A Stable Gray-Box Surrogate for Engine Spool Dynamics in Real-Time Propulsion Simulation

Massimo Di Pierro
University of California, Santa Cruz, CA 95064, USA

2026

Abstract

Real-time vehicle simulation must evaluate propulsion forces at every integration step. A physics-based engine deck (a component-level thermodynamic cycle model) is accurate but heavy and intrusive to embed in a closed loop, while the usual interpolated performance map is fast but *quasi-static*: it returns a steady thrust or mass flow for the instantaneous command and ignores the engine's spool dynamics — the lagged, history-dependent rise and fall of thrust, mass flow, and shaft speed that dominate throttle transients, starts, and shutdowns. We present a compact gray-box surrogate with a Wiener structure: a bank of learnable second-order IIR filters supplies stable, history-dependent dynamics; a small multilayer perceptron supplies the nonlinear command-to-output map; and an explicit linear path carries near-linear effects. The filter poles are parameterized as conjugate pairs of radius $r = \sigma(\rho) < 1$, so the linear dynamics are stable by construction and re-checked before deployment. The model has fixed per-step cost, scales linearly with the number of command channels, and is a trainable nonlinear extension of the block-oriented models used in gas-turbine and liquid-rocket system identification. We verify it on a synthetic engine with known spool-up dynamics — reproducing the lagged step response and the throttle-versus-thrust loop — and validate it on *measured* turbofan data from the NASA C-MAPSS family: the static command-to-response map against the C-MAPSS fleet (held-out engines, $R^2 \geq 0.998$), and the *unsteady* spool dynamics against a held-out real flight from N-CMAPSS, where the filter memory roughly halves the prediction error of a memoryless map.

Keywords: engine modeling; spool dynamics; real-time simulation; system identification; Wiener model; block-oriented models; gas turbine; liquid rocket engine; machine learning.

1 Introduction

A real-time vehicle simulation must compute the propulsion force on the airframe at every integration step — often hundreds to thousands of times per second, and no slower than wall-clock time so the loop can close around a flight computer or a pilot. Resolving the engine directly with a component-level thermodynamic cycle model (an "engine deck") at each step is accurate but expensive and awkward to embed in a closed loop, and a high-fidelity transient model of a liquid-rocket or gas-turbine engine is heavier still. The propulsion system must therefore be reduced to a cheap surrogate evaluated from a handful of command and condition variables.

The standard surrogate is an interpolated lookup table (a performance map) of thrust, propellant mass flows, and shaft speed as functions of throttle or valve commands, ambient or flight condition, and a start/mode flag, with dimensional loads following directly. Tables are fast, but they scale badly. Their memory grows exponentially with the number of indexing variables: a thrust

that depends on throttle, two propellant valves, chamber state, and altitude quickly becomes an unmanageable high-dimensional grid, and the accuracy achievable is limited by the resolution one can afford to store. Worse, a table is *quasi-static* — it returns an output for the instantaneous command — whereas a real engine's output depends on the recent *history* of its commands: thrust and mass flow lag a throttle step as the turbomachinery spools up, shaft speed overshoots and settles, and starts and shutdowns are slow transients with their own time constants. Capturing such spool dynamics in a table means adding rate or history dimensions, compounding the memory blow-up; a plain neural network of the instantaneous command fares no better, as neither has any internal memory.

The research question addressed here is whether a small, explicitly stable dynamical surrogate can capture the dominant history dependence of an engine's response while remaining cheap enough for real-time simulation. We seek a model that is (i) fast and fixed-cost per step, (ii) compact in memory and graceful as the number of command channels grows, (iii) able to represent unsteady, history-dependent outputs (spool-up, spool-down, mode transitions), (iv) numerically stable inside a closed-loop simulator, and (v) fittable directly from logged test-stand, flight, or high-fidelity-simulation data. The proposed model is a *Wiener* block structure: a bank of learnable second-order IIR filters (biquads) supplies the spool dynamics, and a small multilayer perceptron supplies the static nonlinear map to every output. The filters carry the command history in a few internal registers, so memory is represented *natively* rather than tabulated; the model is a few hundred parameters in the configurations shown below rather than a sampled grid over the operating envelope, and its cost grows linearly — not exponentially — with the number of commands. The filter poles are parameterized so that the linear dynamics are stable by construction.

The contribution is a deliberately propulsion-oriented synthesis. Differentiable biquad filters are used as the stable linear block of a Wiener model for engine output prediction, and the resulting structure is interpreted as a trainable nonlinear extension of the block-oriented models long used in engine system identification. The paper states the runtime checks needed to make the stability claim meaningful in a deployed simulation, verifies the approach on a controlled synthetic engine, and — unlike a purely synthetic study — validates it against *measured* turbofan data in both the static and the unsteady regime.

