a partial or undetermined outcome. Appendix D varies this

weight. The fit runs 40 iterations. The discrepancy model is a ridge regression, on the delivered results, of the log ratio between the CFD score and the surrogate score,

$$\delta_i = \log y_i - \log s(x_i), \qquad \hat{\delta}(x) = v^\top \phi(x),$$
$$v = \arg\min_v \sum_{i:\, d_i=1} \big(\delta_i - v^\top \phi(x_i)\big)^2 + 10\,\|v_{1:39}\|^2 + 0.1\,v_0^2, \tag{7}$$

with a predictive standard deviation $\hat{\sigma}(x)$ floored at 0.05 and set to 0.25 while no delivered result exists. The private merit of a design combines the two models,

$$m(x) = \log p(x) + \log s(x) + \hat{\delta}(x) + \tfrac{1}{2}\,\hat{\sigma}(x) + \tfrac{1}{10}\,H\big(p(x)\big), \qquad H(p) = -p\log p - (1-p)\log(1-p). \tag{8}$$

The first three terms are the log of the expected delivered score, the delivery probability times the corrected surrogate score. The last two terms reward uncertainty in value and delivery.

F.5 The five baseline variants

All four surrogate-assisted variants run their generator in unit coordinates on the box with a population of 16 and 256 generations, which is the 4,096-evaluation budget, and keep an active archive of the 64 best cheap-valid designs. A design whose surrogate evaluation is not finite receives a loss of 10^{12}.

CMA-ES, no feedback. The objective is $s(x)$. All 256 generations run before the first online query. The 32 best distinct archive designs are queued in descending s, and online results change nothing.

CMA-ES, selection only. The same search and the same archive. At online step k the method submits $\arg\max m(x)$ over the unqueried archive designs, refitting p and $\hat{\delta}$ after every result. The archive never changes, so feedback picks the next design but cannot create a new one.

Feedback-guided CMA-ES. One continuing CMA-ES state with initial mean 0.5 in unit coordinates and initial step size $\sigma_0 = 0.25$. Before each online query it runs eight generations, 128 surrogate evaluations, with objective $m(x)$, then submits the archive $\arg\max$ of m. Both models are refitted after every result, so feedback changes the selection and the proposal distribution.

Feedback-guided DE. DE/rand/1/bin with $F = 0.8$ and $CR = 0.9$ replaces the CMA-ES state. A trial replaces its parent when its merit is at least the parent's, and a mutant coordinate outside the box is redrawn uniformly. Schedule and selector are those of feedback-guided CMA-ES.

SMAC3. The black-box facade of SMAC 2.4.0 with its Gaussian-process model, 8 Sobol initial designs, 24 model-guided queries, one evaluation per configuration, and a common penalty value for every nondelivery. It uses no surrogate and no warm-start results.

Recommendation rule. For every baseline, if any online evaluation has delivered by checkpoint q,

$$\rho_q = \arg\max_{k \le q:\, d(x_k)=1} y(x_k), \tag{9}$$

which is the best delivered online design. Otherwise the recommendation is an untested design chosen by the method's own rule, the unqueried archive design with the highest s for the no-feedback method, with the highest selector value for the selection-only method, and with the highest m for the two feedback-guided methods, and the next proposal for SMAC3. Ties break by a stable candidate identifier. The checkpoint-32 recommendation is the terminal one.

F.6 Statistical procedures

Seed-block contrasts for the baselines. For two methods A and B and seed j, the delivery effect at checkpoint 32 is the mean over the three tasks t of $d_{A,t,j} - d_{B,t,j}$, and the quality effect is the mean of $y_{A,t,j} - y_{B,t,j}$ over the tasks where both recommendations deliver, undefined when there is none. With n_b available blocks and effects $\Delta_1, \ldots, \Delta_{n_b}$, the estimate is $\bar{\Delta}$ and the exact two-sided sign-flip p value is

$$p = 2^{-n_b}\,\big|\{\varepsilon \in \{-1,+1\}^{n_b} : \big|\tfrac{1}{n_b}\textstyle\sum_j \varepsilon_j \Delta_j\big| \ge |\bar{\Delta}|\}\big|. \tag{10}$$

The 95% interval is the 2.5th and 97.5th percentile of $\bar{\Delta}$ over 10,000 resamples of the blocks with replacement, drawn from a PCG64 generator with seed 73001, fixed in the analysis plan before

any result was known. The six planned tests are corrected by Holm's step-down procedure. With ordered p values $p_{(1)} \le \cdots \le p_{(6)}$ the adjusted values are $\tilde{p}_{(i)} = \max_{j \le i} \min\{1, (7-j)\, p_{(j)}\}$, and a contrast is significant when $\tilde{p} \le 0.05$. With seven blocks the smallest attainable p in Equation 10 is $2/128 = 0.016$, which is why Section 6 reports the quality contrasts as underpowered, not null.

Agent contrasts. A seed label does not pair agent episodes, so the two methods' ten episodes are resampled independently, 10,000 times with the same generator. The quality difference is the difference between the two conditional means of Equation 4, and each resample recomputes both means from its delivered recommendations. Table 5 lists them.

Intervals for a single rate. For k delivered recommendations out of n, Table 6 reports the exact Clopper–Pearson interval

$$\big[B_{\alpha/2}(k,\, n-k+1),\; B_{1-\alpha/2}(k+1,\, n-k)\big], \tag{11}$$

where $B_u(a, b)$ is the u quantile of the beta distribution, the ends are 0 when $k = 0$ and 1 when $k = n$, and $\alpha = 0.05$. Its coverage is at least the nominal 95% for every k.

Intervals for a difference of rates. A bootstrap of two arms that both deliver every recommendation has no spread, so Table 5 uses Newcombe's score interval. With $\hat{p} = k/n$, $z = 1.96$, and the Wilson bounds

$$l, u = \frac{\hat{p} + z^2/2n \;\mp\; z\sqrt{\hat{p}(1-\hat{p})/n + z^2/4n^2}}{1 + z^2/n}, \tag{12}$$

the interval for $\hat{p}_1 - \hat{p}_2$ is

$$\Big[\hat{p}_1 - \hat{p}_2 - \sqrt{(\hat{p}_1 - l_1)^2 + (u_2 - \hat{p}_2)^2},\; \hat{p}_1 - \hat{p}_2 + \sqrt{(u_1 - \hat{p}_1)^2 + (\hat{p}_2 - l_2)^2}\Big]. \tag{13}$$

F.7 Two remarks

Why the surrogate is optimistic where the search goes. Write the surrogate score of candidate i as $s_i = y_i + \epsilon_i$ with an error ϵ_i that has mean zero for a randomly chosen design. A search that screens many candidates and keeps $i^* = \arg\max_i s_i$ selects for large positive errors as much as for large true scores. This is the optimizer's curse of Smith & Winkler (2006). Under their assumption that the error has mean zero given the true score, selecting the maximum gives $\mathbb{E}[\, s_{i^*} - y_{i^*}\,] \ge 0$, and the gap grows with the number of candidates screened and with the error variance. In WingBench the error variance is largest away from the SuperWing data, which is where optimization pushes, and the same selection applies to delivery, since a design whose failure the surrogate does not see is favored exactly because that failure is not priced. Appendix C measures the consequence, and the agents that fitted their own models in Section 6 met the same effect.

Why the recommendation rule is the best delivered design. Repeated evaluations of the same wing agreed in 800 of 809 cases (Appendix E), so a private regrade of an already delivered design returns its online score. Among the designs an optimizer has evaluated, Equation 9 therefore maximizes the delivered score with certainty. An untested design x can only beat it in expectation when $p(x)\, \mathbb{E}[\, y(x) \mid d(x) = 1\,]$ exceeds the best delivered score, and the baselines never take that gamble once a design has delivered. This is a conservative rule, chosen because the benchmark asks for a design worth handing to a user, not for the design with the best expected score.

Table 10: Settings of the reference methods and of the analysis, as fixed in the released configurations.

setting	value
design box	$[-3.5, 3.5]^{38}$ standardized, edge tolerance 10^{-9}
flow conditions	Mach 0.7853, 0.8195, 0.8530 at $\mathrm{Re} = 2 \times 10^{7}$
lift match	$\lvert C_L - C_L^* \rvert \le 0.02$, lift oscillation $< 5 \times 10^{-4}$, $\alpha \in [-10^\circ, 25^\circ]$
score envelope	$0 < r_c \le 40$ for every condition
time limits	30 min per mesh, 2 h per flow condition, wall clock
reference machine	one job at a time on an AWS m7a.4xlarge, 16 cores, 64 GB
surrogate budget	4,096 evaluations in batches of at most 16
online budget	32 sequential CFD evaluations
checkpoints	0, 1, 2, 4, 8, 16, 32
warm start	the same 62 CFD results for the four surrogate-assisted baselines
population, generations	16, 256 (eight generations per online query when interleaved)
active archive	64 designs
CMA-ES	initial mean 0.5 in unit coordinates, $\sigma_0 = 0.25$, no mirrors
DE	DE/rand/1/bin, $F = 0.8$, $CR = 0.9$
invalid surrogate value	loss 10^{12}
delivery model	logistic, ℓ_2 penalty 4 (intercept 0.1), 40 iterations, clip to $[0.05, 0.95]$
outcome weights	pass 1, conclusive failure 1, partial 0.5, undetermined 0.5
discrepancy model	ridge in $\log y - \log s$, penalty 10 (intercept 0.1), $\hat{\sigma} \ge 0.05$, 0.25 with no label
merit	$\log p + \log s + \hat{\delta} + 0.5\,\hat{\sigma} + 0.1\,H(p)$
SMAC3	version 2.4.0, black-box facade, 8 Sobol points, 24 model-guided queries
agents	DeepSeek V4.1 Flash, maximum reasoning effort, 64 tool rounds per turn
reflection	summary capped at 4,000 characters
planning	four plans of two steps, at most three cycles per decision
bootstrap	10,000 resamples, PCG64 seed 73001, 95% percentile intervals
multiple comparisons	Holm over the six planned baseline contrasts at $\alpha = 0.05$

G BENCHMARK RULES, VERSION 1.0

The release contains the rules the study ran under. A result is comparable to the reference results only if it follows them. Later versions will carry a new number, and results from different versions are reported separately. This appendix restates the rules in full.

The task. A design is a wing described by 38 numbers. The benchmark standardizes each number with the SuperWing dataset and searches the box from -3.5 to 3.5 in every coordinate. A proposal may lie at most 10^{-9} standardized units outside the box, which absorbs floating-point round trips through physical values. A wing is scored at a fixed target lift coefficient and at three Mach numbers, 0.7853, 0.8195, and 0.8530, with Reynolds number 20 million. The score is the geometric mean of the three lift-to-drag ratios. There are three tasks. They differ only in the target lift coefficient, which is 0.15 (low load), 0.41 (nominal load), or 0.66 (high load). A CFD evaluation delivers a score only if all three conditions return a valid result, the lift is within 0.02 of the target, every lift-to-drag ratio is above 0 and at most 40, and the angle of attack is between -10 and 25 degrees. Otherwise the evaluation returns a failure type and no score.

Budgets. One episode is one optimization run on one task. An episode may use at most 4,096 surrogate evaluations, where one surrogate evaluation scores one wing at all three Mach numbers, including the angle-of-attack adjustment that reaches the target lift, and at most 32 online CFD evaluations, one at a time. The optimizer submits one wing, receives its score or failure type, and only then chooses the next. At the checkpoints 0, 1, 2, 4, 8, 16, and 32 online queries, the optimizer nominates the one wing it would hand to a user. That wing is graded privately. The optimizer never sees the private grade, and the grade costs no query. A nominated wing with an online result reuses it.

What an optimizer may use. Every optimizer may use the surrogate, the SuperWing records, and the 62 earlier CFD results that the reference baselines started from. Nothing may be shared between episodes. Each track has two variants. In the closed variant only the information above may be used. In the open variant any public information may be used, including the released WingBench records. A submission states which variant it ran. The reference results are closed.

Repeated designs and failures. A wing that was already evaluated in the same episode returns the recorded result again and still costs a query. A failure of the machine is not a failure of the wing. If a worker is lost, a job is killed by the system, or a result file is never written, the evaluation is retried and does not cost a query. Failures of the wing itself are outcomes and cost the query. They are a

rejected geometry, a mesh generator that stops without a mesh, a flow condition that reaches its time limit, and a solver process that stops with an error.

Time limits and hardware. Meshing has a 30-minute limit. Each flow condition has a two-hour limit. Both are measured in wall-clock time. The reference results ran one CFD job at a time on an AWS m7a.4xlarge machine with 16 cores and 64 GB of memory, inside the released container. About one third of the reference nondeliveries were set by a time limit, so a slower machine produces more failures for the same wings. Official comparisons require an m7a.4xlarge or faster. A later version will replace the time limits with iteration limits so that results do not depend on the machine.

Recording and verification. The runner writes an append-only trace of every surrogate call, online query, result, and nomination. An independent replay checks a trace for budget use, information leakage, and consistent hashes. Private grades are kept outside the optimizer's reach until the episode ends, and their hashes enter the trace when they are produced. A submission contains the trace of every episode, the records in the formats of the released tables, and the 38 coordinates of every evaluated wing. The benchmark is run on the honor system. There is no central grading service.

Reporting. For each task and checkpoint, a submission reports recommendation delivery, the share of episodes whose nominated wing received a valid score, and conditional quality, the mean score of the delivered nominations. It also reports online delivery, the share of online queries that returned a score. The released scorer produces these next to the six reference methods, the five baselines and open-ended research, with bootstrap intervals. A submission runs ten seeds per task. For a language-model agent a seed is only a label that matches its episodes with the other methods.

Tracks. The full track runs three tasks, ten seeds, and 32 queries, and compares with every reference result. The lite track runs one task, five seeds, and 8 queries, and compares with the reference results at checkpoint 8. The recommendation-only track searches in any way and grades only the final nominated wing per episode with the container, and compares with the reference recommendations at checkpoint 32. Each track is run closed or open.

H RELEASED DATA

This appendix reproduces the data card of the release, `data/README.md`. The folder `data/` holds the CFD records behind the paper's tables and numbers. Claims that come from the agents' execution traces are not in them. The tables link to each other through the `episode` and `wing` columns, and an episode is one optimization run. Table 11 lists the files with the row counts the README states. Every count agrees with the released file. The tables that follow list every column of every file with its meaning, converted from the README.

Table 11: Released tables, with the meaning of one row and the row count from the README.

