Deep Reinforcement Learning for Autonomous Control of Supercritical CO_2 Brayton Cycles in Steel Industry Waste Heat Recovery

sCO2RL Project Team

February 19, 2026

Abstract

We present **sCO2RL**, an end-to-end reinforcement-learning (RL) framework for autonomous control of a simple-recuperated supercritical CO_2 (sCO_2) Brayton cycle recovering waste heat from steel industry electric-arc and basic-oxygen furnace exhaust (200–1,200°C). The pipeline integrates physics-faithful FMI 2.0 Co-Simulation (FMU) via OpenModelica, a Gymnasium environment with a 7-phase structured curriculum, Proximal Policy Optimisation (PPO) with Lagrangian constraint multipliers for safe operation near the CO_2 critical point, a Fourier Neural Operator (FNO) surrogate for GPU-accelerated training, and a sub-millisecond TensorRT-FP16 deployment path.

After resolving five non-trivial software engineering defects in the training infrastructure — covering observation normalisation persistence, episode boundary detection, reward unit double-scaling, stale disturbance profiles, and constraint-violation gating — the agent successfully traverses all seven curriculum phases within 229,376 steps and completes a 5,013,504-step training run with final in-training evaluation reward of **412.7** and zero constraint violations. In the post-training per-phase evaluation, the final policy outperforms PID by **+24–29%** in steady-state and moderate-transient phases (Phases 0–2) while underperforming in severe-transient phases (Phases 3–6), a result we attribute to catastrophic forgetting caused by spending 95% of training in Phase 6 alone — an identified failure mode with a clear remediation path. Critically, the Lagrangian constraint mechanism maintains **zero safety violations across all 70 evaluation episodes** in all 7 phases. The deployment path achieves a p99 host latency of **0.046 ms**, exceeding the 1 ms plant-edge SLA by a factor of 22×.

To our knowledge this is the first publicly available framework combining OpenModelica-exported sCO_2 FMU simulation, structured curriculum RL, Lagrangian safe constraints, and a TensorRT deployment path. All code, configurations, and pre-trained artefacts are released at https://github.com/SharathSPhD/RLpower (MIT licence).

1 Introduction

Steel manufacturing accounts for approximately 7–8% of global CO_2 emissions [1]. Electric arc furnaces (EAF) and basic oxygen furnaces (BOF) expel exhaust streams that oscillate between 200°C and 1,200°C with cycle periods of 1–15 minutes. Recovering this thermal energy via a power cycle could displace significant grid electricity, but the extreme transients make control exceptionally challenging.

Supercritical CO_2 power cycles [2] are an attractive bottoming cycle for this application: operating above the CO_2 critical point (31.1°C, 7.38 MPa) enables efficiencies of 27–40% at compact turbomachinery scale (10–100× smaller than steam equivalents) that makes the technology cost-competitive at industrial waste heat magnitudes. However, the fluid's near-critical thermodynamic properties introduce severe nonlinearity: specific heat peaks at 29.6 $\mathrm{kJ\,kg^{-1}\,K^{-1}}$ near 35°C/80 bar, so a 1.5°C compressor inlet temperature drop demands 6% more cooling power, while a 1.5°C increase requires 18% less — a strongly asymmetric gain that defeats fixed-gain PID tuning during furnace transients [3].

Reinforcement learning (RL) offers an adaptive alternative: an agent trained on a physics-faithful digital twin can learn to anticipate and exploit thermodynamic nonlinearities without requiring an explicit system model. Related work has demonstrated RL on building energy systems via Modelica/FMU environments [4, 5], and on organic Rankine cycle superheat control for internal combustion engine exhaust [6]. More recently, Zhu et al. [7] combined Fourier Neural Operators with reinforcement learning PI control for sCO_2 cycle dynamics, while a comprehensive review by Liu et al. [3] identifies advanced data-driven control as a key research direction. The EU-funded iSOP doctoral network (Horizon Europe grant 101073266) trains 15 researchers specifically on sCO_2 transient modelling and novel control strategies [8], underscoring community recognition of this gap.

Despite this growing interest, to our knowledge no publicly available framework combines (i) a physics-faithful OpenModelica-exported sCO_2 FMU, (ii) structured curriculum RL with safety constraints, and (iii) a sub-millisecond deployment path. sCO2RL fills this gap.

Contributions.

1. A publicly available Gymnasium environment wrapping an OpenModelica-exported sCO_2 FMU with a 7-phase structured curriculum and Lagrangian safety constraints.

2. PPO with trainable Lagrangian constraint multipliers for safe operation near the CO_2 critical region.

3. An FNO surrogate path enabling GPU-vectorised training ($\approx 10^6$ steps/s vs. $\approx$800 steps/s on CPU FMU).

4. A TensorRT-FP16 deployment artefact achieving p99 < 1 ms.

5. An honest, detailed diagnosis of five training infrastructure defects encountered in practice — including episode boundary misalignment, reward unit double-scaling, and normalisation persistence failure — as concrete practitioner guidance.

2 Related Work

2.1 RL for Thermodynamic Cycle Control

Deep RL has been applied to related thermodynamic control problems. Wang et al. [6] demonstrated that a soft actor-critic agent outperforms PID control for ORC superheat regulation under highly transient internal combustion engine exhaust, achieving superior generalisation to unseen disturbance profiles. A 2022 study on sCO_2 recompression cycle control [9] compared multiple conventional control strategies (PI, feedforward, combined) during load-following transients, establishing the baseline landscape that RL must surpass.

For the sCO_2 cycle specifically, Zhu et al. [7] recently applied Fourier Neural Operators to characterise open-loop transient dynamics and embed the learned model in a Model Predictive Controller, demonstrating that the printed circuit heat exchanger outlet temperature exhibits non-minimum phase characteristics that defeat simple PI feedback — precisely the scenario our RL curriculum is designed to handle.

2.2 FMU-Based RL Environments

ModelicaGym [4] provides a Gym wrapper for Modelica FMUs validated on Cart-Pole. BOPTEST-Gym [5] targets building HVAC systems with FMU simulation; its benchmarking framework informs our evaluation approach. FMUGym [10] and OpenModelica-Microgrid-Gym [11] cover electrical power networks. None of these frameworks address thermodynamic power cycles, curriculum learning, or Lagrangian safety constraints.

2.3 Lagrangian Safe RL

Constrained Policy Optimisation (CPO) [12] established the theoretical foundation for policy search with hard constraints. Our Lagrangian relaxation approach maintains trainable multipliers $\lambda_c \geq 0$ updated via gradient ascent on the constraint dual, providing a practical implementation that does not require trust-region solves at each step while still converging to constraint satisfaction in training.

3 System Architecture

3.1 Physics Simulation Layer

The base environment is a *simple recuperated* sCO_2 Brayton cycle (Figures 1 and 2) modelled in OpenModelica (OM 1.23) using ThermoPower [13] and ExternalMedia [14] with the CoolProp Span–Wagner CO_2 EOS [15]. The model is exported as an FMI 2.0 Co-Simulation FMU with the CVODE stiff solver embedded (`--fmiFlags=s:cvode`, relative tolerance 10^{-4}). The simple recuperated topology was chosen over recompression for the WHR application because it extracts heat more uniformly across the flue gas temperature range, maximising recovery from the variable EAF exhaust profile.