2 Engine-modeling and identification background

Engine performance maps and cycle models. Real-time propulsion is classically modeled either by interpolated steady-state performance maps or by component-level cycle models such as the Numerical Propulsion System Simulation (NPSS) [1] and, for transient health and control studies, the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [2]. Cycle models capture transients faithfully but are heavy; steady maps are cheap but quasi-static. The present surrogate sits between them: it is map-cheap at runtime yet, like a transient cycle model, carries the spool dynamics in internal state.

Block-oriented system identification. Block-oriented models — a static nonlinearity in series with a linear dynamic block (Hammerstein, Wiener, and their cascades) — are a standard tool for identifying nonlinear dynamical systems from input–output data [3], and have long been applied to gas-turbine engines and to liquid-rocket transient behaviour, where throttle and valve commands drive lagged, nonlinear responses that reach steady state after seconds [4]. The model here is a Wiener structure (linear dynamics then static nonlinearity) in which the linear block is a bank of second-order rational filters and the nonlinearity is a small network; its parameters are fit by

gradient descent and its poles are constrained stable.

Differentiable IIR filters (audio). The stable, differentiable linear block is borrowed from neural audio-effect modeling. Nercessian [5] introduced differentiable biquads to match a target equalizer curve by gradient descent. Nercessian, Sarroff and Werner [6] extended it to nonlinear effect modeling — a cascade of differentiable biquads followed by a nonlinearity, with stability-ensuring (conjugate-pole) activations and conditioning on an effect's external controls. Colonel et al. [7] design biquad cascades directly by deep learning, and the line continues to time-varying all-pole filters [8]. Our linear block and its conjugate-pole stability parameterization follow this work; the present paper transfers it from audio to propulsion.

Data-driven dynamics. Data-driven reduced-order and surrogate models of nonlinear dynamical systems — including the use of neural networks and system-identification methods to learn input–output dynamics from data — are surveyed by Brunton and Kutz [9]. Against a black-box recurrent network, the model here trades representational generality for three concrete properties: denominator stability by construction, a small interpretable parameter set whose poles correspond to physical spool time constants, and a fixed-cost real-time forward pass.

3 Model

3.1 Structure

Let the input at discrete time n be $u[n] \in \mathbb{R}^N$ — e.g. a throttle, two propellant valve commands, and a hot/cold-start flag, or any set of command and condition channels — and the target be the output vector $y[n] \in \mathbb{R}^K$, e.g. $(\text{thrust}, \dot{m}_{\text{ox}}, \dot{m}_{\text{fuel}}, \text{shaft speed})$. The model is a Wiener block model (Fig. 1).

Input standardization. Each input channel is first standardized to zero mean and unit variance, $\hat{u}_i[n] = (u_i[n] - \mu_i)/\sigma_i$, using statistics (μ_i, σ_i) computed from the training data and stored with the model. This balances the channel scales so the optimizer does not let a large-variance input dominate the gradient and starve smaller signals. The filters, the skip path, and the linear path below all act on $\hat{u}$.

Linear dynamics. Each standardized input channel i feeds a bank of M biquad filters. Filter f on channel $c(f)$ produces a Direct-Form-I second-order IIR response,

$$g_f[n] = b_{0f}\hat{u}_{c(f)}[n] + b_{1f}\hat{u}_{c(f)}[n-1] + b_{2f}\hat{u}_{c(f)}[n-2] - a_{1f}g_f[n-1] - a_{2f}g_f[n-2]. \tag{1}$$

With N inputs and M filters per input there are $F = NM$ filtered features. The bank is parallel: the filters do not feed one another; they produce a set of learned dynamical basis signals that are concatenated before the static readout. A single second-order section already captures a lagged, possibly overshooting spool response; several per channel cover the mix of time scales in a real engine.

Static nonlinearity and linear path. The feature vector

$$v[n] = (g_0[n], \ldots, g_{F-1}[n], \hat{u}_0[n], \ldots, \hat{u}_{N-1}[n]) \in \mathbb{R}^D, \qquad D = F + N$$

(filtered features plus the standardized inputs) is mapped by a one-hidden-layer MLP with H tanh units; an explicit *linear path* $L \in \mathbb{R}^{K\times N}$ adds a direct linear dependence of every output on the inputs:

$$h_j[n] = \tanh\Big(b^1_j + \textstyle\sum_d W^1_{jd} v_d[n]\Big), \qquad o_k[n] = b^2_k + \textstyle\sum_j W^2_{jk} h_j[n] + \sum_i L_{ki}\hat{u}_i[n], \tag{2}$$

