A Stable Gray-Box Surrogate for Unsteady Aerodynamic Coefficient Prediction in Real-Time Simulation

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

2026

Abstract

Real-time flight simulation must evaluate aerodynamic forces and moments at every integration step, where computational fluid dynamics is far too expensive and the usual interpolated coefficient table scales exponentially with its indexing variables and stays quasi-static. 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 coefficient map; and an explicit linear path carries near-linear rate 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 inputs, and is a trainable nonlinear extension of the rational-function approximations used in unsteady-aerodynamic modeling. We verify it on a synthetic unsteady airfoil — reproducing lag, hysteresis, and rate effects — and validate the static map against measured S809 wind-tunnel data; unsteady validation on oscillating-airfoil or flight data remains future work.

Keywords: unsteady aerodynamics; aerodynamic coefficient prediction; real-time simulation; rational-function approximation; Wiener model; dynamic stall; machine learning.

1 Introduction

A real-time vehicle simulation must compute the aerodynamic force and moment 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 flow directly with computational fluid dynamics (CFD) at each step is many orders of magnitude too slow for this budget. The aerodynamics must therefore be reduced to a cheap surrogate evaluated from a handful of state variables.

The standard surrogate is an interpolated lookup table of the dimensionless coefficients C_D, C_L, C_m, C_Y, C_l, C_n as functions of angle of attack α, sideslip β, Mach number, body rates, and control-surface deflections, scaled to dimensional loads by dynamic pressure and reference geometry. Tables are fast, but they scale badly. Their memory grows exponentially with the number of indexing parameters: a coefficient that depends on α, β, Mach, and several independently actuated control surfaces 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 a coefficient for the instantaneous state — whereas real aerodynamic loads depend on the recent *history* of the state: lift and moment respond to a lagged effective angle of attack, body rates induce additional loads, and near stall the response forms hysteresis loops as α rises and falls. Capturing

such unsteady behavior in a table means adding history or rate dimensions, compounding the memory blow-up; a plain neural network of the instantaneous state 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 aerodynamic coefficients 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 inputs grows, (iii) able to represent unsteady, history-dependent coefficients, (iv) numerically stable inside a closed-loop simulator, and (v) fittable directly from logged wind-tunnel, flight-test, or high-fidelity simulation data. The proposed model is a *Wiener* block structure: a bank of learnable second-order IIR filters (biquads) supplies the unsteady dynamics, and a small multilayer perceptron supplies the static nonlinear map to every coefficient. The filters carry the state history in a few internal registers, so memory is represented *natively* rather than tabulated; the model is a few hundred parameters in the configuration shown below rather than a sampled grid over the operating envelope, and its cost grows linearly — not exponentially — with the number of inputs. The filter poles are parameterized so that the linear dynamics are stable by construction.

The contribution is a deliberately aeronautical synthesis. Differentiable biquad filters are used as the stable linear block of a Wiener model for unsteady aerodynamic coefficient prediction, and the resulting structure is interpreted as a trainable nonlinear extension of the rational-function approximations widely used in aeroelasticity. The paper also states the runtime checks needed to make the stability claim meaningful in a deployed simulation and verifies the approach on a controlled unsteady-airfoil example. The example is not a substitute for wind-tunnel validation; it isolates whether the proposed structure can reproduce lag, hysteresis, rate effects, and dimensional load scaling in a setting where the true dynamics are known.

2 Aeronautical and modeling background

Differentiable IIR filters (audio). Closely related differentiable-biquad techniques were recently developed in neural audio-effect modeling. Nercessian [1] introduced differentiable biquads to match a target equalizer curve by gradient descent. Nercessian, Sarroff and Werner [2] 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; their best model emulated a distortion pedal with 40 biquads and roughly 210 interpretable parameters. Colonel et al. [3] design biquad cascades directly by deep learning, and the line continues to time-varying all-pole filters [4] and Koopman-linearised audio networks. Our linear block and its conjugate-pole stability parameterization follow this work; the present paper transfers it from audio to aerodynamics.

Unsteady aerodynamics as rational transfer functions. In aeroelasticity, unsteady aerodynamic loads are classically represented by *rational* transfer functions of the Laplace variable. Theodorsen's theory [5] and rational approximations of the Wagner indicial response lead to Roger's rational-function approximation (RFA) [6] and Karpel's minimum-state method [7], which fit pole–residue forms to frequency-domain aerodynamic data for flutter and gust analysis. A bank of biquads provides a finite set of learned second-order rational dynamical basis functions; the present model is therefore closely related to a *trainable* pole–residue/RFA representation whose coefficients are fit by gradient descent and whose output passes through a learned static nonlinearity. This extends the classically linear RFA idea toward the nonlinear, separated-flow regime where the

semi-empirical Leishman–Beddoes dynamic-stall model [8] is the usual reference.

Neural and reduced-order aerodynamic models. Data-driven unsteady-aerodynamic and aeroelastic reduced-order models — Volterra series, recurrent neural networks, neural state-space, and proper-orthogonal-decomposition hybrids — are surveyed by Kou and Zhang [10], within the broader machine-learning-for-fluid-mechanics landscape of Brunton et al. [9]. Block-oriented (Wiener/Hammerstein) models are also standard for nonlinear unsteady aero. Against a black-box RNN, the model here trades representational generality for three concrete properties: denominator stability by construction, a small interpretable parameter set whose poles correspond to physical time scales, 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. (α, flap, q), or any set of state and control channels — and the target be the coefficient vector $C[n] \in \mathbb{R}^K$, e.g. (C_D, C_L, C_m). 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 (such as a body rate) 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.