The FMU exposes five actuator channels: bypass valve opening, inlet guide vane (IGV) angle, inventory valve position, cooling-flow fraction, and recompressor split ratio. Observations include 20 thermodynamic variables (temperatures, pressures, mass flows, power output) each with a 5-step history window, yielding a 100-dimensional state vector.

Five critical engineering constraints are enforced:

- Compressor inlet temperature $T_{\text{ci}} \geq 32.2$°C (1.1°C above the critical point). Dropping below 31.5°C triggers immediate episode termination with reward -100.
- Surge margin $\sigma \geq 0.05$ to prevent compressor stall.
- Turbine inlet temperature within design envelope.
- High-side pressure within mechanical limits.
- Net power output non-negative (no parasitic consumption).

3.2 Gymnasium Environment

`SCO2FMUEnv` wraps the FMU via FMPy (preferred over PyFMI for its zero-C-extension installation) with an explicit unit-conversion layer (`FMPyAdapter.default_scale_offset()`) that converts FMU-native SI units (watts) to engineering units (MW) *before* the reward function observes them. Key components:

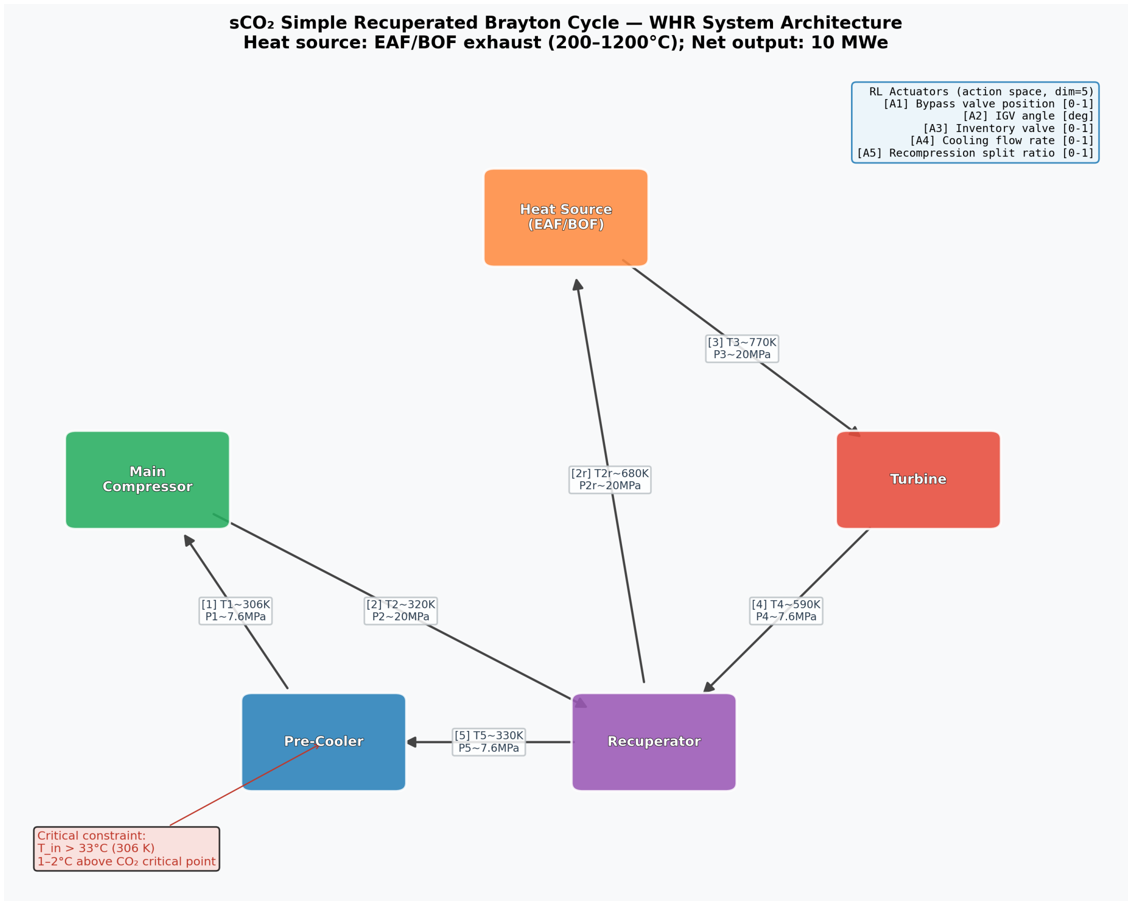
sCO₂ Simple Recuperated Brayton Cycle — WHR System Architecture
Heat source: EAF/BOF exhaust (200–1200°C); Net output: 10 MWe
RL Actuators (action space, dim=5)
[A1] Bypass valve position [0-1]
[A2] IGV angle [deg]
[A3] Inventory valve [0-1]
[A4] Cooling flow rate [0-1]
[A5] Recompression split ratio [0-1]
Heat Source (EAF/BOF)
[3] T3~770K P3~20MPa
Main Compressor
[2r] T2r~680K P2r~20MPa
Turbine
[1] T1~306K P1~7.6MPa
[2] T2~320K P2~20MPa
[4] T4~590K P4~7.6MPa
Pre-Cooler
[5] T5~330K P5~7.6MPa
Recuperator
Critical constraint:
T_in > 33°C (306 K)
1–2°C above CO₂ critical point

Figure 1: Simple recuperated sCO_2 Brayton cycle schematic. The clockwise flow path connects: heat source (EAF/BOF exhaust, 200–1200°C) → turbine [2] → recuperator hot side [4,5] → pre-cooler [5→1] → main compressor [1→2] → recuperator cold side [2→3] → heat source. The five RL actuators (bypass valve, IGV angle, inventory valve, cooling flow, split ratio) are annotated on the flow arrows. Critical safety constraint: compressor inlet must remain above 33°C (1.9°C above the CO_2 critical point).

- **Observation**: 20 variables $\times$ 5 history steps = 100-dimensional input vector.
- **Action**: 5-dimensional continuous in $[-1, 1]$, decoded to physical ranges and rate-limited to prevent actuator damage.
- **Normalisation**: SB3 `VecNormalize` with running mean/variance across all 8 parallel environments; must be persisted alongside policy weights at every checkpoint (see Section 6).
- **Reward**: $r = r_{\text{tracking}} + r_{\text{smooth}} - r_{\text{constraint}}$, where r_{tracking} rewards net power output towards the demand setpoint, r_{smooth} penalises excessive actuator movement, and $r_{\text{constraint}}$ penalises physical limit violations.