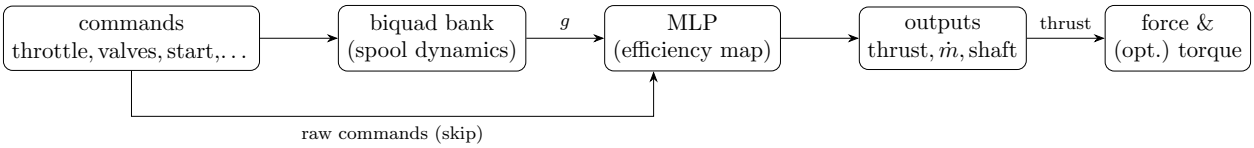

Figure 1: Proposed engine-output pipeline: learnable biquads (spool dynamics) feed a small MLP (static efficiency / output map); the raw commands also skip to the MLP. The predicted thrust can optionally be applied to a rigid body as a (gimbaled) force/torque provider.

each de-standardized as $y_k = o_k s_k + m_k$. The linear path carries the near-linear part of the command-to-output map so the MLP is left to model the residual nonlinear, history-dependent part rather than competing with a large linear term. The filter bank is *shared* across outputs: one internal spool state drives every output, with a per-output static readout. Including the standardized inputs alongside the filtered features lets a purely static input (e.g. a discrete hot/cold-start flag) affect the map without passing through the dynamics: it rides along as a raw feature and modulates the static efficiency map, not the shared filter dynamics.

Derived outputs. Quantities that are algebraic functions of the predicted outputs — for a bipropellant engine, the mixture ratio $\dot{m}_{\text{ox}}/\dot{m}_{\text{fuel}}$ — are computed from the network outputs rather than predicted by their own head, so they stay exactly consistent with the mass flows the model predicts.

3.2 Stability by construction

Each filter carries two free parameters (ρ_f, θ_f) rather than the denominator coefficients directly, with the conjugate-pole map

$$r_f = \sigma(\rho_f) \in (0, 1), \quad a_{1f} = -2r_f \cos\theta_f, \quad a_{2f} = r_f^2, \tag{3}$$

σ the logistic sigmoid. The poles are then a conjugate pair of radius $r_f < 1$, so every filter is strictly stable for all parameter values: no projection or clamping during training. At runtime the materialized coefficients are also validated by the corresponding discrete-time pole-stability condition before the coefficient file is accepted. This is a property not generally guaranteed by an unconstrained black-box recurrent reduced-order model, and it is essential for a model running in closed loop inside a flight simulator, where an unstable surrogate would inject divergent thrust into the integrator.

3.3 From outputs to forces and torques

In its predictor role the model simply publishes the learned outputs (thrust, mass flows, shaft speed) and any derived ratios as scalar channels. In a *provider* role it additionally applies the predicted thrust to a rigid body as a force, so the learned thrust enters the integrator exactly like an analytic thruster would. The thrust is clamped non-negative (an engine cannot pull) and applied along a chosen body axis; with two gimbal-deflection command channels and an attach point r, the thrust direction is deflected and the offset produces a control torque $r \times F$, letting a learned engine tilt a 6-DOF vehicle exactly like a gimbaled thruster. The dynamical outputs are therefore the learned scalar channels, and the dimensional force and torque follow from the predicted thrust and the geometry.

4 Training

Given a time-ordered dataset of commands and measured outputs, parameters are fit by minimizing the per-output-standardized mean squared error

$$\mathcal{L} = \frac{1}{TK} \sum_{n=1}^{T} \sum_{k=1}^{K} (o_k[n] - y_k[n])^2, \tag{4}$$

with Adam, where T is the number of time samples in the training sequence. The MLP gradients are standard backprop. The biquad gradients use the adjoint (backpropagation-through-time) recursion for (1),

$$\lambda[n] = \frac{\partial \mathcal{L}}{\partial g[n]} - a_1 \lambda[n+1] - a_2 \lambda[n+2], \tag{5}$$

from which the numerator and denominator gradients accumulate as sums of $\lambda[n]$ against delayed inputs and outputs, and the chain rule through (3) gives gradients in the free parameters (ρ, θ). Our analytic gradients agree with central finite differences to better than 10^{-9} (Section 7), confirming the hand-derived backprop-through-time. Because the filters carry state, examples are treated as time-ordered sequences rather than shuffled independent samples. At the start of a sequence the filter histories are reset; an initial warm-up interval can be excluded from the loss when arbitrary initial conditions would otherwise dominate the fit. Validation should use held-out trajectories or contiguous time intervals rather than randomly sampled points, to avoid leakage through the temporal state. The per-channel input-standardization statistics are computed from the training data and stored with the model, so the runtime is supplied inputs in physical units and standardizes them internally. Training is offline; the runtime is inference only.

5 Real-time deployment model

For use in a flight-mechanics simulation, the learned engine model is wrapped by a deterministic runtime layer. At each simulator step the runtime reads the declared command channels, advances the filter states once using Eq. (1), evaluates the MLP in Eq. (2), publishes the predicted outputs and any derived ratios, and — in provider mode — applies the predicted thrust (and gimbal torque) to the target body. The runtime state consists only of the two-sample input and output histories of each biquad; these histories are the only quantities that must be saved and restored across simulator checkpoints.