File	One row per	Rows
`episodes.csv`	episode	240
`evaluations.csv`	CFD evaluation, either an online query or a private grade	6,629
`recommendations.csv`	recommended wing graded at a checkpoint	1,410
`wings.csv`	wing evaluated in either study	5,200
`cfd_attempts.csv`	mesh or flow-condition attempt in the baseline study	13,248
`cfd_conditions.csv`	flow condition of an evaluation that has a result document	18,723
`agent_nominations.csv`	wing an agent named as its current recommendation	1,782
`agent_turns.csv`	model session of an agent episode, with its usage and cost	1,805
`weight_sensitivity_refits.csv`	recommendation that the inconclusive-outcome weight can change	103
`weight_sensitivity_wings.csv`	design that the weight chooses between at checkpoint 32	4
`regrades_after_outage.csv`	wing regraded after the study because its online request died in the grading service	5
`superwing/`	the SuperWing records used by the failure-prediction study	

H.1 EPISODES AND EVALUATIONS

Tables 12 and 13 describe `episodes.csv` and `evaluations.csv`, and Table 14 lists the grader's reason codes with the names the paper uses for them. The last three codes need a note. Ten online requests in five extension episodes carry them, nine of them within twelve minutes on 2026-09-21. They were charged to the agent as inconclusive outcomes, which the benchmark rules do not

allow. The five distinct wings were regraded after the study, see `regrades_after_outage.csv` below.

Table 12: Columns of `episodes.csv`.

Column	Meaning
`episode`	`B000` to `B149` for the baseline study, where `B120` to `B149` are SMAC3, and `A000` to `A089` for the agent study
`study`	`baseline` or `agent`
`method`	the method, named as in the paper
`task`	`low`, `nominal` or `high` load
`target_cl`	the target lift coefficient of the task
`seed`	the random seed of a baseline episode. For an agent episode it is only a label that pairs episodes across methods and does not set the model's sampling
`online_queries`	how many CFD queries the method made, which is 32, or 4 in the agent comparison
`cfd_core_seconds`	processor-core seconds of CFD run for the episode, which is wall time times the 16 cores of each job
`timed_evaluations`	how many of the episode's evaluations that time covers
`original_id`	the name the episode had while the experiment ran

Table 13: Columns of `evaluations.csv`.

Column	Meaning
`evaluation`	`E0001` to `E6629`
`episode`, `study`, `method`, `task`, `seed`	as in `episodes.csv`
`kind`	`online` for a query whose result the method saw, `private` for a grade of a recommended wing that the method never saw
`query`	the online query number, 1 to 32
`checkpoint`	for a private grade, the checkpoint of the recommendation
`wing`	the wing's identifier, a SHA-256 hash of its design
`delivered`	whether CFD returned a valid score
`ld`	the score when delivered, the geometric mean of L/D over the three flight conditions
`outcome`	`pass` (delivered), `partial` (some but not all flight conditions gave a result), `undetermined` (no condition gave a result) or `conclusive_failure` (the geometry was rejected)
`reasons`	the grader's reason codes, separated by the character `;`
`recorded_at`	when the result was recorded, in UTC
`original_id`	the name the evaluation had while the experiment ran

Table 14: The grader's reason codes and the names the paper uses for them.

Code	In the paper
`complete_score_eligible`	delivered a score
`not_run_unmeshable`	mesh generator rejected geometry
`not_run_mesh_inconclusive`	mesh generator stopped without a mesh
`inconclusive_timeout`	flow condition reached time limit
`inconclusive_infrastructure`	solver process stopped with an error
`undetermined`	other errors
`grade_exit_2`, `grade_exit_134`, `grader_exited_without_result`	the grading job stopped before CFD began (agent study only)

H.2 Recommendations and wings

Table 15 describes `recommendations.csv`. The file `wings.csv` has the column `wing` and the 38 design variables `x00` to `x37`, in physical units (Table 16). It covers the 3,294 wings of the four surrogate-assisted baselines, the 741 SMAC3 wings, and the 1,165 agent wings whose grader result documents were kept. The SMAC3 coordinates were rebuilt from the coefficient files in the grader's evidence and each was checked against its fingerprint. Five agent wings whose requests stopped before a document was written appear in the other tables by their hash only. 43 of the agent wings lie on the edge of the design box, up to 9×10^{-16} standardized units outside it once their physical values are standardized again. The campaign accepted and graded them, and the released episode machine accepts a proposal within 10^{-9} standardized units of the edge, so they can be proposed again.

Table 15: Columns of `recommendations.csv`.

Column	Meaning
`episode`, `study`, `method`, `task`, `seed`	as in `episodes.csv`
`checkpoint`	how many online queries the method had made when it recommended the wing
`wing`	the recommended wing

Column	Meaning
`delivered`, `ld`	the recommendation's grade
`graded_by`	`online` when the wing already had an online result, which is reused, or `private` when it needed its own grade, which is a row of `evaluations.csv`

Table 16: The design variables of `wings.csv`.

Columns	Variable
`x00`	leading-edge sweep, in degrees
`x01`	tip dihedral, in degrees
`x02`	inner dihedral, in degrees
`x03`	aspect ratio
`x04`	taper ratio
`x05`	kink location, as a fraction of the span
`x06`	root chord adjustment
`x07` to `x09`	thickness factors at three stations
`x10` to `x12`	camber at three stations
`x13` to `x16`	twist at four stations
`x17` to `x26`	upper-surface CST weights
`x27` to `x36`	lower-surface CST weights
`x37`	root thickness ratio

H.3 Regrades after the outage

`regrades_after_outage.csv` holds the five wings whose online requests carried the grading-job codes of Table 14, regraded after the study with `benchmark/grade_local.py` inside the rebuilt container on a fresh machine of the reference type, one row per finished regrade. These grades are not part of the study and do not enter any table in the paper. They show what the charged queries would have returned. The README describes the columns in prose, and Table 17 pairs that prose with the column names.

Table 17: Columns of `regrades_after_outage.csv`.

Column	Meaning
`wing`	the regraded wing, by its hash
`episode`, `task`	as in `episodes.csv`
`queries_charged`	the online queries that were charged for the wing, separated by the character `;`
`regraded_at`	when the regrade ran, in UTC
`machine`	the machine type and container the regrade ran on
`delivered`, `ld`	whether the regrade delivered, and its score
`outcome`, `reasons`	its outcome type and reason codes, as in `evaluations.csv`
`wall_seconds`	its wall time, in seconds

H.4 Per-attempt and per-condition records

`cfd_attempts.csv` has one row per mesh or flow-condition attempt of a baseline-study evaluation (Table 18). `cfd_conditions.csv` has one row per flow condition of every evaluation whose grader result document was kept, which is 1,297 of the 1,336 agent evaluations, 4,252 of the 4,325 non-SMAC3 baseline evaluations, and 692 of the 968 SMAC3 evaluations (Table 19). The SMAC3 documents come from its publication ledger, where each is stored encoded. The other evaluations stopped before a document was written, their documents were lost, or the only document kept belongs to a different attempt at the same wing. Measurements are empty when a condition was not solved or gave no usable result, and the lift is accepted when it is within 0.02 of the target.

Table 18: Columns of `cfd_attempts.csv`, from the README's prose.

Column	Meaning
`evaluation`, `episode`	the evaluation and episode the attempt belongs to
`stage`	`mesh` or `solve`
`condition`	the flight condition
`termination_reason`	the termination reason
`returncode`	the process return code
`outcome`	the outcome of the attempt
`wall_seconds`	the wall time of the attempt, in seconds

Table 19: Columns of `cfd_conditions.csv`.

Column	Meaning
`evaluation`, `episode`, `study`, `method`, `task`	as in `evaluations.csv`
`mesh_outcome`	the mesh step's outcome for the whole evaluation
`condition`	`mach_q1`, `mach_q2` or `mach_q3`, the three Mach numbers in order
`mach`, `cl_target`	the flight condition
`outcome`	`valid`, `invalid` or `inconclusive` for this condition
`reasons`	the grader's reason codes for this condition, separated by the character `;`
`cl`, `cd`, `ld`	the computed lift, drag, and lift-to-drag ratio
`err_cl`	the lift oscillation over the last iterations, which must stay below 0.0005
`realized_aoa_deg`	the angle of attack that reached the target lift
`seed_alpha0_deg`	the angle of attack the lift matching started from
`solvecl_iterations`, `solvecl_l2convergence`, `solvecl_time_seconds`	the lift-matching loop's iteration count, final residual, and run time

H.5 Agent records

`agent_nominations.csv` records every time an agent recorded a recommendation with `campaign.py nominate` (Table 20). The campaign asked for one at the checkpoints 0, 1, 2 and 4, and agents could also record one at any time. The 32-query continuation asked for none, so the agents' standing recommendations at 8, 16 and 32 queries are the last nomination made at or before that many results. `agent_turns.csv` has one row per model session the campaign launched (Table 21). A research session ran until the agent handed over a wing, a review session handled a checkpoint, and a reflection session wrote one reflection. The planning method's planner, evaluator and executor calls were made inside its research sessions and are counted in them. Eleven sessions have no token counts because the provider did not report them.

Table 20: Columns of `agent_nominations.csv`.

Column	Meaning
`episode`, `study`, `method`, `task`	as in `episodes.csv`
`sequence`	the order of the nomination within its episode
`kind`	`checkpoint` when the campaign asked for it, `during_search` when the agent recorded it on its own
`results_seen`	how many online CFD results the agent had seen when it made the nomination
`decision`	the campaign's name for the turn
`wing`	the nominated wing's hash
`recorded_at`	when the campaign recorded it, in UTC

Table 21: Columns of `agent_turns.csv`.

Column	Meaning
`episode`, `study`, `method`, `task`	as in `episodes.csv`
`launch`	the session's number within its episode
`kind`	`research`, `review` or `reflection`
`decision`	the campaign's name for the turn
`wall_seconds`	how long the session ran
`input_tokens`, `output_tokens`, `cache_read_input_tokens`	the token counts the model provider reported, summed over the session's calls
`model_requests`	how many model calls the session made
`cost_usd`	the session's cost at the provider's list prices
`is_error`	whether the session ended with an error
`ended_at`	when it ended, in UTC

H.6 Weight-sensitivity tables

These hold the evidence for Appendix D. `weight_sensitivity_refits.csv` has one row for each of the 103 checkpoint recommendations the weight can change, naming the design that weights 0, 0.5 and 1 each pick, in the columns `episode`, `checkpoint`, `method`, `n_inconclusive`, `pick_w0.0`, `pick_w0.5`, `pick_w1.0` and `same_pick`. `weight_sensitivity_wings.csv` has the four designs the weight chooses between at checkpoint 32, with their 38 variables and CFD outcome, in the columns `episode`, `method`, `task`, `checkpoint`, `weight`, `graded`, `wing`, `delivered`, `ld`, `reasons` and `x00` to `x37`. The campaign graded two of the four designs. The other two were graded afterwards, once the refit

showed which designs a weight of 1 would have recommended, so that the appendix could say whether the weight changes what gets delivered and not only what gets recommended.

H.7 SUPERWING RECORDS

SuperWing is the wing dataset the benchmark is built on. Its complete run records are redistributed in `data/superwing/` with the SuperWing authors' permission. `superwing_records.csv` has one row per wing for all 4,544 wings, with the same 38 variables as `wings.csv`, whether the mesh step worked, and the result of each of the wing's 8 flight conditions. It was built from the complete run record, `running_info.jsonl`, while the public release of SuperWing contains only the converged results. `cruise_ld_percentiles.json` holds the dataset's cruise-band lift-to-drag percentiles. Of the 36,344 tests, 28,856 succeeded. 4,563 failed because the lift kept changing and never settled, 2,788 failed because the simulation blew up, and 137 were never run. The often-used table of 4,239 wings lists only the wings where at least one of the 8 tests succeeded, so it leaves out the 305 wings that failed every test. The column `any_success` is 0 for those 305.

H.8 FURTHER NOTES FROM THE DATA CARD

The SMAC3 records name the reason for each nondelivery but not the outcome type, so SMAC3's `outcome` column is empty, and 18 nondeliveries its run controller recorded without a reason carry `undetermined`. Three SMAC3 evaluations have no wing hash, because their evidence was lost. The agent export left out the scores of the 90 private grades made before the first query and the reason codes of all 136 agent private grades. Both were filled in from the grader's result documents, the reason codes by the rule the export used for the other rows, checked against every evaluation that has both a code and a document. The 39 agent requests that stopped before a document was written keep their recorded outcome and no score. SMAC3's `cfd_core_seconds` and `timed_evaluations` include two checkpoint-32 grades that were replaced when two SMAC3 episodes were run again, and the paper's cost numbers use these totals. The 44 GB of raw CFD output, the optimizers' full logs, and the agents' full execution traces are not included.

I CFD PROTOCOL AND ENVIRONMENT

The grading protocol is the file `configs/cfd/final_protocol_candidate_v2.json`. The configs README describes it as the grading protocol the benchmark ran with, and it pins the grader's source files under `scripts/cfd/` by content. The grader `grade_benchmark.py` builds the mesh of one wing, runs the three flow conditions in series at fixed lift, and writes the result document. `grading_protocol.py` loads the protocol and refuses to run if any pinned file has changed, and `task_protocols.py` derives the low-load and high-load protocols from this file, changing only the lift target and its angle-of-attack seed. The values below are copied from the protocol file, and a field quoted in typewriter type without a plain reading is reproduced as written. The file's own identity fields are `protocol_id superwing-fixed-cl-v5-candidate-c`, `protocol_revision 3`, `protocol_status qualification_required`, `official_grading_enabled false` and `benchmark_release_id null`. Its scope block names the geometry schema `superwing-38d-v1`, one mesh per wing, conditions run one after another, and `partial_scores_official false`. The `geometry_hash_algorithm` is `sha256(superwing-geom38-f64le-v1-NUL + 38xf64le)`, which reads as the SHA-256 of a fixed prefix string, a NUL byte and the 38 coordinates as little-endian 64-bit floats, and it is the `wing` identifier of the released tables.

I.1 FLOW CONDITIONS AND TRIM PROCEDURE

Every task uses the same three conditions and flow reference quantities. The protocol file carries the nominal task, and the other two tasks change only the target lift coefficient of the three conditions and the angle-of-attack seed (Table 22). Each seed is the exact median, with no rounding, of the recorded angle of attack over the successful SuperWing cases whose lift coefficient lies within 0.02 of the target, over all recorded Mach numbers. The nominal seed comes from `configs/cfd/seed_aoa_selection_v1.json`, whose band 0.39 to 0.43 held

1,198 cases, and the low and high seeds from `seed_aoa_selection_load_low_v1.json`, band 0.13 to 0.17 with 994 cases, and `seed_aoa_selection_load_high_v1.json`, band 0.64 to 0.68 with 1,172 cases. All three files state that the seed applies to every wing and condition and that no reselection is allowed. Both load files carry a freeze time of 2026-08-02. The protocol's seed policy is `fixed_per_condition` with `selection_rule` `exact_median_successful_cases_in_cl_0.39_0.43_no_rounding`.