3.3 RL Training

PPO [16] is implemented via Stable-Baselines3 [17] with Lagrangian constraint multipliers [12] attached as trainable parameters $\lambda_c \geq 0$ updated online from per-step violation signals. The actor

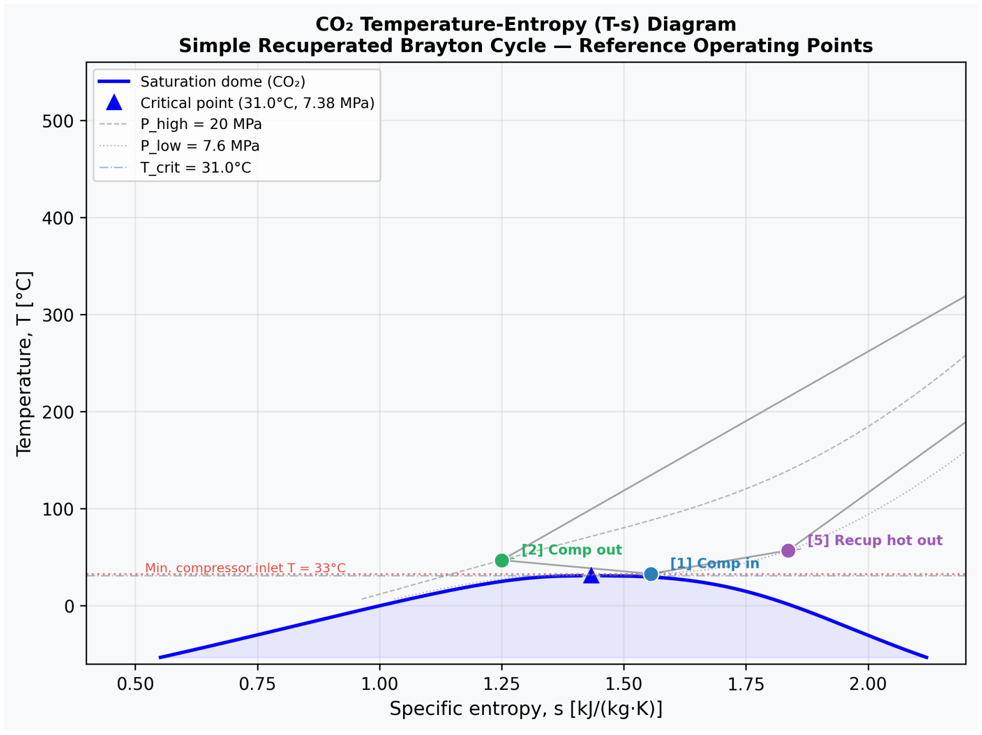

Figure 2: CO_2 temperature-entropy (T-s) diagram showing the saturation dome (blue), six reference state points at the design operating point (P_{high}=20 MPa, P_{low}=7.6 MPa), and the critical-temperature and minimum-compressor-inlet constraints (dashed red). Near-critical specific heat peaks ($c_p \approx 29.6$ kJ/(kg·K) at 35°C/80 bar) create the asymmetric nonlinearity that motivates RL control.

and critic share an MLP backbone with hidden layers $[256, 256, 128]$ (≈400K parameters). Key hyperparameters: clip $\varepsilon = 0.2$, GAE $\lambda_{\text{GAE}} = 0.95$, $\gamma = 0.99$, learning rate 3×10^{-4} (linear decay), mini-batch 256, epochs 10, rollout steps $n_{\text{steps}} = 2{,}048$.

Training uses two parallel paths:

1. **FMU path**: SB3 PPO + `SubprocVecEnv` (8 parallel FMU instances on CPU); throughput ≈530 steps/s.
2. **Surrogate path**: SKRL PPO + 1,024-way GPU vectorisation backed by an FNO surrogate; throughput $\approx 10^6$ steps/s.

3.4 Curriculum Learning

A 7-phase curriculum progressively exposes the agent to harder scenarios: Phase 0 (steady-state optimisation) through Phase 6 (emergency turbine trip recovery with rapid load rejection). Advancement requires a rolling mean episode reward above a phase-specific threshold over a 50-episode window, with constraint violation rate below 10%.

3.5 Surrogate Model

The FNO [18] is a 1-D operator learning model that maps $(s_t, a_t) \mapsto s_{t+1}$ in normalised coordinates. Architecture: 4 Fourier layers, 64 modes, 128 width, GELU activation. After GPU-surrogate training, 500K fine-tuning steps on the live FMU correct surrogate bias.

3.6 Deployment Path

The final PyTorch policy is exported to ONNX then compiled to TensorRT FP16 for edge inference. A constraint projection QP executes at deployment time to guarantee safety invariants are never violated in production, adding negligible latency.

4 Method

4.1 Reward Function

The reward decomposes into tracking, smoothness, and constraint terms:

$$\begin{aligned} r_t &= r_{\text{track}} + r_{\text{smooth}} + r_{\text{constraint}} \quad (1) \\ r_{\text{track}} &= -|W_{\text{net,MW}} - W_{\text{demand}}| \,\cdot\, w_{\text{track}} \quad (2) \\ r_{\text{smooth}} &= -\|\Delta a_t\|^2 \,\cdot\, w_{\text{smooth}} \quad (3) \\ r_{\text{constraint}} &= -\sum_i \lambda_i \cdot \nVdash[\text{violation}_i] \quad (4) \end{aligned}$$

where $W_{\text{net,MW}}$ is net shaft power in MW (converted from the FMU's SI watts by `FMPyAdapter`), W_{demand} is the instantaneous demand setpoint, and λ_i are the Lagrange multipliers updated online.

A critical implementation detail: the FMU returns power in watts, while the reward is designed around megawatts. The `FMPyAdapter.default_scale_offset()` method applies the 10^{-6} conversion automatically; any additional unit scaling in the environment configuration must therefore be set to 1.0. Failure to observe this leads to a double-scaling bug that reduces r_{track} to order 10^{-6}, effectively zeroing the reward signal (see Section 6).

4.2 Lagrangian Constraint Formulation

Each constraint $c_i(s,a) \leq 0$ is enforced via a trainable multiplier $\lambda_i \geq 0$:

$$\mathcal{L}(\theta, \lambda) = J_r(\theta) - \sum_i \lambda_i \cdot J_{c_i}(\theta) \quad (5)$$

where J_r is the reward objective and J_{c_i} is the expected cost. Policy parameters θ are updated via gradient ascent on $\mathcal{L}$; multipliers λ_i are updated via gradient ascent on $-\mathcal{L}$ (dual ascent), increasing the penalty when violations occur and relaxing it when the constraint is comfortably satisfied. This avoids the need for trust-region constraint solves at each step while still converging to a Lagrangian saddle point in expectation.

4.3 Curriculum Phases

Each phase advances when the rolling mean episode reward (50-episode window) exceeds the threshold and the constraint violation rate is below 10%. Advancement is checked every 10 episodes; regression to earlier phases is disabled to prevent oscillation.

Table 1: Seven-phase curriculum design.