The deployment constraints are intentionally conservative. The number of inputs, outputs, filters, and hidden units is fixed when the coefficient file is loaded, and the forward pass uses fixed-size buffers with no allocation or interpolation search. The materialized denominator coefficients are checked for discrete-time stability before use. These checks separate the mathematical stability of the trained parameterization from the engineering problem of accepting a serialized coefficient file in a real simulator.

The cost and memory of the deployed model are small and, crucially, *independent* of how finely the operating envelope is sampled. The synthetic model of Section 7 stores about 400 parameters (a few kB at double precision) and a runtime state of four samples per biquad; each step costs a few hundred multiply–add operations and $H = 16$ hyperbolic-tangent evaluations, with no table lookup or interpolation search. By contrast, a lookup table of a single output over d command/condition arguments at g points per axis stores g^d entries: even a coarse $g = 20$ grid over four arguments (throttle, two valves, and a chamber state) is 1.6×10^5 entries per output, and representing the

spool dynamics by adding rate or history axes multiplies this further. This contrast, together with the ablations below, is the practical case for a compact learned dynamical model when an output depends on many arguments and on their history.

Although the prototype was implemented in a compiled simulator plugin, no particular simulator architecture is required. The equations above fully define the inference kernel, and the same model can be embedded in a flight-dynamics code, hardware-in-the-loop environment, or batch trajectory evaluator.

6 The HiroSim simulation platform

The reference implementation of the model targets HiroSim, a new simulation platform for aerospace, robotics, and industrial applications. HiroSim provides a deterministic, closed-loop runtime organized around a few orthogonal concepts. Simulated time advances with nanosecond precision. The dynamics are assembled from *pluggable models* — shared-library components that register one or more model types at load time — and each model runs at its own update rate, so a fast inner loop (a controller, or the engine model of this paper) and a slower outer process coexist in a single schedule. Numerical integration is likewise pluggable. Every model variable is published as *telemetry*, and the runtime accepts *commanding* while a run is in progress, including a save/restore facility that snapshots and reinstates the complete simulation state.

Two further features make HiroSim a flight-software development environment rather than only a physics sandbox. It includes a Python-like flight-software specification language in which control logic is expressed, and a hardware abstraction layer that decouples flight-computer code from its inputs and outputs. As a result the *same* flight software runs unchanged against the simulator's models or against real sensors and actuators on target hardware. The engine model of this paper is registered as a pluggable model type — fitted offline, loaded from a serialized coefficient file, and advanced each step by the nanosecond-precision scheduler alongside the vehicle dynamics and the flight software it closes the loop with. In a companion example a learned, gimbaled engine flies a 6-DOF lander on a hop-and-land trajectory, driven only by the controller's throttle and gimbal commands.

7 Verification: synthetic engine with known spool dynamics

The first study uses a synthetic generator with known dynamics. This case should be read as verification of the model structure, gradient calculation, and real-time behaviour, not as validation against measured data (Sections 8–9 do that). It is useful because the desired spool-up lag and rate effects are known in advance and can be checked independently of measurement noise.

The generator maps three commands — two propellant valves and a hot/cold-start flag — to four outputs (thrust, two mass flows, shaft speed). The mass flows follow a first-order *lag* of the valve commands (spool-up), the shaft speed is a second-order resonant filter of the drive (so it overshoots and settles), thrust is a saturating nonlinearity of the combined flow scaled by a hot/cold efficiency, and measurement noise is added. The lagged response is the memory the biquad bank exists to capture; the start flag is a static feature. Model dimensions and the held-out fit are in Table 1. The analytic gradients of the proposed model match central finite differences to 8.0×10^{-11}, confirming the backprop-through-time.

To confirm that the temporal dynamics are necessary rather than incidental, we refit the same data with the biquads frozen to unit pass-through, so that only the static MLP on the instantaneous commands is learned. Held-out standardized error rises by almost an order of magnitude — from