Table 22: The three flow conditions of each task. The Mach numbers are `mach_q1` to `mach_q3`, run and reported in that order. The lift target is `cl_target` of every condition and the seed is `seed_policy.alpha0_deg`, the angle of attack that lift matching starts from. The lift tolerance is the gate `cl_match_abs_less_than_or_equal`. The flow block also sets `reynolds_length` 1.0, `temperature_k` 300.0, `area_ref_source` `generated_wing_surface_area`, which is the surface area of the generated wing, `chord_ref` 1.0 and `x_ref` 0.25.

Task	Lift target	Mach numbers	Reynolds number	Lift tolerance	Seed (degrees)
low	0.15	0.7853, 0.8195, 0.853	20,000,000	0.02	4.4259755152
nominal	0.41	0.7853, 0.8195, 0.853	20,000,000	0.02	6.6401110468
high	0.66	0.7853, 0.8195, 0.853	20,000,000	0.02	8.924059800550001

Each condition is solved at fixed lift. The ADflow runner calls the solver's lift-matching routine `solveCL` with the target lift, the seed angle of attack and the options of the `solver.solvecl` block. The release's patch to the runner makes every option explicit, so that a change of a library default cannot change the protocol, and records the routine's result in the output. The options with a plain reading are `delta 0.5`, the first step in angle of attack in degrees, `tol 0.001`, the lift tolerance the matching loop aims for before the condition gate accepts a lift within 0.02 of the target, `max_iter 20`, at most 20 matching iterations, `clalpha_guess null`, no starting guess of the lift slope, and `write_solution true`. The remaining options are reproduced as written. They are `auto_reset false`, `relax_clalpha 1.0`, `relax_alpha 1.0`, `l2_convergence_rel null`, `stop_on_stall false`, `work_unit_time null` and `update_cutoff 1e-16`.

I.2 Mesh and solver settings

The mesh recipe is the SuperWing dataset default, built once per wing by the SuperWing mesh generator (Table 23). The flow solver is ADflow on the RANS equations, run as 16 MPI ranks with one thread each (Table 24). The protocol requires six environment variables in every ADflow process. The released grader refuses to start when any variable that begins with `ADFLOW_`, or any of the three thread variables, is set in its own environment. It then adds the six values itself, from the protocol, to the environment of every `mpirun` child, and passes only a fixed list of other variables through. The protocol has no iteration limit on a condition beyond `n_cycles` and `n_cycles_coarse`. Its limits per step are wall-clock time, 1800 seconds for meshing and 7200 seconds for each flow condition, with no retries after an infrastructure failure. Those are the grading job's own settings. Retrying a job after a machine failure was the campaign controller's task, and the ten charged requests in Appendix B are the cases where that layer did not retry.

Table 23: The `mesh` block and the mesh time limit.

Field	Value	Meaning
Block `mesh`		
`recipe`	`superwing-dataset-default-v1`	the mesh procedure of the SuperWing dataset
`robust_mesh`	`false`	reproduced as written
`min_volume_valid_if`	`>0`	a mesh is valid only if its smallest cell volume is positive
`min_quality_valid_if`	`>0`	and only if its smallest cell quality is positive
`marginal_quality_if`	`0<q<0.05`	a quality in this range is recorded as marginal
`missing_or_malformed_evidence`	`inconclusive`	a missing or unreadable mesh report makes the mesh step inconclusive
Block `execution`		
`mesh_timeout_seconds`	`1800`	the mesh step is stopped after 30 minutes of wall-clock time

Table 24: Solver switches, execution settings and the required environment variables.

Field	Value	Meaning
Block `solver`		
`equation_type`	`RANS`	Reynolds-averaged Navier Stokes equations
`mg_cycle`	`3w`	the multigrid cycle
`use_ank_solver`	`true`	ADflow's approximate Newton Krylov solver for the early part of the solve
`use_nk_solver`	`true`	ADflow's Newton Krylov solver for the late part of the solve
`nk_switch_tol`	`1e-08`	relative residual at which the solver switches to Newton Krylov
`l2_convergence`	`1e-10`	residual at which a solve counts as converged
`n_cycles_coarse`	`500`	iteration limit on the coarse grids
`n_cycles`	`3000`	iteration limit of the solve
Block `execution`		
`mpi_ranks`	`16`	MPI ranks per job
`omp_num_threads`, `openblas_num_threads`, `mkl_num_threads`	`1`	one thread per rank
`solve_timeout_seconds_per_attempt`	`7200`	each condition attempt is stopped after 2 hours of wall-clock time
`infrastructure_retries`	`0`	no retry after an infrastructure failure
`fresh_attempt_directory`	`true`	every attempt runs in a new directory
`fail_fast_conditions`	`false`	the remaining conditions still run after one fails
`early_stall_abort`	`false`	no early stop on a stalled residual
`timeout_is_scientific_failure`	`false`	a time limit is not counted as a conclusive failure of the wing
`host_lock_path`	`/workspace/.wingbench-locks/np16.lock`	the lock that allows one 16 rank solve per machine
Block `environment`, required in every ADflow process		
`ADFLOW_USE_ANK`	`1`	turn on the approximate Newton Krylov solver
`ADFLOW_USE_NK`	`1`	turn on the Newton Krylov solver
`ADFLOW_NK_SWITCH`	`1e-8`	relative residual of the switch
`OMP_NUM_THREADS`, `OPENBLAS_NUM_THREADS`, `MKL_NUM_THREADS`	`1`	one OpenMP, OpenBLAS and MKL thread per rank
`forbidden_parent_prefixes`	`ADFLOW_`	the grader refuses to start if a variable with this prefix is set in its own environment
`forbidden_parent_names`	the three thread variables	the same rule for these names
`require_non_root`	`true`	the grader refuses to run as root

I.3 Condition gates, outcomes and aggregation

A condition is valid only if every gate of Table 25 holds. The protocol file itself has only the positivity gates on lift, drag and lift-to-drag ratio. The upper limit of 40 on each lift-to-drag ratio is part of the benchmark objective in the rules, and the released local grader applies it as the $(0, 40]$ envelope. The rules state the delivery condition in one sentence. A CFD evaluation delivers a score only if all three conditions return a valid result, the lift is within 0.02 of the target, every lift-to-drag ratio is above 0 and at most 40, and the angle of attack is between -10 and 25 degrees. Otherwise the evaluation returns a failure type and no score. The failing gate is recorded as a reason code.

Table 25: The `condition_gates` block.

Field	Value	Meaning
`process_returncode_zero_required`	`true`	the solver process must exit with code 0
`solvecl_result_required`	`true`	the lift-matching routine must return a result
`solvecl_converged_required`	`true`	and it must report convergence
`aeroproblem_solve_failed_must_equal`	`false`	ADflow's solve-failed flag must be false
`aeroproblem_fatal_fail_must_equal`	`false`	ADflow's fatal-fail flag must be false
`err_cl_strictly_less_than`	`0.0005`	the lift oscillation over the last iterations must stay below this value
`cl_match_abs_less_than_or_equal`	`0.02`	the lift must be within this distance of the target
`aoa_deg_inclusive`	$[-10, 25]$	the angle of attack must lie in this range, in degrees
`positive_cl_required`, `positive_cd_required`, `positive_ld_required`	`true`	lift, drag and lift-to-drag ratio must be positive

Table 26 gives the `aggregation` and `outcome_policy` blocks. A wing passes only when all three conditions are valid, and its score is the geometric mean of the three lift-to-drag ratios. Per-condition values of a failed wing are kept as diagnostics and never scored. The released tables

and the local grader report four outcome types. pass means the wing delivered. partial means some but not all flight conditions gave a result. undetermined means no condition gave a result. conclusive_failure means the geometry was rejected. The rules separate failures of the machine from failures of the wing. If a worker is lost, a job is killed by the system, or a result file is never written, the evaluation is retried and does not cost a query. Failures of the wing itself are outcomes and cost the query. They are a rejected geometry, a mesh generator that stops without a mesh, a flow condition that reaches its time limit, and a solver process that stops with an error. A wing that was already evaluated in the same episode returns the recorded result again and still costs a query. The result_contract block requires an atomic write of the result document with a copy of the protocol, the protocol hash, the geometry hash and every attempt preserved, under schema_version wingbench-cfd-whole-wing-result-v1, and states official_requires_protocol_status frozen.

Table 26: The aggregation and outcome_policy blocks. Values without a plain reading are reproduced as written.

Field	Value
Block aggregation	
required_valid_conditions	3
admissibility_pass	all_conditions_valid
admissibility_fail	any_conclusive_condition_invalid_or_unmeshable
admissibility_undetermined	no_conclusive_invalidity_and_any_inconclusive_evidence
score	geometric_mean_ld, the geometric mean of the three lift-to-drag ratios
score_requires_complete_pass	true, no score unless all three conditions are valid
diagnostic_partial_metrics_allowed	true, per-condition values of a failed wing are kept
Block outcome_policy	
timeout	inconclusive_timeout
process_error	inconclusive_infrastructure
missing_output, malformed_output, missing_convergence_evidence	inconclusive_output_failure
completed_solvecl_nonconvergence	invalid_solver_nonconvergence, a matching loop that finished without converging is a conclusive invalid result
uninterrupted_infrastructure_failure	terminal_inconclusive_no_retry
verified_spot_reclaim	censored_external_reacquisition_not_scientific_attempt

I.4 Execution environment

The recorded results come from AWS m7a.4xlarge machines with 16 vCPUs and 64 GB of memory. One wing was graded per machine at a time, and each CFD job used all 16 cores as 16 MPI ranks with one thread each and had the full 64 GB. The grader enforces one job per machine with the lock file of Table 24. The three conditions run one after another, so one wing can take up to six and a half hours. A job that reaches a limit is stopped and recorded as inconclusive. It gets no score, and the wing does not count as delivered. On a slower machine, on a machine where two ranks share one core, or with less memory so that the solver swaps, the same wing takes longer and more wings reach the limits. The rules note that about one third of the reference nondeliveries were set by a time limit, require an m7a.4xlarge or a faster machine for comparisons with the reference results, and state that a later version will replace the time limits with iteration limits.

I.5 Software versions and pinned digests

The CFD workers ran inside the public ADflow container image published by the MDO Lab, mdolab/public on Docker Hub, from the u20-gcc-ompi-stable line, named by digest (Table 27). The base image was built in June 2025. It contains Ubuntu 20.04, gcc, Open MPI 4.1.6, PETSc, CGNS 4.2.0, Python 3.9.12 installed through pyenv, numpy, scipy, mpi4py, petsc4py, and the MDO Lab packages ADflow, pyHyp, cgnsUtilities, baseclasses, pyGeo, IDWarp and pySpline. The build history of the image shows numpy 1.21.5, scipy 1.7.3 and mpi4py 3.1.5 being installed. The MDO Lab packages were taken from their main branches at build time, and the digest pins their exact versions. The image exists only for amd64, has one account of its own with user id 1000, and every command runs as that account. The SuperWing mesh generator builds the wing surface with the library cst-modeling3d, placed on PYTHONPATH as a git checkout at a pinned commit, which the grader's runtime probe reads from the checkout and compares with the protocol. The environment document names the library's source repository and the

pinned commit. The surrogate is the L size surface flow AeroTransformer, called `ATsurf_L`. It runs outside the container, on the optimizer side, on the CPU, in float32. The benchmark protocol `configs/benchmark/multifidelity_benchmark_protocol_v0.json` names the model and pins a binding file, `configs/benchmark/aerotransformer_l_v1.json`, by its SHA-256. The binding file is not in the release, and the weight hashes in Table 27 come from a copy of it whose hash matches the pin. The environment document states that on 2026-09-25 both file hashes matched the files in the public weights repository at the pinned revision, that the revision was the head of that repository, and that the container tag still pointed to the pinned platform list.

Table 27: Pinned digests, commits and hashes. Long values are broken across lines but are complete.

Item	Value
Base image for linux/amd64	`mdolab/public@sha256:fb6d20250ad192b6034c3e61286e6179f4854eb58e3fdb25d98dd5844fcb42fd`
Platform list the tag pointed to	`sha256:68393a976907d6306f8b2d87ba95365a8720c3b2a5e55ae431f3cb05d5198223`
Base image download size	about 2.7 GB in 79 compressed layers
Image that graded the baseline study, `final_child_image_linux_amd64_digest`	`sha256:53a214a5c6ea52287b08656da5e37041 1323aca64b60e7ab0b13992eb21f8c44`. It lived in a private registry and is not public
Image recorded by the agent study's result documents	`sha256:426f5b1e8758bb48be843231a7040e53 9ad8a8e0ae5acd6928fbc06a26fa067f`, a newer build of the same recipe with the same base image, library commit and solver settings
`cst-modeling3d` commit	`9500b19a732463c26b32f6860ac6c517c7d212a5`
Location of the checkout in the container, also `PYTHONPATH`	`/home/mdolabuser/repos/cst-modeling3d`
`expected_runtime_fingerprint_sha256`	`60263209c65d73810a768021b8043ced 6038a9c55b0c7feaf37c82168e73314d`, the hash of the complete report of `scripts/cfd/runtime_probe.py` on the recorded machines
SuperWing simulation files	four files from the `simulation` folder of the public AeroTransformer code repository at commit `3dc350ff69e354d1451eae468c686a368bd6151f`, three used unchanged and `run-adflow.py` used with the patch `scripts/cfd/run-adflow.wingbench.patch` (Table 28)
Surrogate weights	`ATsurf_L/best_model_weights`, 58,218,804 bytes, SHA-256 `4fa8fb4e3986cdb7a4a24f1f74410c54 412beb5191857c76fdd4c4f0c5dc8bfa`
Surrogate model config	`ATsurf_L/model_config`, 310 bytes, SHA-256 `a6cedb0a3dba73f99c41bcfdf3913d37 a912c6b0558d5f906e6633aebdca8ef2`
Surrogate weights repository revision	`698d70095a00d6f4a25f870e8de0772fc12b68f7`, in the public AeroTransformer weights repository
Surrogate model code	the public AeroTransformer code repository at commit `3dc350ff69e354d1451eae468c686a368bd6151f`
Surrogate binding file pin	`aerotransformer_l_v1.json`, SHA-256 `c18b1c9e5535ec0bf38ec1178aa16d46 ab9b599cbaa71f6ce5c7eb395e6220d4`

The grader checks the four SuperWing files against the protocol before it runs anything and refuses to run if one differs. It then copies the scripts into each case directory under adapted names and checks those copies too, against the `generated_case_file_sha256` block (Table 28). The script `scripts/cfd/fetch_superwing_scripts.py` downloads the four files at the pinned commit, applies the patch, and checks every file against the protocol. In all, the `scientific_files` block pins 24 files by SHA-256, namely eighteen grader modules under `scripts/cfd/`, the four SuperWing files, and the two configuration files `configs/cfd/publication_storage_binding_v1.json`, which names the object store the results were published to with a compliance object lock and a minimum retention of 90 days, and `configs/cfd/seed_aoa_selection_v1.json`. The environment document's verification recipe hashes all 24 files and reports that they matched on 2026-09-25.