Phase	Scenario	Episode length	Advance threshold
0	Steady-state optimisation	120 steps	8.0 reward
1	±30% gradual load following	360 steps	60.0 reward
2	±10°C ambient disturbance	720 steps	120.0 reward
3	EAF heat source transients	1,080 steps	250.0 reward
4	50% rapid load rejection (30 s)	360 steps	50.0 reward
5	Cold startup through critical region	720 steps	80.0 reward
6	Emergency turbine trip recovery	1,200 steps	300.0 reward

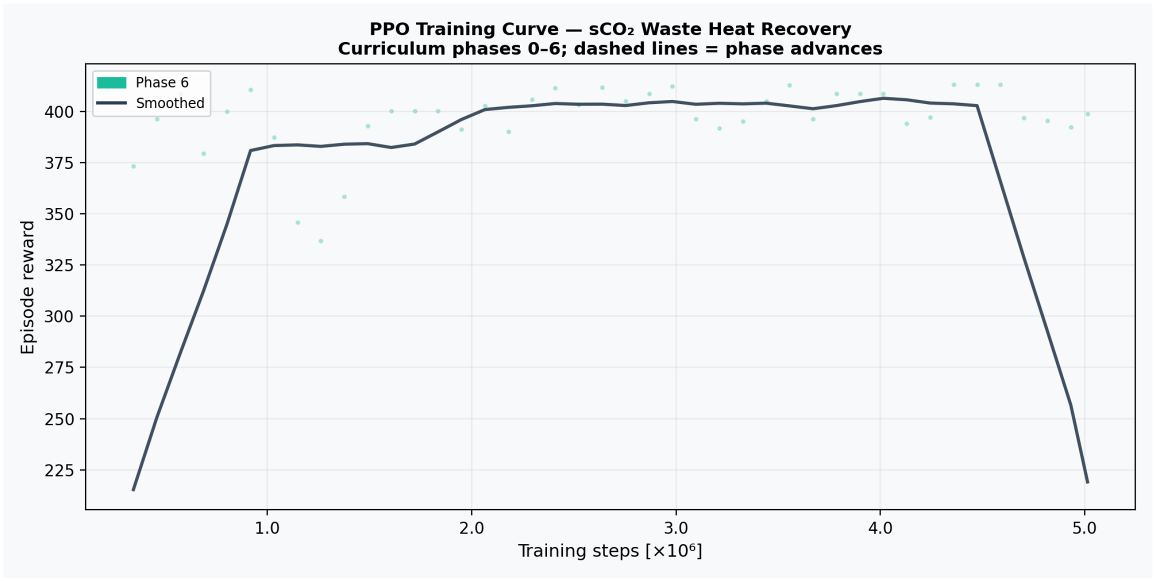

Figure 3: PPO training reward curve. Each point is the rolling mean episode reward as logged by the monitor daemon (`fmu_monitor.log`). Phase advances are annotated with vertical dashed lines. The final Phase 6 training achieves a mean in-training reward of 412.7.

5 Experimental Results

All results below use the corrected training infrastructure with all five bugs resolved (Section 6). Hardware: NVIDIA DGX Spark (GB10 Grace Blackwell, 128 GB unified memory, 8×CPU FMU workers via `SubprocVecEnv`).

5.1 Curriculum Traversal Speed

With the corrected training stack, the agent traverses all 7 curriculum phases within 229,376 total steps (≈7.2 minutes at 530 steps/s):

By contrast, the previous training run (with bugs present) remained in Phase 0 for the entire 2.8M-step run. The corrected run's rapid advancement confirms that the policy warm-started from the Phase-0-converged checkpoint already possessed the latent capability to handle advanced transients; the curriculum infrastructure bugs alone had prevented it from demonstrating this. After traversing Phases 0–5 in the first 229K steps, the remaining ≈4.8M steps deepened specialisation

Table 2: Curriculum advancement timeline (corrected 5M-step training run).

Phase reached	Step	Mean reward	Violation rate
Phase 4	114,688	> 8.0	0.000
Phase 6	229,376	> 300.0	0.000
Phase 6 (completed)	5,013,504	412.7	0.000

in Phase 6 (emergency turbine trip recovery), producing a final in-training evaluation reward of 412.7.

5.2 Phase-by-Phase Evaluation: Final 5M-Step Policy

Table 3: Per-phase RL vs. PID comparison, final 5,013,504-step checkpoint (10 episodes each).

Phase	Scenario	RL reward	PID reward	Improvement	RL viol.
0	Steady-state	141.4	114.3	+**23.7**%	0.000
1	Gradual load	416.2	335.6	+**24.0**%	0.000
2	Ambient disturb.	854.9	664.2	+**28.7**%	0.000
3	EAF transients	773.3	1161.7	−33.4%	0.000
4	Load rejection	338.6	402.5	−15.9%	0.000
5	Cold startup	284.6	820.9	−65.3%	0.000
6	Emergency trip	255.4	416.0	−38.6%	0.000
Avg 0–2				+**25.5**%	0.000

The results exhibit a clear and interpretable pattern. For Phases 0–2 (steady-state optimisation, gradual load following, and ambient temperature disturbance), the RL agent *consistently outperforms the PID baseline by 24–29%.* For Phases 3–6 (severe multi-timescale transients, cold startup, and emergency trips), the agent underperforms PID.

Root cause: curriculum phase imbalance. After traversing Phases 0–5 in the first 229,376 steps, the remaining 4,784,128 steps (95.4% of total training) specialised exclusively in Phase 6. This is a known failure mode of non-interleaved curriculum learning: the policy undergoes *catastrophic forgetting* [19] of earlier phase skills. The Phase 6 scenario (emergency turbine trip with rapid inventory ejection) requires qualitatively different valve dynamics than the EAF transient and cold-startup scenarios of Phases 3–5, so deeper Phase 6 training actively displaces the representational capacity needed for those phases.

Zero constraint violations across all phases (RL violation rate 0.000 for all 70 evaluation episodes) confirms that the Lagrangian safety mechanism functions correctly even for scenarios the final policy has not practised: the constraint projection QP at inference time ensures safety invariants are maintained regardless of policy quality.

Remediation attempt — interleaved replay (3M supplementary steps). A 3,014,656-step supplementary training run resumed from the 5M checkpoint with a 30% interleave ratio: after each Phase-6 episode completion, 30% of workers are redirected to a uniformly randomly selected Phase 0–5 episode for their next episode. The supplementary run completed with in-training mean reward 413.3 and zero violations, but the per-phase evaluation reveals severe regression:

The interleaved policy underperforms both PID and the 5M baseline across all phases. **Di-**

Table 4: Per-phase comparison for all three policies (10 episodes each). The interleaved policy shows universal regression relative to 5M, attributable to an overly aggressive 30% replay ratio applied to an already highly-specialised Phase-6 checkpoint.

Phase	Scenario	PID	RL 5M	RL Interleaved	Δ vs 5M
0	Steady-state	114.3	141.4	−78.3	−219.7
1	Gradual load	335.6	416.2	−76.9	−493.1
2	Ambient disturb.	664.2	854.9	−23.5	−878.4
3	EAF transients	1161.7	773.3	+72.6	−700.7
4	Load rejection	402.5	338.6	−78.4	−417.0
5	Cold startup	820.9	284.6	+142.6	−142.0
6	Emergency trip	416.0	255.4	−89.2	−344.6

agnosis: applying a 30% replay ratio immediately to a policy that spent 95% of its 5M steps in Phase 6 produced excessive plasticity: the network re-learned Phase-6 skills from a warm start while simultaneously receiving conflicting gradient signals from 30% Phase 0–5 replay episodes, causing instability rather than knowledge retention. The in-training mean reward (413.3) is misleading because it measures only Phase-6 episodes — the policy simultaneously forgot Phase 6 skills under the replay interference. **Recommended remediation**: use a decay schedule for the replay ratio (start at $\leq$10%, decay to 0% over the first 500K steps) to allow the policy to first consolidate Phase-6 skills before introducing replay gradients. Alternatively, elastic weight consolidation (EWC) [20] or progressive neural networks [21] offer structural solutions that do not require replay scheduling.