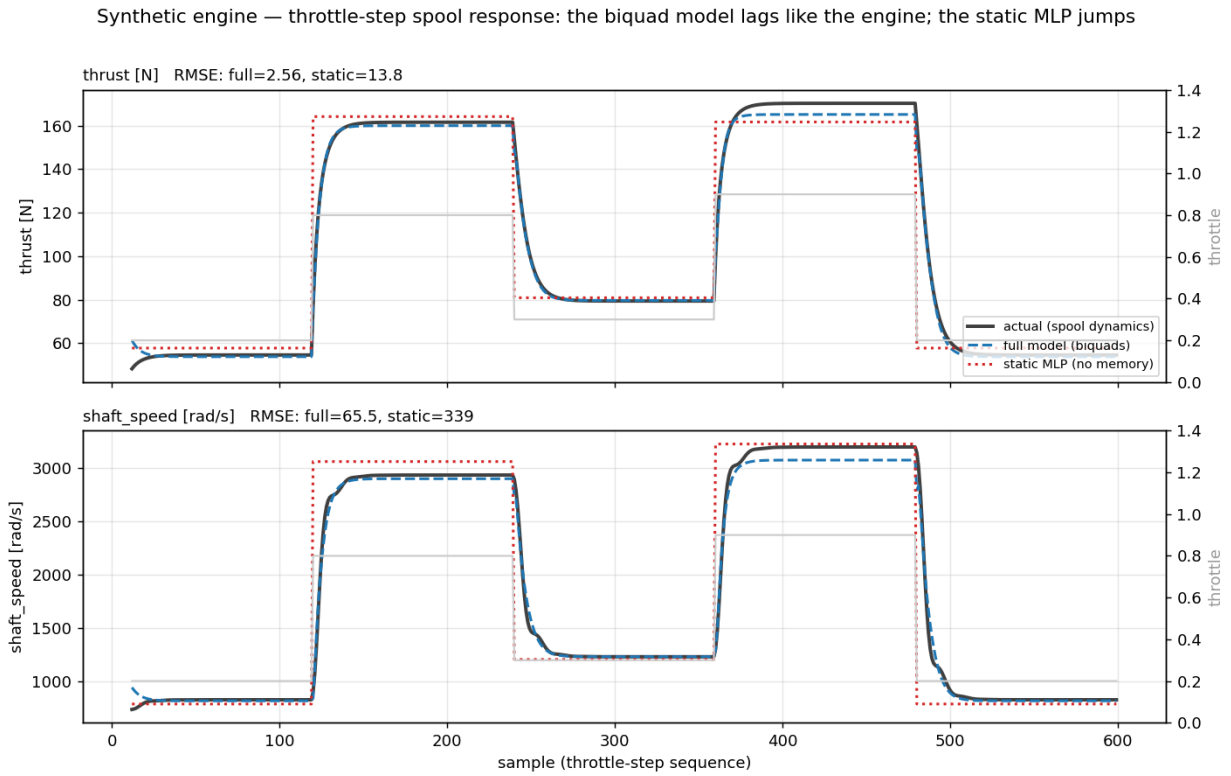

Figure 2: Synthetic-engine verification: throttle-step spool response. The full biquad-Wiener model (dashed) tracks the lagged spool-up and spool-down of thrust (top) and shaft speed (bottom); the memoryless static-MLP ablation (dotted) jumps instantly to each steady level. The grey trace is the throttle command (right axis).

1.8×10^{-2} to 1.7×10^{-1} (Table 1) — even though the static model has only about 15% fewer parameters. The memory supplied by the biquad bank, not the MLP nonlinearity, is what captures the spool lag; this is exactly the regime in which a quasi-static lookup table or a memoryless network is structurally unable to fit the data.

Figures 2 and 3 make this concrete. Driving the trained model through a clean throttle-step sequence (Fig. 2), the full model reproduces the lagged spool-up and spool-down of thrust and shaft speed, while the static-MLP ablation jumps instantly to each steady level and cannot represent the transient. Driving it through a sinusoidal throttle sweep (Fig. 3), the output traces a rate-dependent loop against the throttle — the engine analogue of a hysteresis loop, here caused by spool lag rather than aerodynamic memory; the full model follows the loop while the static MLP collapses it to a single-valued curve.

8 Validation I: measured static map (C-MAPSS fleet)

As a first check against *measured* engine data rather than a synthetic generator, the static command-to-response map is fit to the NASA C-MAPSS turbofan degradation dataset [10, 2] (subset FD002, which spans six real operating conditions). The three operating-condition variables — altitude, Mach number, and the throttle-resolver angle — drive four steady engine outputs: the HPC-outlet total temperature T_{30}, the physical fan- and core-shaft speeds N_f and N_c, and the HPC-outlet static