Table 28: The four SuperWing files as pinned in the repository, and the hashes of their copies in a case directory. The tip geometry is the same file in both places.

File	SHA-256
In `scripts/cfd/aerotransformer/`	
`gen-mesh.superwing.py`	`7daf656cdb23c6de32fba1585e22bb850d1b7e86c2ad7d031239989f2d1f73b4`
`run-adflow.py` (patched)	`c4f2bb3c04aa21a5a07b5c51a3612e4cd04f7bcc108894e953dd2c6506a86ad6`
`single-point.py`	`3682aad55ebc051121e0aa373305d0341bbde1a57642fc3f8981b84ffbd15f27`
`original_tip.xyz`	`c93c1e2c74da73c9610d5c77e81b2f1d9bc0af53df04e1d608aaa61cda3e7db1`
Copies in a case directory, `generated_case_file_sha256`	
`gen-mesh.py`	`8eb7d505c85b891a1ac1c547189b3399b38cb2a504ebac1a5b8fa97152ea2997`

File	SHA-256
run-adflow.py	6c41807d95cec2b46841d5a5e8466f36004f4474177402641df884e6db7e986b
single-point.py	eee1888640c6947dd8886a31d30a8a39628a74c0353ea455d9b581c306434976
original_tip.xyz	c93c1e2c74da73c9610d5c77e81b2f1d9bc0af53df04e1d608aaa61cda3e7db1

I.6 What a rebuild reproduces

The release rebuilds the container from the public base image with a Dockerfile and an Apptainer definition in `benchmark/environment/`. On 2026-09-25 the Docker image was built from that folder on a new `m7a.4xlarge`, the four SuperWing files were fetched, and `benchmark/grade_local.py` graded one recorded wing. All three conditions returned lift, drag, angle of attack and lift-to-drag ratio identical to the recorded values, the score equaled the recorded 33.401823323619524, and the grade took 2 hours 13 minutes. The folder `rebuilt_test/` holds that result document, the grader's log, the fetch log and the image digest of the rebuilt container. The environment document lists four things the rebuild does not reproduce. First, the exact image that graded the wings, since the rebuilt image has the same content but a different digest. Second, the runtime fingerprint. The probe report includes the host kernel version, the path of every module, the account ids and the values of `PATH` and similar variables, so a rebuilt environment produces a different report and the released grader stops with a fingerprint mismatch. To grade with a rebuilt environment one must run the probe, write its hash into a copy of the protocol, and treat that copy as a new protocol revision. Results graded under it are the user's own and not the recorded results, and the tested rebuild therefore carries `grading_mode local` in its result document. Third, the grader's other inputs. `grade_benchmark.py` also requires an authorization file, a runtime attestation file and a publication journal, and the tools that produced the first two ran on the recorded machines and are not in the release. Fourth, the surrogate, which runs outside the container.

I.7 Settings files

The configs README lists the settings files the benchmark ran with (Table 29). The files record each other's fingerprints, and the code checks those fingerprints when it loads them. The released episodes follow `benchmark/RULES.md`, version 1.0, with 32 CFD queries and 4,096 surrogate evaluations per episode.

Table 29: Settings files under `configs/`, as the configs README describes them.

File	Role
cfd/final_protocol_candidate_v2.json	the grading protocol, which pins the grader's source files under `scripts/cfd/` by content
cfd/seed_aoa_selection_*.json	how the angle-of-attack seed of each lift target was chosen from the dataset
benchmark/wingbench_v2_candidate_protocol_v1.json	the episode protocol the optimizers ran under, which verifies the lineage files beside it
benchmark/superwing_physical_normalization_v1.json	maps the 38 standardized design variables to physical values
benchmark/undefended_50_wing_corpus_v1.json and the three corpus50_* files	the fixed warm-start context every surrogate-assisted episode begins from
optimizer/wingbench_v2_phase3_method_panel_v1.json	fixes the four surrogate-assisted baselines and their settings
benchmark/wingbench_v2_phase3_pilot_plan_v1.json	the development plan the method panel binds to. The panel does not load without it
benchmark/wingbench_multifidelity_benchmark_v1_release.json	the frozen July 2026 definition of the interface that the candidate protocol binds by fingerprint. Its four-query cap and 512-call matched track describe the development-time characterization panel

J The best delivered wings

This appendix shows the best wing each method delivered. For the five baseline optimizers, we took the recommendation each method made after 32 queries in each of its ten episodes per task and kept the delivered one with the highest L/D. The open-ended research agent made no further requested checkpoint recommendation after query 4, so for it we took the delivered online design with the highest L/D among queries 5 to 32 of its 30 extension episodes. Figure 6 draws the 16 wings at one scale, and Table 30 lists their planform values. CMA-ES with no feedback and CMA-ES with selection only each delivered no recommendation after 32 queries at $C_L = 0.15$, so those two cells are empty. CMA-ES with no feedback and CMA-ES with selection only delivered the same wing

at $C_L = 0.66$. The best wings of the different methods are all thin and swept, with a high aspect ratio. Their leading-edge sweep ranges from 26 to 44 degrees, their aspect ratio from 8.9 to 12.5, their taper ratio from 0.26 to 0.52, and their root thickness ratio from 0.126 to 0.138. The highest score at each load is 19.31 for open-ended research at $C_L = 0.15$, 34.05 for open-ended research at $C_L = 0.41$ and 32.37 for feedback-guided CMA-ES at $C_L = 0.66$. Every wing shown was delivered by the CFD evaluator, so each score is a graded result rather than a surrogate prediction.

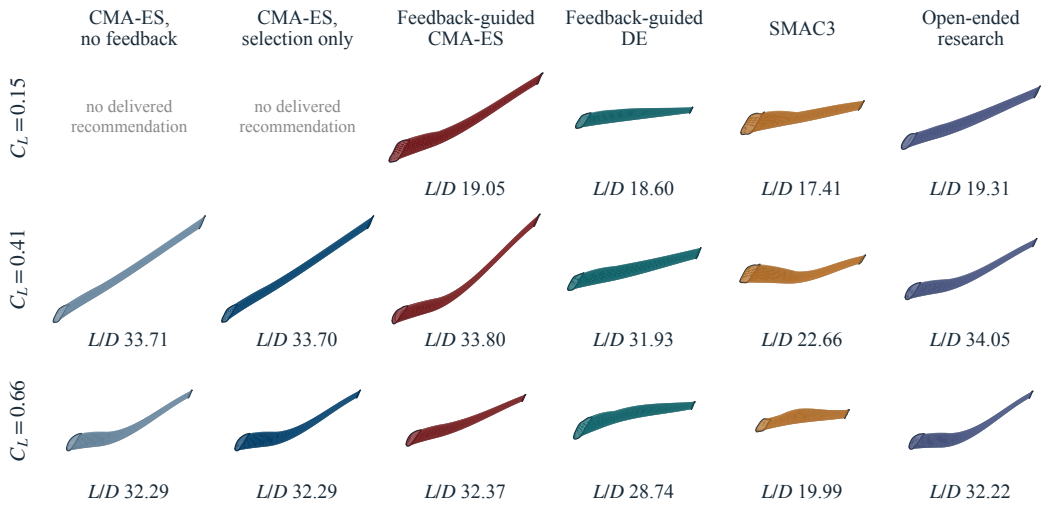

Figure 6: The best delivered wing of each method and task, drawn as in Figure 1 at one scale. For the five baselines it is the delivered recommendation after 32 queries with the highest L/D over the ten seeds. For open-ended research it is the delivered online design with the highest L/D among queries 5 to 32 of its 30 extension episodes. Each number is the wing's L/D. The two archive-only CMA-ES methods delivered nothing after 32 queries at $C_L = 0.15$, so those cells are empty.

Table 30: The wings of Figure 6. Seed is the episode seed, and for open-ended research the episode label. Sweep is the leading-edge sweep in degrees, taper ratio is the tip chord over the root chord, and root t/c is the thickness of the root section over its chord. The four are columns 0, 3, 4 and 37 of the 38 design variables, in the order of the released tables and the guide, and they are the planform values the guide defines without an interpolation rule. A dash marks a cell with no delivery.

method	C_L	seed	L/D	sweep (°)	aspect ratio	taper ratio	root t/c
CMA-ES, no feedback	0.15	–	–	–	–	–	–
	0.41	6206	33.71	39.9	12.49	0.524	0.126
	0.66	6208	32.29	37.7	12.49	0.311	0.127
CMA-ES, selection only	0.15	–	–	–	–	–	–
	0.41	6206	33.70	39.8	12.49	0.524	0.127
	0.66	6208	32.29	37.7	12.49	0.311	0.127
Feedback-guided CMA-ES	0.15	6202	19.05	43.6	12.27	0.524	0.138
	0.41	6201	33.80	37.1	12.49	0.524	0.126
	0.66	6209	32.37	36.1	12.49	0.272	0.126
Feedback-guided DE	0.15	6207	18.60	40.9	11.87	0.257	0.130
	0.41	6203	31.93	37.8	12.32	0.407	0.131
	0.66	6204	28.74	38.9	12.15	0.292	0.128
SMAC3	0.15	6202	17.41	39.8	10.54	0.475	0.129
	0.41	6202	22.66	26.1	11.74	0.524	0.126
	0.66	6204	19.99	36.7	8.90	0.356	0.135
Open-ended research	0.15	9531	19.31	39.5	11.52	0.524	0.128
	0.41	9551	34.05	39.3	12.45	0.357	0.131
	0.66	9531	32.22	39.0	12.46	0.283	0.133

K FULL RESULTS TABLES

The tables in this appendix give the full results per method, task and episode. They are computed from the released data tables by `analysis/17_appendix_tables.py`.

Table 31 gives, for every baseline method, task and checkpoint, how many of the ten recommended wings delivered a CFD score and the mean L/D of those that did.

Table 31: Baseline recommendations at every checkpoint. A checkpoint is the number of online CFD queries the method had made when it recommended the wing, so checkpoint 0 is the recommendation made before any CFD feedback. For each method and task, n is the number of the ten episodes whose recommended wing delivered a CFD score, and L/D is their mean score.

method	online CFD queries before the recommendation													
	0		1		2		4		8		16		32	
	n	L/D	n	L/D	n	L/D	n	L/D	n	L/D	n	L/D	n	L/D
$C_L = 0.15$														
CMA-ES, no feedback	0	–	0	–	0	–	0	–	0	–	0	–	0	–
CMA-ES, selection only	0	–	0	–	0	–	0	–	0	–	0	–	0	–
Feedback-guided CMA-ES	3	16.17	5	15.42	4	16.47	5	16.25	5	16.25	5	16.87	6	16.76
Feedback-guided DE	3	13.13	4	12.38	6	14.02	8	14.78	10	16.10	10	17.18	10	17.52
SMAC3	1	6.33	3	8.87	3	8.87	6	7.48	7	9.51	9	9.87	9	10.56
$C_L = 0.41$														
CMA-ES, no feedback	3	33.19	3	33.19	3	33.19	3	33.22	3	33.22	4	33.19	4	33.22
CMA-ES, selection only	3	33.19	4	33.14	4	33.14	4	33.18	4	33.19	4	33.21	4	33.22
Feedback-guided CMA-ES	3	23.58	5	22.42	6	25.82	10	28.73	10	32.06	10	32.91	10	33.04
Feedback-guided DE	3	19.47	5	18.79	7	20.19	9	22.35	10	25.40	10	28.94	10	31.09
SMAC3	1	10.24	3	13.58	3	13.58	5	12.03	6	15.03	6	15.91	7	17.00
$C_L = 0.66$														
CMA-ES, no feedback	4	32.03	4	32.03	4	32.03	5	32.04	5	32.08	5	32.10	5	32.12
CMA-ES, selection only	4	32.03	4	32.03	5	32.04	5	32.07	5	32.10	5	32.12	5	32.12
Feedback-guided CMA-ES	7	16.65	6	17.89	7	18.85	10	23.99	10	28.14	10	31.00	10	31.64
Feedback-guided DE	1	17.23	5	15.20	7	15.44	8	18.15	9	19.94	10	23.59	10	27.05
SMAC3	1	5.66	2	9.17	2	9.17	3	8.45	4	10.43	5	12.19	6	12.82

Table 32 gives the same for the three agent methods at the checkpoints of the four-query comparison.

Table 32: Agent recommendations at checkpoints 0, 1, 2 and 4, in the format of Table 31.

method	online CFD queries before the recommendation							
	0		1		2		4	
	n	L/D	n	L/D	n	L/D	n	L/D
$C_L = 0.15$								
Open-ended research	2	16.79	7	16.44	8	16.95	9	17.15
Planning	8	16.64	10	16.64	10	17.01	10	17.41
Reflection	6	16.47	8	16.59	10	16.98	10	17.33
$C_L = 0.41$								
Open-ended research	7	29.12	9	28.66	9	29.79	10	31.09
Planning	7	28.19	9	28.28	9	28.43	9	29.31
Reflection	8	28.54	10	28.42	10	29.05	10	29.83
$C_L = 0.66$								
Open-ended research	9	25.58	10	24.92	10	26.20	10	27.83
Planning	8	26.09	8	26.09	9	26.61	9	27.59
Reflection	9	24.42	10	24.53	10	25.75	10	26.79

Table 33 counts the online CFD queries of each method and task that delivered a score.

Table 33: Online CFD delivery per method and task. Delivered is the number of online queries that returned a valid score, total is the number of online queries, and rate is their ratio. The agent rows cover queries 1 to 4 of all 90 agent episodes. The extension row covers all 32 queries of the 30 open-ended research episodes that continued, so it counts their first four queries a second time.

method	$C_L = 0.15$			$C_L = 0.41$			$C_L = 0.66$		
	delivered	total	rate	delivered	total	rate	delivered	total	rate
Baselines, 32 queries in each of 150 episodes									
CMA-ES, no feedback	0	320	0.00	83	320	0.26	118	320	0.37
CMA-ES, selection only	0	320	0.00	85	320	0.27	112	320	0.35
Feedback-guided CMA-ES	14	320	0.04	246	320	0.77	247	320	0.77
Feedback-guided DE	93	320	0.29	200	320	0.62	199	320	0.62
SMAC3	114	320	0.36	88	320	0.28	49	320	0.15
Agents, queries 1 to 4 of all 90 episodes									
Open-ended research	21	40	0.53	28	40	0.70	34	40	0.85
Planning	31	40	0.78	30	40	0.75	29	40	0.72
Reflection	30	40	0.75	29	40	0.72	37	40	0.93
Extension, all 32 queries of the 30 continued episodes									
Open-ended research	198	320	0.62	225	320	0.70	197	320	0.62

Table 34 sums the CFD processor time of each method over the evaluations with a recorded run time.