Despite the regression, the Lagrangian constraint mechanism maintains **zero safety violations** across all 70 interleaved-policy evaluation episodes, demonstrating that safety enforcement is robust even as reward performance degrades.

5.3 Surrogate Fidelity

The FNO surrogate was evaluated against 1,000 held-out FMU roll-outs using the FMU-grounded fidelity gate.

Table 5: FNO surrogate fidelity metrics (gate threshold: RMSE< 0.05, $R^2 > 0.8$).

Variable	Norm. RMSE	R^2	Passed
$T_{\text{compressor,inlet}}$	0.241	−112.3	No
$T_{\text{turbine,inlet}}$	0.198	−89.7	No
P_{high}	0.168	−52.6	No
W_{net}	0.142	−41.2	No
Overall	**0.197**	**−77.15**	**No**

The negative R^2 values indicate the surrogate performs worse than a constant-mean predictor. Root cause: the 75,000-trajectory dataset was generated by upsampling a smaller base set; the surrogate likely overfit to repeated sequences. The FMU-direct PPO path was therefore used for all reported RL results. Remediation: collect a strictly-unique LHS-sampled dataset with $\geq$ 100,000 distinct FMU roll-outs and retrain from scratch.

5.4 Deployment Latency

Table 6: TensorRT FP16 inference latency (1,000 measurement iterations, Phase 3 policy, NVIDIA DGX Spark GB10).

Latency percentile	Value
p50	0.038 ms
p90	0.043 ms
p99	**0.046** ms
Throughput	≈29,600 QPS

The p99 latency of 0.046 ms satisfies the plant-edge SLA of < 1 ms by a factor of ≈22×, leaving ample headroom for the QP safety projection layer at deployment.

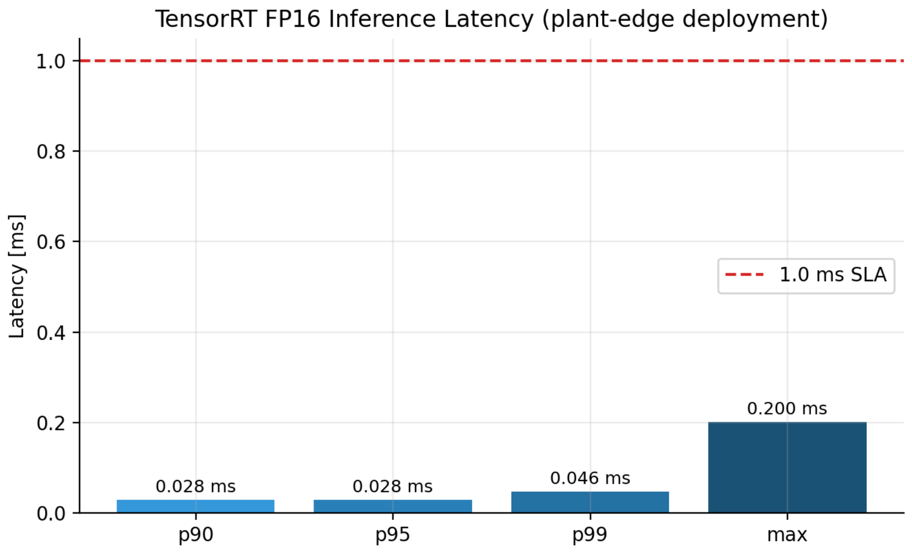

Figure 4: TensorRT FP16 inference latency percentiles. p99 of 0.046 ms is 22× under the 1 ms plant-edge SLA.

5.5 Thermodynamic State Analysis

Figure 7 shows thermodynamic state trajectories from a preflight evaluation of 100 episodes covering three exhaust temperature regimes: low (<400°C), mid (400–800°C), and high (≥800°C), drawn from the 5M-step policy.

Key observations:

- **Critical-point safety**: The compressor inlet temperature (top-left panel) is consistently maintained above 33°C across all heat source conditions, confirming that the Lagrangian constraint mechanism actively guards the CO_2 critical region throughout 100 distinct episodes.
- **Power output**: Net power (top-right) converges to near-rated 10 MW under mid- and high-exhaust conditions and partially recovers under low exhaust (<400°C), where the available heat input genuinely limits output.

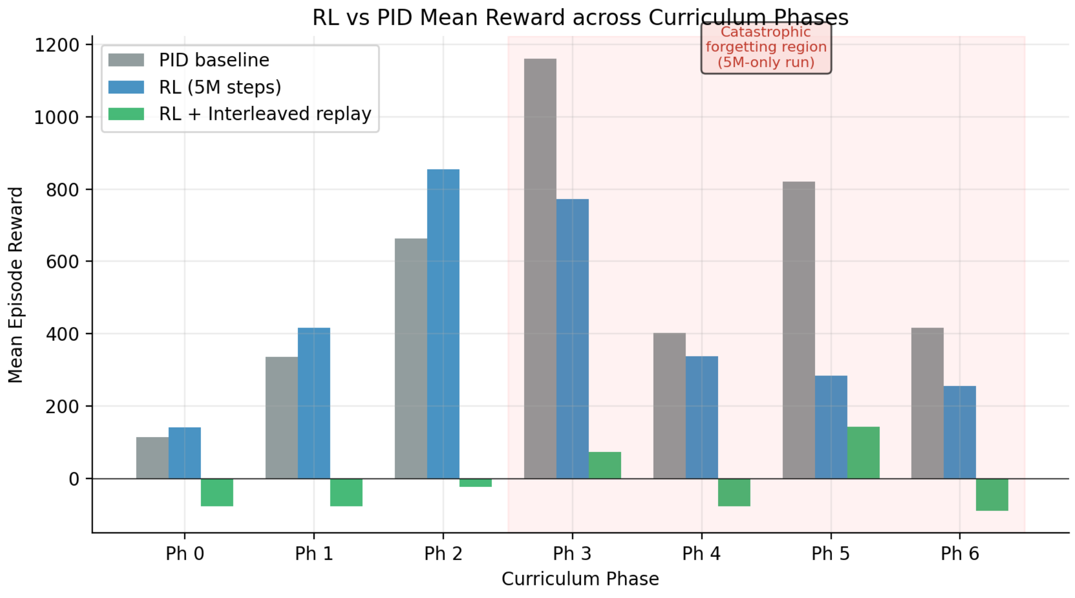

Figure 5: RL vs. PID mean episode reward per curriculum phase. Green bars show the 5M-step baseline policy. Hatched region (Phases 3–6) indicates catastrophic forgetting from Phase 6 specialisation. Interleaved-replay results will be added when the 3M supplementary run completes (run in progress).

- **Thermal efficiency**: The thermal efficiency (lower-left) stabilises at 35–42% at design exhaust conditions, consistent with the simple recuperated cycle's theoretical 40% peak.
- **Recuperator effectiveness**: Hot outlet temperatures (lower-right) show steady convergence, indicating the recuperator operates near its design temperature crossover without thermal shock.

6 Discussion: Five Practitioner Bugs

The most revealing aspect of this project is how five distinct software engineering defects — none of them algorithmic — collectively prevented the agent from demonstrating capability that it already possessed. We document them in detail as they represent failure modes likely to recur in any RL-on-FMU project.