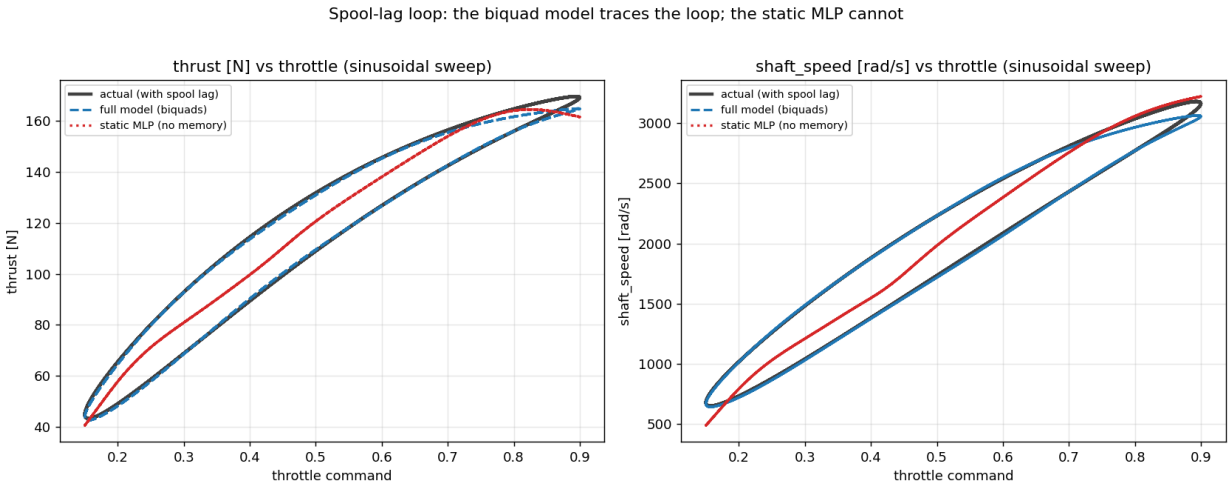

Figure 3: Synthetic-engine verification: output versus throttle on a sinusoidal sweep. Spool lag opens a rate-dependent loop (the engine analogue of a hysteresis loop). The full model (dashed) traces the loop; the memoryless static-MLP ablation (dotted) collapses it to a single-valued curve.

Table 1: Synthetic-engine experiment ($N = 3$ commands, $K = 4$ outputs, 4000 samples). MSE is standardized and averaged over the four outputs on a held-out 20% time tail. The static baseline freezes the biquads to unit pass-through, leaving only the MLP acting on the instantaneous commands; the proposed model also trains the biquad bank. The analytic gradients match central finite differences to 8.0×10^{-11}.

Configuration	Biquads	Trainable params	Held-out MSE
Static MLP (no memory)	0 (pass-through)	336	1.7×10^{-1}
Proposed (biquad dynamics)	12	396	1.8×10^{-2}

pressure P_{s30}. A C-MAPSS cycle is a single steady operating snapshot, so the experiment exercises the model's nonlinear MLP readout, not the spool memory: the model is used in its static-readout configuration (the biquads reduce to unit pass-through). We take a near-healthy slice (the first ten cycles of every train engine, where degradation drift is minimal) and hold out one engine in five *entirely*, so the test measures cross-engine generalization.

Figure 4 shows predicted versus measured outputs on the held-out engines. The static readout reproduces the measured steady response across all six operating conditions and generalizes to unseen engines essentially perfectly: held-out $R^2 = 0.998$ for T_{30}, 1.000 for N_f, 0.999 for N_c, and 0.998 for P_{s30}. This validates the nonlinear static map against real engine measurements; it does not exercise the spool memory, which is the subject of Section 9.

9 Validation II: measured spool dynamics (N-CMAPSS)

The central claim — that the biquad bank captures real engine *spool dynamics* — is tested on time-resolved measured data from N-CMAPSS, the run-to-failure turbofan dataset recorded under real flight conditions [11]. We take two consecutive near-healthy flights of one engine (unit 2 of subset DS02, sampled at 1 Hz). The model is trained on the first flight and evaluated on the second — a fully held-out flight that follows a different route, and therefore a different throttle and

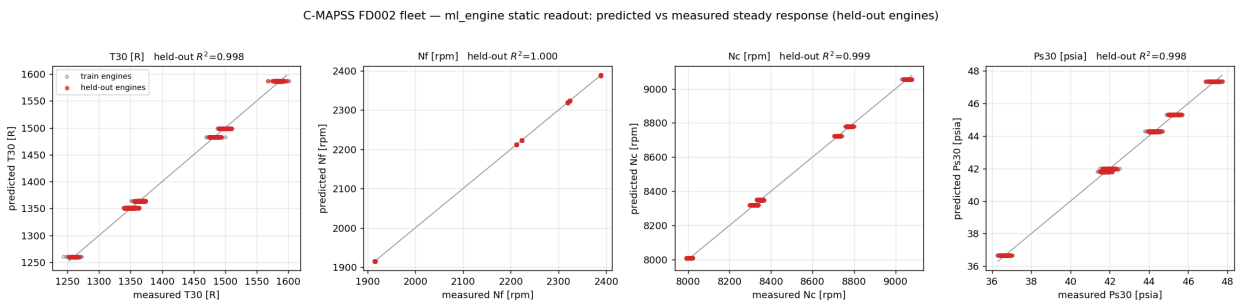

Figure 4: Validation on measured data (static map): predicted versus measured steady response for the C-MAPSS FD002 turbofan fleet across six operating conditions, on engines held out of training. Open circles are training engines, red points the held-out engines, the grey line is the identity. Held-out $R^2 \geq 0.998$ for every output.

altitude trajectory. The four inputs are the throttle-resolver angle, altitude, Mach number, and fan-inlet temperature; the three outputs are the fan- and core-shaft speeds N_f, N_c and the fuel flow W_f. Here the full unsteady model is used: the biquads carry the spool state across the flight. As a baseline we again refit with the biquads frozen (the memoryless static map).