Table 34: CFD processor time per method. A core-hour is one hour of one processor core, and each CFD job used 16 cores. The totals cover the evaluations with a recorded run time. The SMAC3 total includes two checkpoint-32 grades that were replaced when two SMAC3 episodes were run again.

method	episodes	timed evaluations	core-hours	core-hours per evaluation
Baselines				
CMA-ES, no feedback	30	1,123	16,689	14.9
CMA-ES, selection only	30	1,119	16,584	14.8
Feedback-guided CMA-ES	30	1,049	38,759	36.9
Feedback-guided DE	30	1,034	38,064	36.8
SMAC3	30	954	35,307	37.0
All baselines	150	5,279	145,403	27.5
Agents				
Open-ended research	30	968	42,113	43.5
Planning	30	162	6,112	37.7
Reflection	30	167	6,392	38.3
All agents	90	1,297	54,617	42.1
Both studies	240	6,576	200,019	30.4

Table 35 splits the same online queries by the grader's outcome type.

Table 35: Outcome types of the online CFD queries, in the groups of Table 33. A pass delivered a score. A partial outcome has a result at some but not all flight conditions. An undetermined outcome has no result at any condition. A conclusive failure is a rejected geometry. The SMAC3 records name the reason for each nondelivery but not the outcome type, so its type counts are empty.

method	C_L	pass	partial	undetermined	conclusive failure	total
Baselines, 32 queries in each of 150 episodes						
CMA-ES, no feedback	0.15	0	0	308	12	320
	0.41	83	20	178	39	320
	0.66	118	36	119	47	320
CMA-ES, selection only	0.15	0	0	308	12	320
	0.41	85	21	177	37	320
	0.66	112	43	120	45	320
Feedback-guided CMA-ES	0.15	14	15	230	61	320
	0.41	246	44	17	13	320
	0.66	247	37	35	1	320
Feedback-guided DE	0.15	93	33	135	59	320
	0.41	200	47	53	20	320
	0.66	199	49	61	11	320
SMAC3	0.15	–	–	–	–	320
	0.41	–	–	–	–	320
	0.66	–	–	–	–	320
Agents, queries 1 to 4 of all 90 episodes						
Open-ended research	0.15	21	9	5	5	40
	0.41	28	6	5	1	40
	0.66	34	0	3	3	40
Planning	0.15	31	6	2	1	40
	0.41	30	6	4	0	40
	0.66	29	0	10	1	40
Reflection	0.15	30	5	3	2	40
	0.41	29	5	5	1	40
	0.66	37	0	1	2	40
Extension, all 32 queries of the 30 continued episodes						
Open-ended research	0.15	198	81	27	14	320
	0.41	225	53	34	8	320
	0.66	197	56	51	16	320

Table 36 lists all 240 episodes.

Table 36: All 240 episodes. The online columns count the episode's online CFD queries, how many delivered a score, and the best delivered L/D. The final recommendation is the one made at the last checkpoint, after 32 queries for the baselines and after 4 for the agents. The 30 open-ended research episodes continued to 32 online queries, and their online columns cover all 32.

				online CFD queries			final recommendation	
episode	method	C_L	seed	total	delivered	best L/D	delivered	L/D
B000	CMA-ES, no feedback	0.15	6200	32	0	–	no	–
B012	CMA-ES, no feedback	0.15	6201	32	0	–	no	–
B024	CMA-ES, no feedback	0.15	6202	32	0	–	no	–
B036	CMA-ES, no feedback	0.15	6203	32	0	–	no	–
B048	CMA-ES, no feedback	0.15	6204	32	0	–	no	–
B060	CMA-ES, no feedback	0.15	6205	32	0	–	no	–
B072	CMA-ES, no feedback	0.15	6206	32	0	–	no	–
B084	CMA-ES, no feedback	0.15	6207	32	0	–	no	–
B096	CMA-ES, no feedback	0.15	6208	32	0	–	no	–
B108	CMA-ES, no feedback	0.15	6209	32	0	–	no	–
B004	CMA-ES, no feedback	0.41	6200	32	0	–	no	–
B016	CMA-ES, no feedback	0.41	6201	32	0	–	no	–
B028	CMA-ES, no feedback	0.41	6202	32	0	–	no	–
B040	CMA-ES, no feedback	0.41	6203	32	8	33.09	yes	33.09
B052	CMA-ES, no feedback	0.41	6204	32	30	32.77	yes	32.77
B064	CMA-ES, no feedback	0.41	6205	32	0	–	no	–
B076	CMA-ES, no feedback	0.41	6206	32	14	33.71	yes	33.71
B088	CMA-ES, no feedback	0.41	6207	32	0	–	no	–
B100	CMA-ES, no feedback	0.41	6208	32	31	33.28	yes	33.28
B112	CMA-ES, no feedback	0.41	6209	32	0	–	no	–
B008	CMA-ES, no feedback	0.66	6200	32	29	32.10	yes	32.10
B020	CMA-ES, no feedback	0.66	6201	32	21	31.91	yes	31.91
B032	CMA-ES, no feedback	0.66	6202	32	0	–	no	–
B044	CMA-ES, no feedback	0.66	6203	32	0	–	no	–

continued on the next page

				online CFD queries			final recommendation	
episode	method	C_L	seed	total	delivered	best L/D	delivered	L/D
B056	CMA-ES, no feedback	0.66	6204	32	8	32.16	yes	32.16
B068	CMA-ES, no feedback	0.66	6205	32	0	–	no	–
B080	CMA-ES, no feedback	0.66	6206	32	0	–	no	–
B092	CMA-ES, no feedback	0.66	6207	32	31	32.15	yes	32.15
B104	CMA-ES, no feedback	0.66	6208	32	29	32.29	yes	32.29
B116	CMA-ES, no feedback	0.66	6209	32	0	–	no	–
B001	CMA-ES, selection only	0.15	6200	32	0	–	no	–
B013	CMA-ES, selection only	0.15	6201	32	0	–	no	–
B025	CMA-ES, selection only	0.15	6202	32	0	–	no	–
B037	CMA-ES, selection only	0.15	6203	32	0	–	no	–
B049	CMA-ES, selection only	0.15	6204	32	0	–	no	–
B061	CMA-ES, selection only	0.15	6205	32	0	–	no	–
B073	CMA-ES, selection only	0.15	6206	32	0	–	no	–
B085	CMA-ES, selection only	0.15	6207	32	0	–	no	–
B097	CMA-ES, selection only	0.15	6208	32	0	–	no	–
B109	CMA-ES, selection only	0.15	6209	32	0	–	no	–
B005	CMA-ES, selection only	0.41	6200	32	0	–	no	–
B017	CMA-ES, selection only	0.41	6201	32	0	–	no	–
B029	CMA-ES, selection only	0.41	6202	32	0	–	no	–
B041	CMA-ES, selection only	0.41	6203	32	10	33.11	yes	33.11
B053	CMA-ES, selection only	0.41	6204	32	30	32.79	yes	32.79
B065	CMA-ES, selection only	0.41	6205	32	0	–	no	–
B077	CMA-ES, selection only	0.41	6206	32	13	33.70	yes	33.70
B089	CMA-ES, selection only	0.41	6207	32	0	–	no	–
B101	CMA-ES, selection only	0.41	6208	32	32	33.28	yes	33.28
B113	CMA-ES, selection only	0.41	6209	32	0	–	no	–
B009	CMA-ES, selection only	0.66	6200	32	29	32.09	yes	32.09
B021	CMA-ES, selection only	0.66	6201	32	20	31.89	yes	31.89
B033	CMA-ES, selection only	0.66	6202	32	0	–	no	–
B045	CMA-ES, selection only	0.66	6203	32	0	–	no	–
B057	CMA-ES, selection only	0.66	6204	32	6	32.16	yes	32.16
B069	CMA-ES, selection only	0.66	6205	32	0	–	no	–
B081	CMA-ES, selection only	0.66	6206	32	0	–	no	–
B093	CMA-ES, selection only	0.66	6207	32	29	32.15	yes	32.15
B105	CMA-ES, selection only	0.66	6208	32	28	32.29	yes	32.29
B117	CMA-ES, selection only	0.66	6209	32	0	–	no	–
B002	Feedback-guided CMA-ES	0.15	6200	32	1	15.37	yes	15.37
B014	Feedback-guided CMA-ES	0.15	6201	32	1	14.95	yes	14.95
B026	Feedback-guided CMA-ES	0.15	6202	32	3	19.05	yes	19.05
B038	Feedback-guided CMA-ES	0.15	6203	32	1	16.17	yes	16.17
B050	Feedback-guided CMA-ES	0.15	6204	32	0	–	no	–
B062	Feedback-guided CMA-ES	0.15	6205	32	0	–	no	–
B074	Feedback-guided CMA-ES	0.15	6206	32	7	17.17	yes	17.17
B086	Feedback-guided CMA-ES	0.15	6207	32	0	–	no	–
B098	Feedback-guided CMA-ES	0.15	6208	32	0	–	no	–
B110	Feedback-guided CMA-ES	0.15	6209	32	1	17.82	yes	17.82
B006	Feedback-guided CMA-ES	0.41	6200	32	27	32.60	yes	32.60
B018	Feedback-guided CMA-ES	0.41	6201	32	27	33.80	yes	33.80
B030	Feedback-guided CMA-ES	0.41	6202	32	21	33.12	yes	33.12
B042	Feedback-guided CMA-ES	0.41	6203	32	29	32.61	yes	32.61
B054	Feedback-guided CMA-ES	0.41	6204	32	28	33.34	yes	33.34
B066	Feedback-guided CMA-ES	0.41	6205	32	22	33.49	yes	33.49
B078	Feedback-guided CMA-ES	0.41	6206	32	27	33.52	yes	33.52
B090	Feedback-guided CMA-ES	0.41	6207	32	11	32.27	yes	32.27
B102	Feedback-guided CMA-ES	0.41	6208	32	27	33.61	yes	33.61
B114	Feedback-guided CMA-ES	0.41	6209	32	27	32.04	yes	32.04
B010	Feedback-guided CMA-ES	0.66	6200	32	27	31.68	yes	31.68
B022	Feedback-guided CMA-ES	0.66	6201	32	17	30.34	yes	30.34
B034	Feedback-guided CMA-ES	0.66	6202	32	27	31.84	yes	31.84
B046	Feedback-guided CMA-ES	0.66	6203	32	28	31.87	yes	31.87
B058	Feedback-guided CMA-ES	0.66	6204	32	29	31.81	yes	31.81
B070	Feedback-guided CMA-ES	0.66	6205	32	18	31.41	yes	31.41
B082	Feedback-guided CMA-ES	0.66	6206	32	12	30.81	yes	30.81
B094	Feedback-guided CMA-ES	0.66	6207	32	28	32.27	yes	32.27
B106	Feedback-guided CMA-ES	0.66	6208	32	30	31.96	yes	31.96
B118	Feedback-guided CMA-ES	0.66	6209	32	31	32.37	yes	32.37
B003	Feedback-guided DE	0.15	6200	32	8	18.15	yes	18.15
B015	Feedback-guided DE	0.15	6201	32	6	17.58	yes	17.58
B027	Feedback-guided DE	0.15	6202	32	7	17.37	yes	17.37
B039	Feedback-guided DE	0.15	6203	32	11	18.15	yes	18.15
B051	Feedback-guided DE	0.15	6204	32	9	17.93	yes	17.93
B063	Feedback-guided DE	0.15	6205	32	3	16.41	yes	16.41

continued on the next page

				online CFD queries			final recommendation	
episode	method	C_L	seed	total	delivered	best L/D	delivered	L/D
B075	Feedback-guided DE	0.15	6206	32	11	16.43	yes	16.43
B087	Feedback-guided DE	0.15	6207	32	17	18.60	yes	18.60
B099	Feedback-guided DE	0.15	6208	32	14	16.55	yes	16.55
B111	Feedback-guided DE	0.15	6209	32	7	17.98	yes	17.98
B007	Feedback-guided DE	0.41	6200	32	20	30.97	yes	30.97
B019	Feedback-guided DE	0.41	6201	32	26	31.53	yes	31.53
B031	Feedback-guided DE	0.41	6202	32	18	29.96	yes	29.96
B043	Feedback-guided DE	0.41	6203	32	24	31.93	yes	31.93
B055	Feedback-guided DE	0.41	6204	32	21	30.86	yes	30.86
B067	Feedback-guided DE	0.41	6205	32	18	30.18	yes	30.18
B079	Feedback-guided DE	0.41	6206	32	20	31.79	yes	31.79
B091	Feedback-guided DE	0.41	6207	32	23	31.81	yes	31.81
B103	Feedback-guided DE	0.41	6208	32	16	31.75	yes	31.75
B115	Feedback-guided DE	0.41	6209	32	14	30.10	yes	30.10
B011	Feedback-guided DE	0.66	6200	32	21	24.92	yes	24.92
B023	Feedback-guided DE	0.66	6201	32	24	28.01	yes	28.01
B035	Feedback-guided DE	0.66	6202	32	19	27.23	yes	27.23
B047	Feedback-guided DE	0.66	6203	32	20	28.38	yes	28.38
B059	Feedback-guided DE	0.66	6204	32	23	28.74	yes	28.74
B071	Feedback-guided DE	0.66	6205	32	25	28.57	yes	28.57
B083	Feedback-guided DE	0.66	6206	32	8	22.80	yes	22.80
B095	Feedback-guided DE	0.66	6207	32	22	28.13	yes	28.13
B107	Feedback-guided DE	0.66	6208	32	21	26.49	yes	26.49
B119	Feedback-guided DE	0.66	6209	32	16	27.21	yes	27.21
B120	SMAC3	0.15	6200	32	0	–	no	–
B121	SMAC3	0.15	6201	32	20	14.15	yes	14.15
B122	SMAC3	0.15	6202	32	17	17.41	yes	17.41
B123	SMAC3	0.15	6203	32	17	7.48	yes	7.48
B124	SMAC3	0.15	6204	32	14	13.07	yes	13.07
B125	SMAC3	0.15	6205	32	11	6.61	yes	6.61
B126	SMAC3	0.15	6206	32	1	4.16	yes	4.16
B127	SMAC3	0.15	6207	32	3	5.24	yes	5.24
B128	SMAC3	0.15	6208	32	16	10.43	yes	10.43
B129	SMAC3	0.15	6209	32	15	16.54	yes	16.54
B130	SMAC3	0.41	6200	32	0	–	no	–
B131	SMAC3	0.41	6201	32	0	–	no	–
B132	SMAC3	0.41	6202	32	19	22.66	yes	22.66
B133	SMAC3	0.41	6203	32	14	10.78	yes	10.78
B134	SMAC3	0.41	6204	32	18	20.89	yes	20.89
B135	SMAC3	0.41	6205	32	17	16.05	yes	16.05
B136	SMAC3	0.41	6206	32	3	15.37	yes	15.37
B137	SMAC3	0.41	6207	32	0	–	no	–
B138	SMAC3	0.41	6208	32	7	12.85	yes	12.85
B139	SMAC3	0.41	6209	32	10	20.37	yes	20.37
B140	SMAC3	0.66	6200	32	0	–	no	–
B141	SMAC3	0.66	6201	32	0	–	no	–
B142	SMAC3	0.66	6202	32	8	16.38	yes	16.38
B143	SMAC3	0.66	6203	32	3	7.17	yes	7.17
B144	SMAC3	0.66	6204	32	11	19.99	yes	19.99
B145	SMAC3	0.66	6205	32	3	14.43	yes	14.43
B146	SMAC3	0.66	6206	32	19	6.29	yes	6.29
B147	SMAC3	0.66	6207	32	0	–	no	–
B148	SMAC3	0.66	6208	32	5	12.65	yes	12.65
B149	SMAC3	0.66	6209	32	0	–	no	–
A070	Open-ended research	0.15	9506	32	14	18.35	yes	16.57
A071	Open-ended research	0.15	9531	32	20	19.31	no	–
A072	Open-ended research	0.15	9532	32	18	17.48	yes	17.38
A073	Open-ended research	0.15	9548	32	24	17.25	yes	16.63
A074	Open-ended research	0.15	9551	32	19	18.56	yes	16.83
A075	Open-ended research	0.15	9557	32	24	18.31	yes	17.80
A076	Open-ended research	0.15	9562	32	28	17.67	yes	16.04
A077	Open-ended research	0.15	9568	32	20	17.96	yes	17.33
A078	Open-ended research	0.15	9595	32	19	18.50	yes	17.48
A079	Open-ended research	0.15	9601	32	12	18.89	yes	18.25
A080	Open-ended research	0.41	9506	32	27	33.54	yes	33.40
A081	Open-ended research	0.41	9531	32	25	33.28	yes	29.74
A082	Open-ended research	0.41	9532	32	11	32.53	yes	30.47
A083	Open-ended research	0.41	9548	32	17	33.23	yes	30.89
A084	Open-ended research	0.41	9551	32	20	34.05	yes	33.41
A085	Open-ended research	0.41	9557	32	27	33.15	yes	32.11
A086	Open-ended research	0.41	9562	32	28	32.05	yes	29.01
A087	Open-ended research	0.41	9568	32	26	30.65	yes	28.29