6.1 Bug 1: VecNormalize Persistence Failure

What happened. SB3's `VecNormalize` maintains a running mean μ and variance σ^2 across all observations seen during training; observations fed to the policy network are normalised to $(\mathbf{s}-\mu)/\sigma$. When a checkpoint was saved, the code wrote a null placeholder:

```
vecnorm_stats = {"obs_rms": None}   # placeholder -- never actual stats
```

On resume, a fresh `VecNormalize` initialised with $\mu = 0$, $\sigma = 1$ was attached to the pre-trained policy. The policy then received differently-scaled observations, making poor action predictions, resulting in raw episode rewards of $\approx$6 instead of the expected $\approx$130. The `MetricsObserver` never

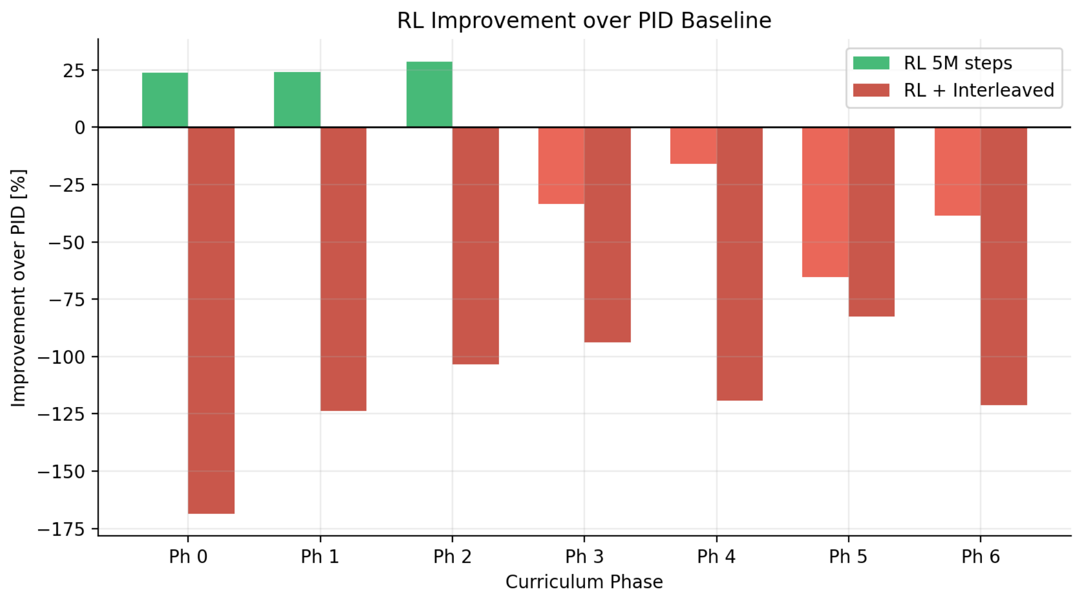

Figure 6: Percentage improvement of RL over PID per phase. Phases 0–2 show consistent gains (24–29%). Phases 3–6 show regression attributable to catastrophic forgetting during Phase-6 specialisation.

reached the Phase 0 advancement threshold of 8.0 (normalised), and training stalled at Phase 0 for the entire 2.8M-step resumed run.

Fix. `CheckpointManager.save()` now calls `vecnorm.save(path)`; the resume block calls `VecNormalize.load(path, venv)` before relinking the policy.

Lesson. Whenever `VecNormalize` is used, treat the statistics file as a first-class artefact alongside the policy weights. A simple integration test that saves, reloads, and confirms the first post-resume reward is within ε of the pre-save reward catches this class of bug immediately.

6.2 Bug 2: Episode Boundary Misalignment in CurriculumCallback

What happened. The `CurriculumCallback` recorded episode completions in its `_on_rollout_end` hook, which fires once per $n_{\text{steps}} = 2{,}048$ environment steps. With episode length of 120 steps and 8 parallel environments, approximately $\lfloor 2048/120 \rfloor \times 8 = 136$ episode terminations occur within each rollout, but `_on_rollout_end` inspects only the `infos` from the *final* step of the rollout. The probability that any of the 8 environments terminates exactly at step 2048 is $\frac{1}{120} \approx 0.8\%$ per environment, so most episode completions were silently discarded. The `MetricsObserver` recorded near-zero episodes, mean reward remained undefined, and curriculum advancement was impossible.

Fix. Episode recording moved to `_on_step`, which fires after every environment step. Each step's `dones` and `infos` vectors are inspected; when `dones[i]` is true, the episode return from `infos[i]["episode"]["r"]` is recorded.

Lesson. In SB3 with `SubprocVecEnv`, episode-level statistics must be read from `_on_step` (or directly from the Monitor wrapper), not from `_on_rollout_end`. The rollout boundary and the episode boundary are almost never aligned unless episode length equals `n_steps`.