Figure 5 shows predicted versus measured outputs on the held-out flight. The full biquad model tracks the real spool transients through climb, cruise, and the throttle excursions of descent. Averaged over the three outputs, the full model's held-out-flight standardized error is roughly half that of the memoryless map (3.3×10^{-2} versus 6.2×10^{-2}). The improvement is concentrated where the dynamics are strongest: for fuel flow the held-out R^2 rises from 0.869 (static) to 0.949 (full), while the core-shaft speed, which tracks the throttle almost quasi-statically at this sampling rate, is already well fit by the static map ($R^2 \approx 0.974$ for both). This is, to our knowledge, exactly the unsteady-on-measured-data validation that a synthetic study cannot provide: on a held-out real flight, the explicit filter memory roughly halves the prediction error of an otherwise identical memoryless surrogate.

10 Known limitations

The model is deliberately a *Tier-1* construction: the filter bank is shared and its poles are fixed across operating regimes. A static condition input modulates the *static* map but not the *dynamics*, so regime-dependent spool time constants — altitude- or speed-dependent acceleration schedules, or configuration-dependent start transients — are not captured. Two natural extensions address this: per-regime filter banks blended by a conditioning input, and hyperconditioning the filter parameters on external inputs as in [6]. The model also inherits the usual limitation of lookup tables and neural surrogates: extrapolation outside the training envelope should be guarded, and a production simulator should flag or reject out-of-envelope commands or blend to a conservative fallback. The measured-data validation here uses turbofan data because it is the public time-resolved engine data available; the synthetic case and the companion 6-DOF example exercise the same model on a throttled, gimbaled rocket, but a measured liquid-rocket transient dataset — ideally with start and shutdown sequences — would be the most demanding next test. A full evaluation should also add quantitative comparison against a component-level transient cycle model and against a black-box recurrent reduced-order model, and should test whether the learned poles recover physically meaningful spool time constants.