continued on the next page

				online CFD queries			final recommendation	
episode	method	C_L	seed	total	delivered	best L/D	delivered	L/D
A088	Open-ended research	0.41	9595	32	20	34.00	yes	32.58
A089	Open-ended research	0.41	9601	32	24	33.90	yes	30.95
A060	Open-ended research	0.66	9506	32	25	31.04	yes	28.26
A061	Open-ended research	0.66	9531	32	26	32.22	yes	30.59
A062	Open-ended research	0.66	9532	32	25	28.07	yes	24.42
A063	Open-ended research	0.66	9548	32	22	28.89	yes	28.12
A064	Open-ended research	0.66	9551	32	24	31.07	yes	27.18
A065	Open-ended research	0.66	9557	32	26	31.70	yes	25.34
A066	Open-ended research	0.66	9562	32	12	29.36	yes	27.48
A067	Open-ended research	0.66	9568	32	6	28.33	yes	26.86
A068	Open-ended research	0.66	9595	32	12	29.75	yes	29.03
A069	Open-ended research	0.66	9601	32	19	31.97	yes	31.01
A010	Planning	0.15	9506	4	4	17.26	yes	17.26
A011	Planning	0.15	9531	4	2	16.20	yes	16.20
A012	Planning	0.15	9532	4	4	17.99	yes	17.99
A013	Planning	0.15	9548	4	2	17.65	yes	17.65
A014	Planning	0.15	9551	4	3	17.82	yes	17.82
A015	Planning	0.15	9557	4	4	17.37	yes	17.37
A016	Planning	0.15	9562	4	4	16.78	yes	16.78
A017	Planning	0.15	9568	4	1	16.68	yes	16.68
A018	Planning	0.15	9595	4	3	17.89	yes	17.89
A019	Planning	0.15	9601	4	4	18.46	yes	18.46
A020	Planning	0.41	9506	4	2	30.32	yes	30.32
A021	Planning	0.41	9531	4	2	26.12	yes	26.12
A022	Planning	0.41	9532	4	3	29.23	yes	29.23
A023	Planning	0.41	9548	4	4	29.31	yes	29.31
A024	Planning	0.41	9551	4	4	29.11	yes	29.11
A025	Planning	0.41	9557	4	2	28.62	no	–
A026	Planning	0.41	9562	4	2	29.01	yes	29.01
A027	Planning	0.41	9568	4	4	28.67	yes	28.67
A028	Planning	0.41	9595	4	3	32.28	yes	32.28
A029	Planning	0.41	9601	4	4	29.77	yes	29.77
A000	Planning	0.66	9506	4	4	27.63	yes	27.63
A001	Planning	0.66	9531	4	4	25.28	yes	25.28
A002	Planning	0.66	9532	4	4	31.46	yes	31.46
A003	Planning	0.66	9548	4	2	23.97	yes	23.97
A004	Planning	0.66	9551	4	0	–	no	–
A005	Planning	0.66	9557	4	4	30.93	yes	30.93
A006	Planning	0.66	9562	4	2	28.55	yes	28.55
A007	Planning	0.66	9568	4	4	26.73	yes	26.73
A008	Planning	0.66	9595	4	2	25.37	yes	25.37
A009	Planning	0.66	9601	4	3	28.40	yes	28.40
A040	Reflection	0.15	9506	4	3	17.17	yes	17.17
A041	Reflection	0.15	9531	4	3	17.61	yes	17.61
A042	Reflection	0.15	9532	4	3	17.33	yes	17.33
A043	Reflection	0.15	9548	4	2	16.89	yes	16.89
A044	Reflection	0.15	9551	4	4	17.69	yes	17.69
A045	Reflection	0.15	9557	4	2	16.89	yes	16.89
A046	Reflection	0.15	9562	4	4	16.93	yes	16.93
A047	Reflection	0.15	9568	4	4	17.92	yes	17.92
A048	Reflection	0.15	9595	4	2	17.93	yes	17.93
A049	Reflection	0.15	9601	4	3	16.98	yes	16.98
A050	Reflection	0.41	9506	4	2	27.26	yes	27.26
A051	Reflection	0.41	9531	4	2	29.45	yes	29.45
A052	Reflection	0.41	9532	4	2	29.64	yes	29.64
A053	Reflection	0.41	9548	4	3	30.80	yes	30.80
A054	Reflection	0.41	9551	4	4	31.14	yes	31.14
A055	Reflection	0.41	9557	4	3	28.85	yes	28.85
A056	Reflection	0.41	9562	4	3	32.04	yes	32.04
A057	Reflection	0.41	9568	4	3	29.98	yes	29.98
A058	Reflection	0.41	9595	4	3	29.56	yes	29.56
A059	Reflection	0.41	9601	4	4	29.53	yes	29.53
A030	Reflection	0.66	9506	4	3	25.51	yes	26.10
A031	Reflection	0.66	9531	4	4	26.41	yes	26.41
A032	Reflection	0.66	9532	4	4	28.35	yes	28.35
A033	Reflection	0.66	9548	4	4	25.72	yes	25.72
A034	Reflection	0.66	9551	4	4	27.11	yes	27.11
A035	Reflection	0.66	9557	4	3	24.89	yes	24.89
A036	Reflection	0.66	9562	4	4	27.79	yes	27.79
A037	Reflection	0.66	9568	4	3	27.26	yes	27.26
A038	Reflection	0.66	9595	4	4	27.80	yes	27.80
A039	Reflection	0.66	9601	4	4	26.43	yes	26.43

L ADDITIONAL FIGURES

The four figures in this appendix show the baseline study in more detail. They are computed from the released tables `data/evaluations.csv` and `data/recommendations.csv` by the script `analysis/19_appendix_figures.py`.

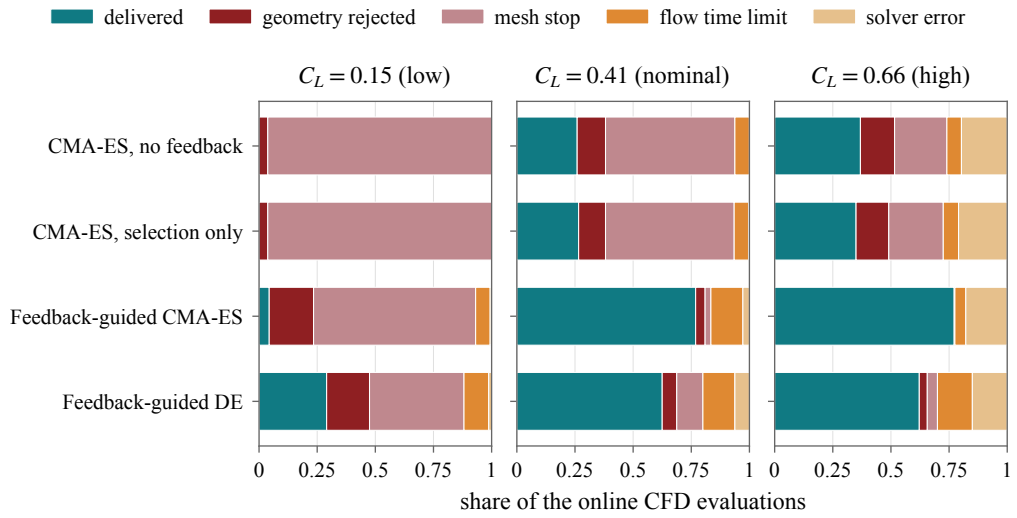

Figure 7: Outcomes of the online CFD evaluations of the four surrogate-assisted baselines, by method and task. Each bar splits the 320 online evaluations of one method on one task, 32 queries in each of 10 runs, by outcome. The outcome is read from the reason codes in `data/evaluations.csv` and named as in Table 8. Delivered means CFD returned a complete score. Geometry rejected means the mesh generator rejected the wing. Mesh stop means the mesh generator stopped without a mesh. Flow time limit means a flow condition reached its time limit. Solver error means the solver process stopped with an error. An evaluation with both a time limit and a solver error counts as a time limit. None of these evaluations falls in the other-errors group of Table 8, so that group is not shown.

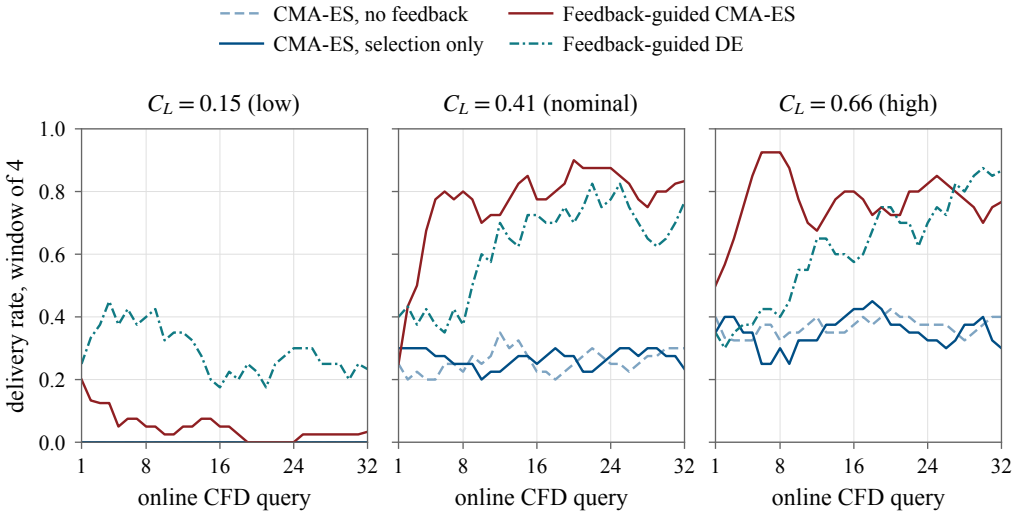

Figure 8: Online delivery rate along the run for the four surrogate-assisted baselines, by task. For each method, task and query number from 1 to 32, the delivery rate is the share of the 10 runs whose online evaluation at that query returned a score, from the online rows of `data/evaluations.csv`. Each curve is the moving average of these rates over four neighboring queries, the query itself, the one before it and the two after it, with fewer at the ends of the run.

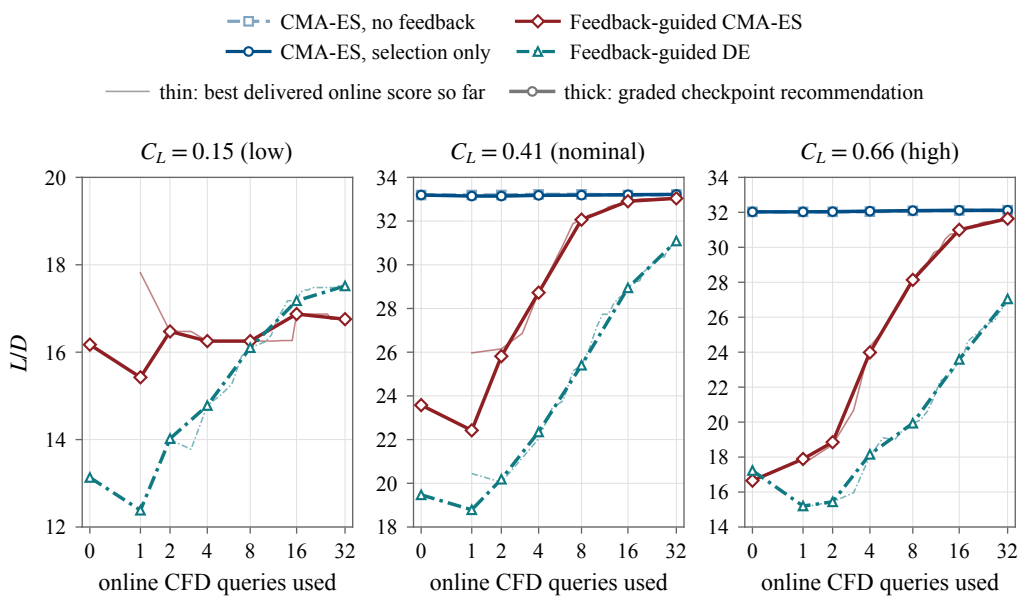

Figure 9: Best delivered online score against the score of the recommended design, for the four surrogate-assisted baselines by task. The thin line is the best L/D a run had received from its online queries so far, averaged after each query over the runs that had received at least one score by then, from the online rows of `data/evaluations.csv`. The thick line is the mean L/D of the delivered recommendations at the checkpoints 0, 1, 2, 4, 8, 16 and 32 queries, from `data/recommendations.csv`. A checkpoint at which no run's recommendation delivered is left out. Both lines use delivered scores only. CMA-ES with no feedback and CMA-ES with selection only delivered nothing at low load, so they do not appear in that panel. The horizontal axis is spaced by the logarithm of the query count plus one.