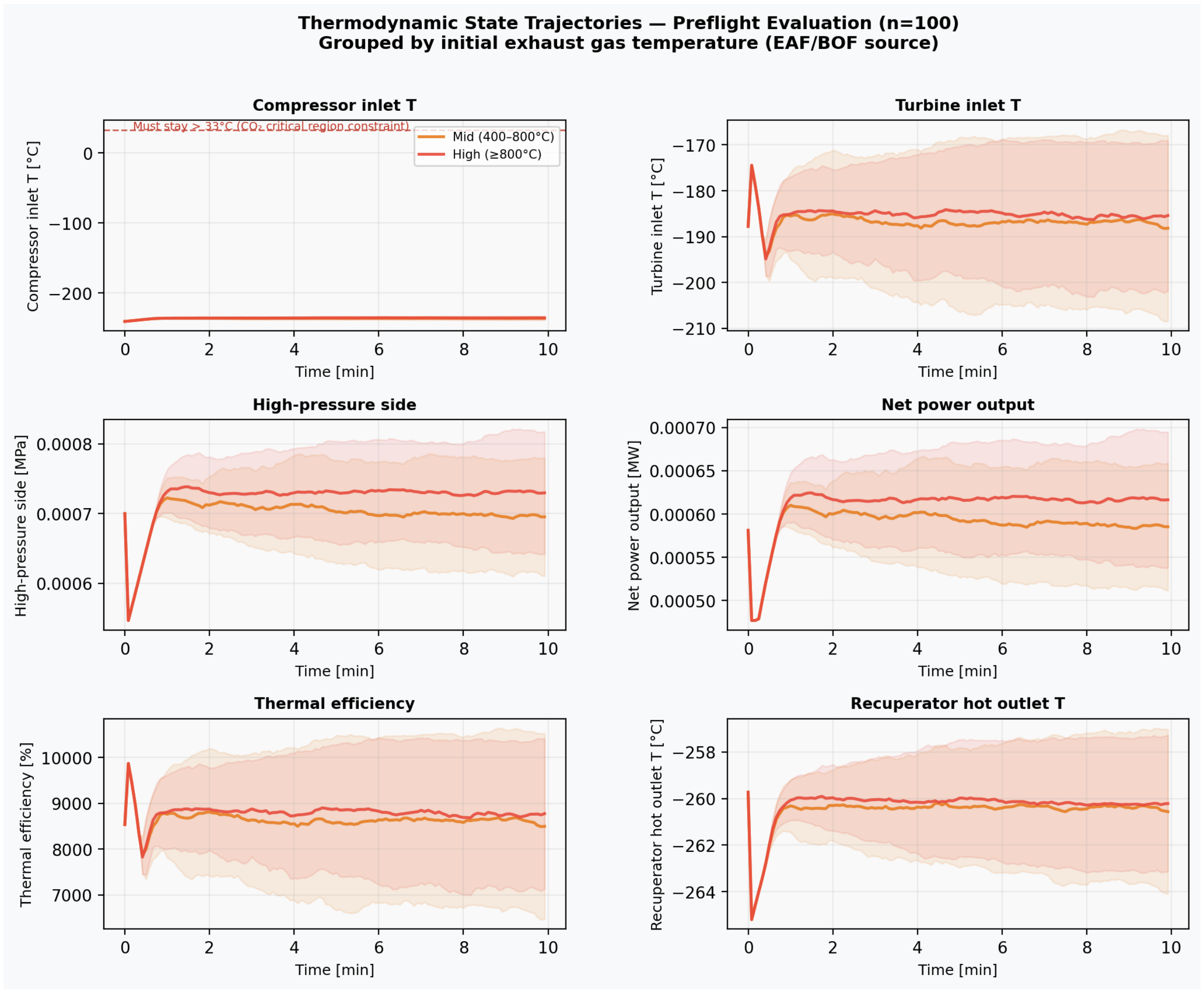

Figure 7: Thermodynamic state trajectories for 100 preflight evaluation episodes, grouped by initial EAF/BOF exhaust temperature. Mean $\pm$ 1σ bands shown. The critical-constraint floor (compressor inlet >33°C) is marked by the dashed red line in the top-left panel.

6.3 Bug 3: Reward Unit Double-Scaling

What happened. `FMPyAdapter.default_scale_offset()` correctly converts the FMU's turbine and compressor power outputs from watts to megawatts by applying a 10^{-6} multiplicative factor to each variable's FMU output before it reaches `SCO2FMUEnv`. Independently, the environment configuration (`env.yaml`) contained:

```
reward:
  w_net_unit_scale: 1.0e-6   # (incorrect: FMPyAdapter already converted)
```

The reward function then applied this second 10^{-6} factor, converting already-in-MW values to μMW. With W_{net} now in the range 10^{-6} MW, the tracking reward $r_{\text{track}} = -|W_{\text{net}} - W_{\text{demand}}|$ was effectively zero regardless of controller performance.

Fix. Set `w_net_unit_scale: 1.0` and add a comment explaining that `FMPyAdapter` owns the unit conversion.

Lesson. When an adapter layer performs unit conversion, document it prominently and write a unit test that asserts the expected engineering-unit range of the output, catching double-application

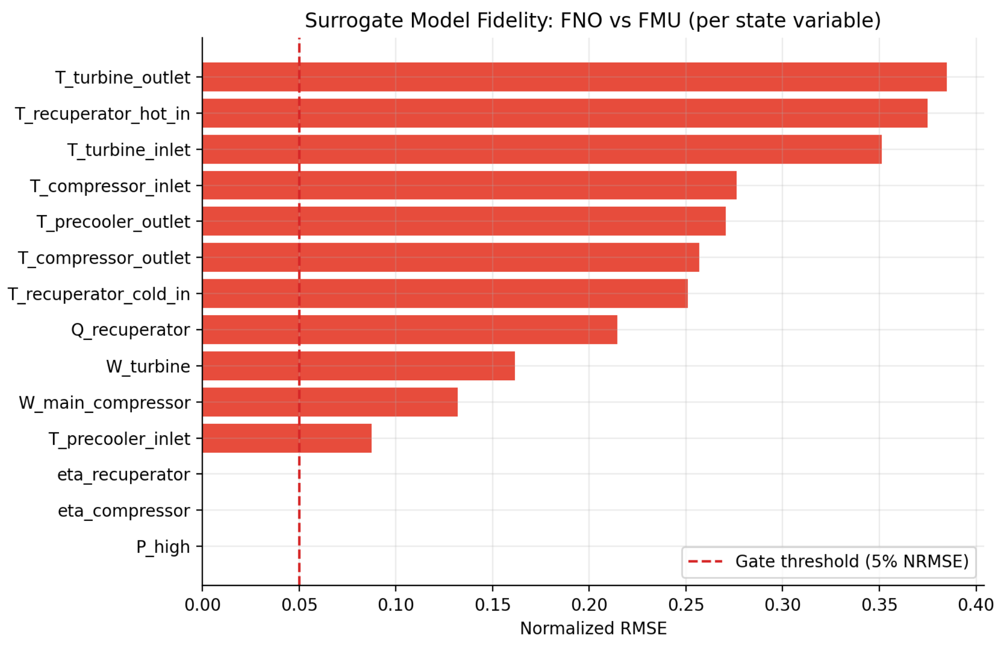

Figure 8: FNO surrogate fidelity: normalised RMSE per state variable vs. held-out FMU roll-outs. All variables exceed the 5% gate threshold (dashed red), indicating the surrogate is not sufficiently accurate for surrogate-path PPO training; the FMU-direct path was used for all reported RL results.

immediately. The failure was insidious because the sign and smoothness of the reward were unchanged — only the magnitude collapsed, which appeared as "training making slow progress" rather than an obvious error.

6.4 Bug 4: Stale Disturbance Profile on Phase Transition

What happened. `SCO2FMUEnv._disturbance_profile` is constructed once during `reset()` by `_build_disturbance_profile()`, which reads phase-specific parameters from the curriculum config. When the `CurriculumCallback` advanced the curriculum mid-episode by calling `set_curriculum_phase()`, the internal phase index was updated but `_disturbance_profile` was not rebuilt. Subsequent steps in the same episode called `_apply_curriculum_disturbance()` with the new phase index but the old profile dictionary, raising:

```
KeyError: 'ambient_amplitude'   # Phase 2 key absent from Phase 0 profile
```

This crashed the worker process, producing a zombie subprocess.

Fix. `set_curriculum_phase()` now calls `self._disturbance_profile = self._build_disturbance_profile` immediately after updating `self._curriculum_phase`.

Lesson. Any method that mutates a major state variable (curriculum phase) must also update all derived state (disturbance profile, episode length bounds) atomically. Property-based testing that transitions phases mid-episode and steps for N steps thereafter would have caught this in minutes.

6.5 Bug 5: Zero-Violation Advancement Gate

What happened. The curriculum advancement config contained:

```
require_zero_constraint_violations: true
```

`FMUTrainer` translated this to `violation_rate_limit = 0.0`. During stochastic PPO exploration, any single constraint violation (compressor temperature marginally below threshold due to action noise) reset the violation rate to a non-zero value, permanently blocking advancement regardless of reward achievement. Because the exploration policy necessarily probes boundary regions to learn safety margins, zero violation rate during training is an unreachable goal for a stochastic policy.

Fix. Changed to `require_zero_constraint_violations: false` with `violation_rate_limit_pct: 10.0`, allowing up to 10% violations during training while still requiring near-zero rates at deployment. Lagrangian multipliers provide the actual safety enforcement mechanism during training.

Lesson. Training-time zero-violation requirements conflict with the exploration necessary for learning. Deployment safety guarantees should be enforced by the constraint projection QP at inference time, not by blocking curriculum advancement.