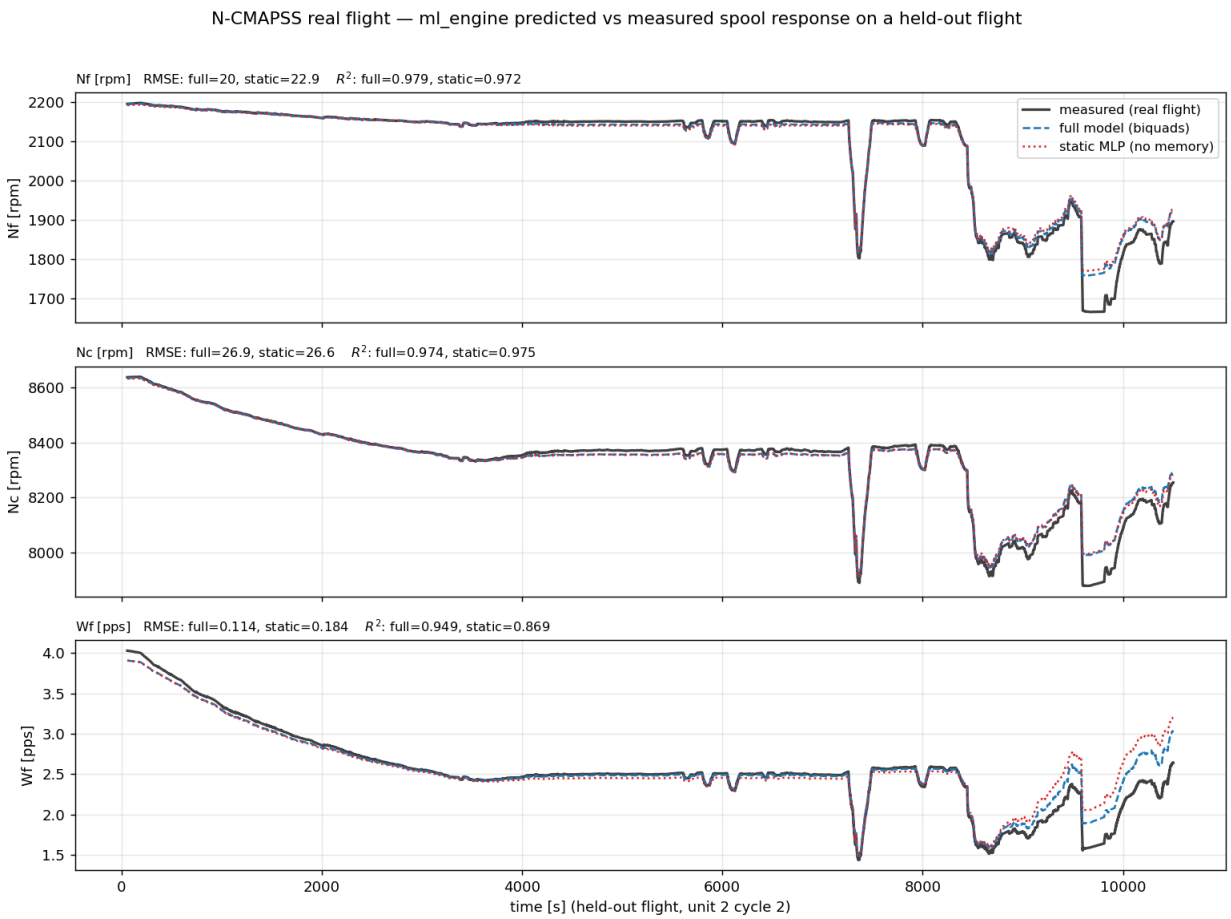

Figure 5: Validation on measured data (spool dynamics): predicted versus measured fan speed N_f, core speed N_c, and fuel flow W_f over a held-out N-CMAPSS flight (unit 2, cycle 2). The full biquad-Wiener model (dashed) tracks the real spool transients; the memoryless static-MLP ablation (dotted) lags and smears them, most visibly in the fuel flow. Per-output R^2 is annotated.

11 Conclusion

We presented a compact differentiable-biquad Wiener model for gray-box prediction of engine spool dynamics in real-time simulation. The model borrows stable differentiable-IIR machinery from signal processing and interprets it through a propulsion lens as a nonlinear, trainable extension of the block-oriented models used in engine system identification. The resulting surrogate has fixed per-step cost, a small internal state, explicit pole-stability constraints, and a direct path from learned thrust to dimensional loads. A synthetic engine verifies the training and runtime procedure, including the spool-up step response and the throttle loop; the nonlinear static readout is validated against the measured C-MAPSS turbofan fleet ($R^2 \geq 0.998$ on held-out engines); and — closing the gap a purely synthetic unsteady study would leave — the spool dynamics are validated against a held-out real flight from N-CMAPSS, where the filter memory roughly halves the error of a memoryless map. The next steps are a measured liquid-rocket transient (with starts and shutdowns) and direct comparison with transient cycle-model and recurrent-network baselines.

References

[1] J. K. Lytle, "The numerical propulsion system simulation: An overview," NASA/TM-2000-209915, 2000.

[2] D. K. Frederick, J. A. DeCastro, and J. S. Litt, "User's guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)," NASA/TM-2007-215026, 2007.

[3] F. Giri and E.-W. Bai (eds.), *Block-oriented Nonlinear System Identification*, Lecture Notes in Control and Information Sciences, vol. 404, Springer, 2010.

[4] G. Waxenegger-Wilfing, K. Dresia, J. C. Deeken, and M. Oschwald, "A reinforcement learning approach for transient control of liquid rocket engines," *IEEE Trans. Aerospace and Electronic Systems*, vol. 57, no. 4, 2021.

[5] S. Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, 2020.

[6] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *Proc. IEEE ICASSP*, 2021.

[7] J. Colonel, C. J. Steinmetz, M. Michelen, and J. D. Reiss, "Direct design of biquad filter cascades with deep learning by sampling random polynomials," in *Proc. IEEE ICASSP*, 2022.

[8] "Differentiable all-pole filters for time-varying audio systems," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, 2024.

[9] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, Cambridge University Press, 2019.

[10] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics and Health Management (PHM)*, 2008.

[11] M. Arias Chao, C. Kulkarni, K. Goebel, and O. Fink, "Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics," *Data*, vol. 6, no. 1, art. 5, 2021.

Declarations

Funding. No external funding was received for this work.

Competing interests. The author declares no competing interests.

Data availability. The synthetic verification case (Section 7) uses synthetic data generated by the equations described there. The static-map validation (Section 8) uses the publicly available NASA C-MAPSS Turbofan Engine Degradation Simulation dataset (subset FD002) [10]. The spool-dynamics validation (Section 9) uses the publicly available N-CMAPSS dataset (file N-CMAPSS_DS02-006, unit 2) [11]; both are distributed through the NASA Prognostics Center of Excellence data repository.

Code availability. The prototype implementation is not publicly available. The inference procedure is fully specified by Eqs. (1)–(3).

Author contributions. The sole author conceived the method, implemented the prototype, performed the numerical experiments, and wrote the manuscript.

Use of AI tools. The manuscript was revised and spell-checked with the assistance of Anthropic's Claude. The author reviewed all such edits and is responsible for the entire content of the paper.