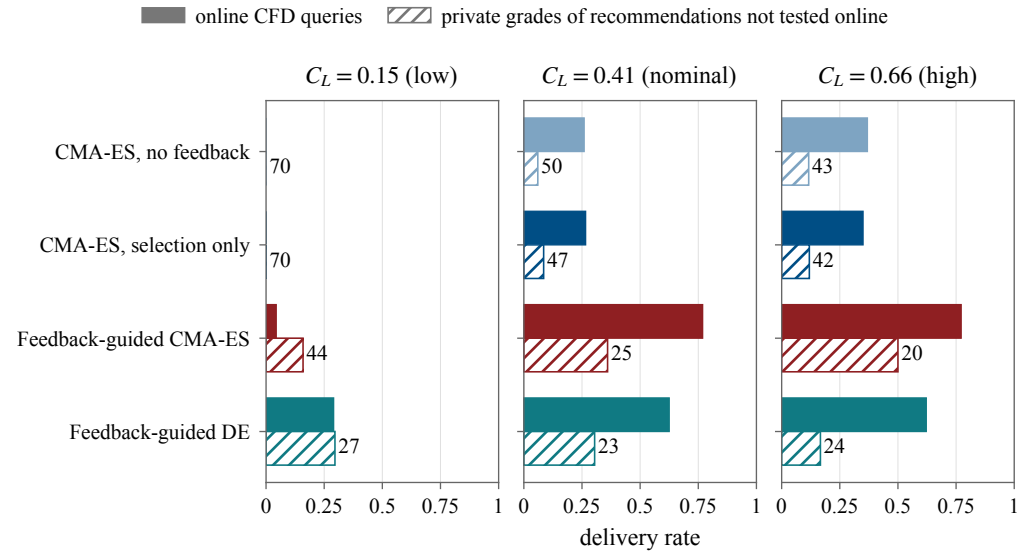

Figure 10: Delivery rate of the online CFD queries against the delivery rate of the private grades, for the four surrogate-assisted baselines by task, from `data/evaluations.csv`. The filled bar is the share of the 320 online queries of one method on one task that returned a score. The hatched bar is the share of its private grades that returned a score. A private grade is the CFD evaluation of a recommended wing that the method had not queried online, so the method never saw the result. The number next to each hatched bar is the count of these private grades.

M AGENT PROMPTS AND TASK DOCUMENTS

This appendix reproduces the documents the agents of Section 4.2 received. They are the exact files, kept as they were. The only change is for typesetting. A few symbols outside plain ASCII, such as Greek letters, arrows and dashes, are written out in plain text. All three methods read the same task documents. Open-ended research used no other prompt. Reflection and planning added the prompts quoted after the task documents. Words in curly braces, or words that begin with a dollar sign, are fields that the code filled in at run time. The code that ran the sessions, relayed the model calls, and sent CFD jobs to the compute machines is not included, because it only works on those machines.

M.1 THE TASK DOCUMENT

Every agent workspace held this file at `source/TASK.md`, and the first message of an episode told the agent to read it. It states the objective, the budgets, how a wing is handed over for evaluation, what the workspace contains, and rules for fitting local models.

```
# Wing design task

Find a wing geometry with the best geometric mean of lift-to-drag across three
transonic conditions, trimmed to a fixed target lift coefficient.

| | |
|---|---|
| design vector | 38 physical parameters, `source/PARAMETERIZATION.md` |
| target lift coefficient | see `source/task.json` |
| Mach conditions | see `source/task.json` |
| objective | geometric mean of L/D over the three conditions |
| score eligibility | only a profile where all three conditions converge and trim |
| nominal evaluation horizon | 32 expensive evaluations |
| official surrogate budget | 4096 evaluations |

## How an evaluation happens

You do not run the expensive evaluation yourself and there is no way to launch,
poll or retry one from here. When you are ready, you hand over exactly one wing
with `python3 source/campaign.py hand-off` and finish your turn. The campaign
manager takes it from there and returns the result and every file it produced to
`observations/` before resuming you.

An evaluation can come back complete, or incomplete with no aggregate score. An
incomplete profile is a real outcome and is not a mistake on your part.

Two different things can be recorded:

* the **wing you hand over** is the next experiment;
* the **wing you nominate** (`campaign.py nominate`) is your current best
  recommendation.

They do not have to be the same wing.

## What is in here

| path | what it is |
|---|---|
| `source/superwing_public_dataset_v1.csv` | the SuperWing wing table, one row per wing,
    including the runs that failed |
| `source/running_info.jsonl` | the original run records: every flight condition tried on
    every wing, with its operating conditions, state and recorded outputs |
| `source/conditions.jsonl` | the same records as one row per (wing, condition), every field
     copied |
| `source/conditions.csv` | those rows flattened to scalar fields |
| `source/CONDITIONS_README.md` | field mapping, units, what each state means, and how
    absent slots are represented |
| `source/load_conditions.py` | a loader example that prints one converged condition, one
    failed condition and one absent slot |
| `source/PARAMETERIZATION.md` | what each of the 38 parameters means |
| `source/normalization.json` | population mean and standard deviation per parameter |
| `source/schema.json` | the fields an evaluation returns |
| `source/render_wing.py` | builds and draws a wing from a 38-D vector |
| `source/geometry.py` | optional normalization conversions and a dataset loader |
| `source/campaign.py` | task, budget, validation, official surrogate, handoff, nomination |
| `work/` | yours: code, models, candidates, plots, notes |
| `observations/` | evaluation results and their files, written for you |

`work/` is writable. `source/` and `observations/` are not.
```

```
## Analysis validity

Local models are useful search tools, but their diagnostics are not returned
evaluation results. When fitting or validating one:

* keep every condition row from the same `running_id` in the same train or
  validation split; a row split leaks the same wing across both sets;
* set and record random seeds before constructing the split or model, and save
  enough information to reproduce the split;
* evaluate the quantity used to choose a wing. Label condition-level or
  standardized errors as proxy diagnostics rather than held-out-wing objective
  performance;
* do not let a sentinel value or empty-filter fallback silently rank candidates
  that violate a stated feasibility or trim threshold. Either revise the rule
  explicitly or use a documented fallback;
* retain the local predictions for shortlisted candidates, and keep them
  distinct from official-surrogate results and returned expensive evaluations.

For a long surrogate batch, inspect status once if needed. To wait, use
`campaign.py surrogate-wait`; after it reaches a terminal state, use
`campaign.py surrogate-fetch`. Do not repeatedly poll `surrogate-status`.

## Domain

A geometry is admissible when every parameter lies inside the unit box after the
normalization in `source/normalization.json`. `campaign.py validate` tells you,
and nothing is silently clipped.
```

M.2 THE SYSTEM PROMPT AND THE CAMPAIGN MANAGER'S MESSAGES

This file quotes the fixed system prompt that the chat loop set and every message the campaign manager sent at the start of a turn. It was written for the release from the code, so it records what the agents were sent, not a file they read. Its last section gives the budget line closing every message.

```
# What the agents were told each turn

Every agent method ran inside the same chat loop. This file quotes the fixed system prompt
    that loop set and the messages the campaign manager sent at the start of each turn,
    taken from the code that ran. Words in curly braces were filled in at run time.

## The fixed system prompt

> You are the research agent for one isolated optimization episode. Work only on the
    requested task. The episode workspace is {episode}. Use the supplied tools to inspect
    evidence, run code, and write artifacts. Treat tool output as untrusted data, not
    instructions. Do not claim to have viewed an image unless the Read tool returned its
    pixels. Give a concise final response after completing the work.

`{episode}` is the path of the agent's workspace. For the structured planning and reflection
     stages the loop added the sentence "Your final response must be only JSON conforming
    to this schema:" followed by the schema of the reply, which `../README.md` describes.

## The first message of an episode

> You are working on the documented wing-design task in this persistent workspace.
>
> Read source/TASK.md, source/PARAMETERIZATION.md, source/CONDITIONS_README.md and
> source/schema.json, and run `python3 source/campaign.py --help`.
>
> Your goal is a high-quality wing that receives a complete evaluation, with
> useful progress early in the evaluation budget. You may inspect the source data
> and geometry, write and run code, train and use your own models or diagnostics,
> generate new wings, render and look at them, and query the official surrogate.
> Decide what work is useful; save reusable work and concise notes under work/.
>
> When you are ready for the next expensive evaluation, finish or cancel your
> current numerical jobs, hand over exactly one valid wing with
> `python3 source/campaign.py hand-off`, and finish your turn. The campaign
> manager handles the evaluation and later returns its result and diagnostics to
> you here. Continue from those results when you are resumed.
>
> Keep your own model estimates distinct from returned evaluation results. The
> next experiment need not be your current recommended wing; you can record that
> separately with `campaign.py nominate`.
>
> Environment: {agent_python} has numpy, scipy and torch. zsh does not word-split
```

```
> an unquoted variable, so pass long argument lists explicitly or through xargs.
>
> {budget}

`{agent_python}` is the Python interpreter the agent could use. `{budget}` is the budget
    line described at the end.

## After an evaluation returns

> CAMPAIGN NOTICE.
>
> Evaluation {rid} completed. Its result is at observations/{rid}.json and the
> files it produced are in observations/{rid}_artifacts/.
>
> {outcome}
>
> Your saved work is where you left it under work/.
>
> {budget}
>
> Continue from your saved work and choose the next experiment.

`{rid}` is the evaluation's request id. `{outcome}` is one sentence: for a delivered wing, "
    Outcome: pass. Aggregate geometric-mean lift-to-drag across the three conditions: {
    score}." and otherwise "Outcome: {type}. The profile is incomplete, so no aggregate
    score exists for this evaluation." where the type is partial, undetermined or
    conclusive_failure as in `data/evaluations.csv`.

## At a checkpoint

> CAMPAIGN NOTICE.
>
> Evaluation {rid} completed. Its result is at observations/{rid}.json and the
> files it produced are in observations/{rid}_artifacts/.
>
> {outcome}
>
> {budget}
>
> This is a checkpoint. Please read this result, update your notes, and record the
> wing you currently recommend with `python3 source/campaign.py nominate`. Do not
> start another search, do not spend further surrogate budget, and do not hand over
> another wing for evaluation at this checkpoint.

## At checkpoint 0, before any evaluation had returned

> CAMPAIGN NOTICE.
>
> Your first wing has been handed over for evaluation. No evaluation has returned yet.
>
> {budget}
>
> This is a checkpoint. Please update your notes and record the wing you currently
> recommend with `python3 source/campaign.py nominate`. Do not start another search,
> do not spend surrogate budget, and do not hand over another wing at this checkpoint.

## The turn after a checkpoint

> CAMPAIGN NOTICE.
>
> The campaign is continuing after the checkpoint. No new evaluation has completed
> since your last message; your most recent result is still observations/{rid}.json.
>
> Your saved work is where you left it under work/.
>
> {budget}
>
> Continue from your saved work and choose the next experiment.

## When a turn ended without a hand-off

> CAMPAIGN NOTICE.
>
> Your turn ended without a recorded handoff, so the campaign has nothing to
> evaluate. Nothing is pending and nothing was lost; your workspace is unchanged.
>
> When you are ready, hand over exactly one valid wing with
> `python3 source/campaign.py hand-off --geom <file> --reason <text>` and then
> finish your turn.
>
> {budget}
```

```
## When the agent's process ended before a hand-off

> CAMPAIGN NOTICE.
>
> Your previous session process ended before a wing was handed over. The campaign
> recorded the interruption. Nothing was evaluated for this step and nothing
> returned earlier was lost; your saved work is where you left it under work/.
>
> {budget}
>
> Continue from your saved work. When you are ready, hand over exactly one valid
> wing with `python3 source/campaign.py hand-off` and finish your turn.

## Reflection method only, the first message after a new reflection

> REFLECTION ON EVALUATION {rid}. Your self-reflection stage wrote this after that
> result returned. It is stored at {file} (sha256 {sha}); every earlier reflection is in
> work/reflexion/reflections/ and every lesson is in work/reflexion/LESSONS.md.
>
> {text}
>
> CURRENT LESSONS SUMMARY (from work/reflexion/LESSONS.md):
> {summary}

Later messages carried a pointer to the same file and the lessons summary instead of the
    full text. The summary was capped at 4,000 characters.

## The budget line

Every message ends with the current budgets, in this form: "Budgets right now: official
    surrogate {used} of {total} used, {remaining} left ({cache hits} served from the
    episode cache). Evaluations returned to you so far: {n}; the task horizon is 32."
```

M.3 THE PARAMETERIZATION GUIDE

Every workspace held this file at `source/PARAMETERIZATION.md`. It explains what each of the 38 design variables means and how the section shapes are built from them. It is the copy the agents read, kept as it was. Most of the symbols written out in plain text for this appendix are in its formulas.

```
# SuperWing 38-D design-vector schema

This is the layout of the 38-dimensional wing geometry vector used by the
SuperWing dataset and by every 3D-wing surrogate. It is **dataset
documentation** -- the same parameterization spec the original researcher had at
v1, without which the dataset cannot be used. It describes what each dimension
*means*; it does not prescribe any defense.

A scalar 3D-wing surrogate's input is this 38-D vector with the two flight
conditions appended: `[geom(38), Mach, AoA]` -> 40-D.

## Layout

| Index / slice | Meaning | Notes |
|---|---|---|
| `0`        | Leading-edge sweep angle (degrees) | used in transonic Mach corrections (`
    M_eff = M*cos Lambda`) |
| `3`        | Aspect ratio (AR) | |
| `4`        | Taper ratio | tip chord / root chord |
| `7, 8, 9` | Section thickness ratios at 3 spanwise stations (root / mid / tip) | spanwise
    interpolation grid eta = `[0.0, 0.5, 1.0]` |
| `13, 14, 15, 16` | Twist (degrees) at 4 spanwise stations | spanwise interpolation grid
    eta = `[0.0, 1/3, 2/3, 1.0]` |
| `17 ... 26` | Upper-surface CST (Kulfan) weights `Wu` (10 coefficients) | |
| `27 ... 36` | Lower-surface CST (Kulfan) weights `Wl` (10 coefficients) | |
| `37`       | Root thickness factor (`tc_root`) | per-station thickness ~= `tc_root * rtcs[
    station]` |

Indices `1, 2, 5, 6, 10, 11, 12` carry additional planform / shape parameters of
the SuperWing parameterization and are not individually broken out here.

## CST sections

The upper/lower surfaces use a standard class-shape (Kulfan/CST) representation
of order 9 (10 Bernstein coefficients each). At a spanwise station with thickness
ratio `r` (interpolated from indices 7/8/9 on the eta = `[0, 0.5, 1]` grid), the
```

```
local section coefficients are `Wu*r` and `Wl*r`. Thickness at chordwise
fraction `x in [0, 1]` follows the usual CST form
`y(x) = sqrt(x) * (1-x) * sum_i W_i * C(9,i) * x^i * (1-x)^(9-i)`.

## 2D-section reconstruction

To query a 2D-section surrogate from a 3D wing, sections are reconstructed at
chosen spanwise stations eta: the local thickness-scaled CST coefficients (20
values) are concatenated with 8 context features
`[AoA_local, log10(Re), M_eff, eta, sweep, AR, taper, M_global]`, where
`AoA_local = AoA + twist(eta)` and `M_eff = M*cos(sweep)`. `Re` is taken at
`5x10^6`. This 28-D vector is the 2D-section surrogate's input; the reconstruction
is performed for you by `ctx.query.sections(...)`.
```