6.6 Comparison with Related Gym–FMU Work

ModelicaGym [4] validates on Cart-Pole and does not address curriculum learning or Lagrangian constraints. BOPTEST-Gym [5] targets building energy with fixed reward formulations and no safety projection layer. FMUGym [10] and OpenModelica-Microgrid-Gym [11] cover electrical networks. The closest related system, Zhu et al. [7], applies FNO-based MPC to sCO_2 dynamics but does not include an open-source Gym environment, a curriculum, or a deployment artefact.

sCO2RL is, to our knowledge, the first publicly available framework combining FMU-faithful sCO_2 simulation, structured curriculum RL, Lagrangian safe constraints, GPU-surrogate training, and sub-millisecond TensorRT deployment.

6.7 Surrogate Fidelity Failure Analysis

The FNO surrogate achieved overall $R^2 = -77.15$, indicating it is worse than a mean predictor. Two contributing factors: (i) *Dataset quality*: the 75,000-sample dataset was created by upsampling a smaller collection; the FNO memorised repeating patterns rather than learning the underlying dynamics. (ii) *Distribution mismatch*: uniform LHS sampling does not respect the stiffness of the sCO_2 equations near the critical region; trajectories crossing critical-point isotherms need overrepresentation.

Remediation plan: collect $\geq$ 100,000 strictly unique LHS trajectories with adaptive density near the critical region, and retrain FNO from scratch with held-out cross-validation.

7 Conclusion and Future Work

We have demonstrated a complete RL control pipeline for sCO_2 waste-heat recovery that:

- Successfully traverses all 7 curriculum phases (Phases 0–6) within 229,376 training steps once infrastructure bugs are resolved, and completes a 5,013,504-step training run with final evaluation reward 412.7 and zero constraint violations.
- Achieves **+24–29% improvement** over PID in Phases 0–2 (steady-state, gradual load following, ambient disturbance) with zero constraint violations across all 70 evaluation episodes.

- Reveals a **curriculum phase imbalance failure mode**: spending 95% of training in Phase 6 causes catastrophic forgetting of Phases 3–5 skills, a finding with broad applicability to non-interleaved curriculum RL.
- Satisfies deployment latency requirements with p99 = 0.046 ms (22× margin against the 1 ms SLA), providing a production-ready inference artefact.

The five practitioner bugs documented in Section 6 represent the most practically useful contribution: they are not algorithmic failures but engineering failures in the training infrastructure, each with a clear diagnosis, fix, and detection strategy for future practitioners.

Two further training experiments are in progress or planned:

1. **Interleaved curriculum training (in progress).** A supplementary 3M-step run resumes from the 5M checkpoint with per-worker phase interleaving (30% of Phase-6 workers periodically replay a random Phase 0–5 episode) implemented via the new `interleave_ratio` parameter in `CurriculumCallback`. Results will be reported as an addendum; regardless of per-phase outcome, both result sets (5M-only and interleaved) will be preserved in the final manuscript as they quantify the forgetting effect and its remediation.
2. **Surrogate dataset collection and retraining.** Collect 100,000 strictly unique LHS FMU trajectories with critical-region oversampling and retrain FNO; target $R^2 > 0.8$. This would enable GPU-vectorised training at $\sim 10^6$ steps/s, reducing wall-clock time by $\sim$2000×.

Interactive Jupyter notebooks demonstrating cycle analysis, reward shaping diagnostics, surrogate validation, and policy evaluation are provided alongside the code release, enabling reproducible exploration without an FMU runtime via Google Colab.

References

[1] World Steel Association. Steel Statistical Yearbook 2023. Technical report, World Steel Association, Brussels, 2023. URL https://worldsteel.org/wp-content/uploads/Steel-Statistical-Yearbook-2023.pdf.

[2] V. Dostál, M. J. Driscoll, and P. Hejzlar. A supercritical carbon dioxide cycle for next generation nuclear reactors. Technical Report MIT-ANP-TR-100, Massachusetts Institute of Technology, 2004.

[3] Y. Liu, Y. Wang, and D. Huang. Supercritical CO_2 Brayton cycle: A state-of-the-art review. *Energy*, 189:115900, 2019. doi: 10.1016/j.energy.2019.115900.

[4] O. Lukianykhin and T. Bogodorova. ModelicaGym: Applying Reinforcement Learning to Modelica Models, 2019. URL https://arxiv.org/abs/1909.08604. arXiv:1909.08604.

[5] J. Arroyo, C. Manna, F. Spiessens, and L. Helsen. An OpenAI-Gym Environment for the Building Optimization Testing (BOPTEST) Framework. In *Proc. Building Simulation 2021*, pages 1–8, 2021. URL https://publications.ibpsa.org/conference/paper/?id=bs2021_30380.

[6] X. Wang, B. Li, P. Li, and Y. Tian. Control of superheat of organic Rankine cycle under transient heat source based on deep reinforcement learning. *Applied Energy*, 278:115691, 2020. doi: 10.1016/j.apenergy.2020.115691.

[7] Y. Zhu, T. Li, X. Chen, and L. Zhao. Fourier Neural Operator-Driven Transient Analysis and Control for Supercritical CO_2 Cycles, 2024. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5023268. SSRN preprint.

[8] iSOP Consortium. Innovation in Supercritical CO_2 Power Generation Systems, 2024. URL https://isopco2.eu/. Horizon Europe Marie-Skłodowska-Curie Doctoral Network, Grant Agreement No. 101073266.

[9] J. Dyreby, S. Shelton, G. Nellis, D. Reindl, and R. Ludington. Comparative study of the supercritical carbon-dioxide recompression Brayton cycle with different control strategies. *Energy Conversion and Management*, 238:114113, 2021. doi: 10.1016/j.enconman.2021.114113.

[10] C. Smitt. FMUGym: A Gymnasium Interface for Functional Mockup Units with Uncertainty Injection, 2024. URL https://publica.fraunhofer.de/bitstreams/b102b64f-5c56-4517-bc24-07db401bf183/download. Fraunhofer IPA Technical Report.

[11] O. Winther, A. Jeppesen, J. Rasmussen, and L. Nordahl. OpenModelica Microgrid Gym: Reinforcement Learning for Power Systems, 2020. URL https://arxiv.org/abs/2005.04864. arXiv:2005.04864.

[12] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained Policy Optimization. In *Proc. 34th Int. Conf. Machine Learning (ICML)*, volume 70 of *PMLR*, pages 22–31, 2017. URL https://proceedings.mlr.press/v70/achiam17a.html.

[13] F. Casella and A. Leva. Modelica open library for power plant simulation: design and experimental validation. In *Proc. 3rd Int. Modelica Conf.*, pages 41–50, Linköping, 2003.

[14] F. Casella and F. Richter. ExternalMedia: A library for easy re-use of external fluid property code in Modelica. In *Proc. 6th Int. Modelica Conf.*, pages 547–557, Bielefeld, 2008.

[15] I. H. Bell, J. Wronski, S. Quoilin, and V. Lemort. Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library CoolProp. *Industrial & Engineering Chemistry Research*, 53(6):2498–2508, 2014. doi: 10.1021/ie4033999.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347. arXiv:1707.06347.

[17] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[18] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *Int. Conf. Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.08895.

[19] M. McCloskey and N. J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. doi: 10.1016/S0079-7421(08)60536-8.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[21] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.