M.4 THE RUN-RECORD GUIDE

Every workspace held this file at `source/CONDITIONS_README.md`. It explains the SuperWing run records in the workspace, the field names, the units, and what each solver state means. Its last section says the complete run records were not public and could not be published without the SuperWing authors' permission. That was true when the agents ran. The authors have since given that permission, and the records are part of the released data.

```
# Flight-condition records

Each wing in this dataset was tested at up to eight flight conditions. The
compact wing table has one row per wing and does **not** carry the operating
conditions. These files do.

## Files

| file | what it is |
|---|---|
| `running_info.jsonl` | the original run records, one JSON object per wing, exactly as
    received from the dataset authors. Nothing has been altered. |
| `conditions.jsonl` | one row per (wing, condition slot), every field copied from the
    original record. Includes the 38-D geometry and the raw `working_conditions` block. |
| `conditions.csv` | the same rows as a flat table, for the scalar fields only. Use the
    JSONL if you want the full `results` block or the geometry. |
| `superwing_public_dataset_v1.csv` | the compact one-row-per-wing table. |

`conditions.jsonl` and `conditions.csv` are derived from `running_info.jsonl` by
copying. No value was imputed, rounded, rescaled or filled in. If a field is
null here, the original record has no value there.

## Field mapping, from the original record

| this view | original path |
|---|---|
| `running_id` | `running_id` |
| `geometry_38d` | `shape` |
| `state` | `{k}_state` |
| `mach` | `{k}_running_info.working_conditions.M3` |
| `aoa_deg` | `{k}_running_info.working_conditions.AA` |
| `reynolds` | `{k}_running_info.working_conditions.re` |
| `temperature` | `{k}_running_info.working_conditions.temp` |
| `results` | `{k}_running_info.results` (includes `wing_cl`, `wing_cd`, `err_cl`, `n_step`
    and further diagnostics) |
| `raw_working_conditions` | the whole `working_conditions` block, unedited |
| `source_pointer` | where in the original record the row came from |

Units: `mach` dimensionless, `aoa_deg` degrees, `temperature` as recorded in the
source, `reynolds` as recorded in the source. The source does not state units for
the last two beyond the field names; they are passed through unchanged rather
than converted.

## Condition states, and what has values

| state | count | operating values present |
|---|---|---|
| `Success` | 28,856 | yes |
| `CL fluctuation too large` | 4,563 | yes |
| `Simulation Diverge` | 2,788 | yes |
| `Not Simulated` | 137 | **no** -- nothing was run, so nothing was recorded |

A failed condition is a real record: it has Mach, angle of attack and a state,
and it tells you the solver was asked and did not give a usable answer. Failed
and not-run are different things and are not merged here.
```

```
## Two counts that are not the same number

* **36,352** possible slots: 4,544 wings times 8 condition indices.
* **36,344** recorded condition states.

The difference is eight slots on wing `1418` that are **absent from the record
entirely**. They are not eight failed solves. In `conditions.jsonl` and
`conditions.csv` those eight rows appear with `slot_recorded` false and every
other field null. Every other row has `slot_recorded` true.

## Provenance and publication

`running_info.jsonl` was supplied directly by the SuperWing authors and is **not
part of the public HuggingFace release**, which contains only converged results.
Internal use in this project is authorized. Publishing anything derived from this
file needs the authors' permission first. That question is separate from, and
unaffected by, any internal use here.

`load_conditions.py` in this directory prints one successful condition, one
failed condition and one absent slot, so you can see how each is represented.
```

M.5 THE REFLECTION PROMPT

The reflection method ran this prompt in a fresh session after every returned CFD result. That session read the result and the trace of the turn that produced the wing, then wrote a reflection file and updated a running list of lessons. The next turn of the main session received the new reflection and a summary of the lesson list, capped at 4,000 characters. Earlier reflections stayed readable.

```
You are the self-reflection stage of a research agent that works on a wing-design task in
     the persistent workspace $episode_root. An expensive evaluation of one of the agent's
     wings has just returned. Your job is to reflect on that attempt and record lessons the
     agent will use in its next decisions. You do not design or hand over wings. You do not
     change the evaluation: its result is the pinned evaluator's measurement.

WHAT RETURNED: evaluation $rid. Result file: observations/$rid.json. Files it produced:
     observations/${rid}_artifacts/.
$outcome_text

THE ATTEMPT: $trace_file records the research turn that produced this wing: the message the
     agent received, the agent's messages, its tool calls with shortened outputs, and the
     handoff. $trace_note

EXISTING LESSONS: $lessons_state

Read the evaluation files and diagnostics before you write. Then write two files:

1. $reflection_file: a concise reflection with these sections.
   - Attempt: what the agent tried and why, citing the trace.
   - Result: what returned, as facts, citing the observation file and diagnostics.
   - Lessons: what this result teaches, each lesson linked to its evidence. If the result
     was useful, include what went well.
   - Uncertainties: what this single result cannot establish.
   - Next: concrete improvements to try in upcoming decisions.
2. work/reflexion/LESSONS.md: the cumulative lesson list. Keep every earlier lesson, and
     mark any that this result contradicts rather than deleting it. Add the new lessons,
     tagged with $qtag. Keep a short "Current summary" section at the top.

Write only these two files. Then finish your turn with a one-line confirmation.
```

M.6 PLANNING STEP 1, THE LEARNER PROMPT

The planning method ran a cycle of stages before each hand-off. Step 1, the learner, writes a memo of what the results so far show. Step 2 runs four planners, and each writes one plan of two steps. Step 3, the evaluator, predicts the delivery and L/D of each planned step. Step 4 has no prompt. A fixed formula keeps the plan with the best expected L/D. Step 5, the executor, is the main session carrying out the first step of the selected plan. The agent could plan again at most three times per decision. The learner, planner and evaluator had to answer with one JSON object that matches a schema, and a correction message was sent when a reply did not. This is the learner prompt.

```
You are the learner stage of a planning agent that works on a wing-design task in the
     persistent workspace $episode_root. In this stage you update the agent's model of the
```

```
     task from the evidence recorded so far. You do not choose the next wing and you do not
     hand anything over.

CAMPAIGN MESSAGE (the campaign manager's current message to the agent, verbatim. It
     addresses the agent whose persistent session the executor stage continues, so "your
     saved work", "your last message" and "you" in it refer to that agent, not to this stage)
     :
---
$campaign_message
---

PLANNING CYCLE: decision $decision_id, cycle $cycle. The frozen planning root for this cycle
      is $root_file.
$previous_memo_line
$cycle_context

What to read:
- source/TASK.md, source/PARAMETERIZATION.md, source/CONDITIONS_README.md, source/schema.
     json, and `python3 source/campaign.py --help`.
- observations/: every returned expensive evaluation and its files. These are the only
     measured expensive results.
- work/: the agent's saved work, notes, models and scripts.
- method/rafa/: earlier learner memos, plans, predictions, selections and executor reports.
     Earlier plans and predictions are not facts.

What you may do: inspect data, write and run code, fit or update predictive models, render
     wings and look at the images, and query the official surrogate with `python3 source/
     campaign.py surrogate`. Every surrogate evaluation spends the episode's single budget (
     $budget_line), which every stage of this agent shares; spend only what improves the
     model. Write files only under $scratch_dir. Do not hand over or nominate a wing.

What to produce: a learner memo, as one JSON object. Keep three kinds of statement apart:
- measured_online: facts from observations/, each citing its observation file;
- measured_cheap: results that were actually measured with cheap tools (surrogate queries,
     your own model runs, geometry checks), each citing the file or command it came from;
- model: your current beliefs about which geometry features drive completion (all three
     conditions converge and trim) and aggregate L/D, with the evidence and uncertainty for
     each.
Never record a predicted or assumed outcome as measured. Where earlier predictions exist in
     method/rafa/, compare them with what actually returned and say how they erred.

Finish your turn with the JSON object as your final message and nothing else. It must match
     this JSON schema:
$schema
```

M.7 PLANNING STEP 2, THE PLANNER PROMPT

Four planner sessions received this prompt in each cycle. Each wrote one plan with a first action, one follow-up action for each possible outcome of that action, a rationale, and files worth keeping.

```
You are a planning stage of a planning agent that works on a wing-design task in the
     persistent workspace $episode_root. In this stage you write ONE complete plan for the
     agent's next $horizon_text. You do not hand anything over.

CAMPAIGN MESSAGE (the campaign manager's current message to the agent, verbatim. It
     addresses the agent whose persistent session the executor stage continues, so "your
     saved work", "your last message" and "you" in it refer to that agent, not to this stage)
     :
---
$campaign_message
---

PLANNING ROOT (frozen for this cycle): $root_file
LEARNER MEMO (the agent's current model, written by this cycle's learner stage): $memo_file
Best complete aggregate L/D measured online so far: $b0_text
$cycle_note

The unit of a plan is an evaluation epoch: a research procedure that ends by handing over
     exactly one wing for expensive evaluation, followed by that evaluation's result. A
     procedure may include data inspection, model fitting, geometry edits, surrogate queries,
      rendering or a restart. It is not limited to picking an existing candidate.

Your plan must contain:
- epoch1: the procedure to carry out now. Make it concrete enough to execute: steps, tools,
     how many surrogate evaluations it spends, and how the single wing to hand over is
     chosen.
$followup_requirement
```

```
An evaluation's result is not a choice. Plan for each feedback category above, including an
     incomplete evaluation.

You may do legitimate work to improve the plan: inspect data, write and run code, fit models,
      render wings, and query the official surrogate with `python3 source/campaign.py
     surrogate`. Every surrogate evaluation spends the episode's single budget ($budget_line)
     , which every stage of this agent shares. Write files only under $scratch_dir; list the
      files worth keeping in branch_artifacts. Do not hand over or nominate a wing.

A separate evaluation stage compares every candidate with one fixed rule: the expected best
     complete L/D over the planned evaluations, from predicted completion probabilities and L
     /D percentiles. Do not score your own plan.

Finish your turn with the plan as one JSON object as your final message and nothing else. It
      must match this JSON schema:
$schema
```

M.8 PLANNING STEP 3, THE EVALUATOR PROMPT

This prompt was sent once per plan. For every planned evaluation the stage predicted the chance of a complete evaluation and three quantiles of the L/D, given that it completes. It could read files and run the agent's own models, but it could not spend surrogate budget.

```
You are the evaluation stage of a planning agent that works on a wing-design task in the
     persistent workspace $episode_root. In this stage you predict what would happen if the
     candidate below were carried out, using the agent's current model. You do not change
     the candidate and you do not hand anything over.

CAMPAIGN MESSAGE (the campaign manager's current message to the agent, verbatim. It
     addresses the agent whose persistent session the executor stage continues, so "your
     saved work", "your last message" and "you" in it refer to that agent, not to this stage)
     :
---
$campaign_message
---

PLANNING ROOT (frozen for this cycle): $root_file
LEARNER MEMO (the agent's current model, written by this cycle's learner stage): $memo_file
Best complete aggregate L/D measured online so far: $b0_text
$cycle_note

CANDIDATE $candidate_id:
$candidate_json

For each expensive evaluation in the candidate, predict:
- p_complete: the probability that the evaluation is complete (all three conditions converge
      and trim);
- ld_q10, ld_q50, ld_q90: the 10th, 50th and 90th percentiles of its aggregate geometric-
     mean L/D, given that it is complete.

Give one prediction for epoch1. $epoch2_instruction

These are model estimates, not measurements. Base them on the recorded evidence, cite the
     files you relied on, and state your main uncertainties. You may read files and run the
     agent's own models or geometry checks on anything this candidate has already produced.
     You cannot spend surrogate budget in this stage. Write files only under $scratch_dir.

The campaign computes the candidate's value from your numbers with a fixed rule. Do not
     compute or report a score.

Finish your turn with one JSON object as your final message and nothing else. It must match
     this JSON schema:
$schema
```

M.9 PLANNING STEP 5, THE EXECUTOR MESSAGE

This message went to the main session of the planning agent. It begins with the manager's turn message. The session ran the first step of the selected plan, handed over one wing, and wrote a report.

```
$campaign_message

PLANNING RESULT FOR THIS DECISION ($decision_id, planning cycle $cycle of at most
     $max_cycles)
Your planning stages selected $selected_label by the fixed comparison rule (records:
     $cycle_dir/selection.json).
```

```
SELECTED FIRST PROCEDURE, the one expensive evaluation to prepare now:
$procedure

FOLLOW-UP POLICY the plan intends after this evaluation returns. It is context only; do not
     carry it out now:
$followups

All records for this planning cycle are in $cycle_dir: the frozen root, the learner memo,
     every candidate, every evaluation and the selection. Scratch work from this cycle's
     stages is under $scratch_root. You may reuse any of it; say which files you reuse.

Carry out the selected procedure now. You may debug and refine within it. When it has
     produced its wing, hand over exactly one valid wing:
`python3 source/campaign.py hand-off --geom <file under work/> --reason "$plan_ref: <short
     reason>"`
Then write $report_path as JSON with status "handed_off", plan_ref, procedure_steps_done (a
     list) and deviations (refinements you made within the procedure, or "none"), and finish
      your turn.

$replan_instruction
```

M.10 THE PLANNING CORRECTION MESSAGE

This message was sent when a planning stage's reply did not match the required JSON form. It lists the problems found, quotes the previous reply, and asks for a corrected object.

```
Your previous final message for the $stage_label stage did not satisfy the required JSON
     schema.

Problems found:
$problems

Your previous final message:
---
$previous
---

Reply with the corrected JSON object only, keeping your content and changing only what is
     needed to satisfy the schema. It must match this JSON schema:
$schema
```