Static nonlinearity and linear path. The feature vector $v[n] = (g_0[n], \ldots, g_{F-1}[n], \hat{u}_0[n], \ldots, \hat{u}_{N-1}[n]) \in \mathbb{R}^D$, $D = F + N$ (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 coefficient on the inputs:

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

each de-standardized as $C_k = o_k s_k + m_k$. The linear path carries the near-linear effects — most importantly the rate (damping) derivatives — so the MLP is left to model the residual nonlinear, history-dependent part rather than competing with a large linear term; this is what lets a small, history-driven coefficient be fit accurately even when a large linear term dominates its variance (Section 7). The filter bank is *shared* across coefficients: one internal unsteady state drives every coefficient, with a per-coefficient static readout. Including the standardized inputs alongside the filtered features lets a purely static input (e.g. a discrete configuration flag) affect the map without passing through the dynamics.

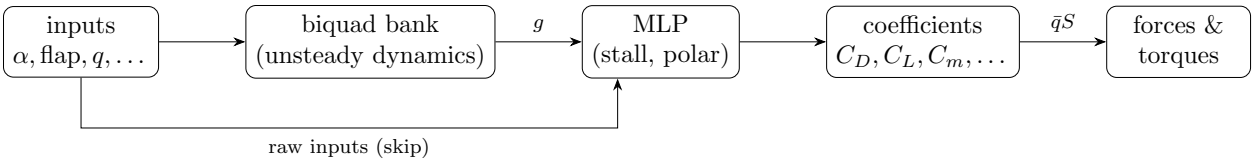

Figure 1: Proposed coefficient-prediction pipeline: learnable biquads (unsteady dynamics) feed a small MLP (static stall / drag-polar map); the raw inputs also skip to the MLP. The predicted coefficients are scaled to dimensional loads by dynamic pressure and reference geometry.

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 RNN reduced-order model, and it is essential for a model running in closed loop inside a flight simulator.

3.3 From coefficients to drag and torque

The K outputs are dimensionless. Dimensional loads follow from dynamic pressure $\bar{q} = \frac{1}{2}\rho V^2$ and reference geometry (area S, chord c, span b):

$$\text{force} = \bar{q}\, S\, C, \qquad \text{torque} = \bar{q}\, S\, L\, C, \tag{4}$$

with $L = c$ for pitch and roll-rate coefficients and $L = b$ for roll/yaw. Thus drag $= \bar{q}S\, C_D$, lift $= \bar{q}S\, C_L$, and pitch moment $= \bar{q}Sc\, C_m$. In a simulation implementation, $\bar{q}$ may either be read from a channel or computed from an airspeed channel and air density. The dimensional outputs are therefore derived scalar force and moment channels, while the learned network itself predicts only dimensionless coefficients.

4 Training

Given a time-ordered dataset of inputs and measured coefficients, parameters are fit by minimizing the per-coefficient-standardized mean squared error

$$\mathcal{L} = \frac{1}{TK} \sum_{n=1}^{T} \sum_{k=1}^{K} (o_k[n] - y_k[n])^2, \tag{5}$$

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),

$$\lambda[n] = \frac{\partial \mathcal{L}}{\partial g[n]} - a_1 \lambda[n{+}1] - a_2 \lambda[n{+}2], \tag{6}$$

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}, 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 coefficient model is wrapped by a deterministic runtime layer. At each simulator step the runtime reads the declared input channels, advances the filter states once using Eq. (1), evaluates the MLP in Eq. (2), publishes the predicted dimensionless coefficients, and optionally computes derived scalar force and moment channels using Eq. (4). 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, coefficients, 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; if dimensional loads are requested, the configuration must provide a valid dynamic-pressure source and positive reference geometry. 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 runtime applies the model's stored per-channel input standardization internally, so the input channels are provided in physical units.

The cost and memory of the deployed model are small and, crucially, *independent* of how finely the operating envelope is sampled. The verification model of Section 7 stores about 390 parameters (roughly 3 kB at double precision) and a runtime state of four samples per biquad (here 48 numbers); 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 coefficient over d arguments at g points per axis stores g^d entries: even a coarse $g = 20$ grid over five arguments (e.g. α, β, Mach, and two control surfaces) is 3.2×10^6 entries — about 26 MB per coefficient — and representing unsteadiness by adding rate or history axes multiplies this further. This contrast, together with the ablation of Section 7, is the practical case for a compact learned dynamical model when a coefficient 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 orga-

nized 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 aerodynamic model of this paper) and a slower outer process coexist in a single schedule. Numerical integration is likewise pluggable: several integrators of different order are available and selected per model. 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, so a control law can be developed and verified in simulation and then deployed without rewriting it. Within this environment the aerodynamic 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.

7 Verification case: synthetic unsteady airfoil

The present study uses a synthetic generator with known unsteady dynamics. This case should be read as verification of the model structure, gradient calculation, and real-time load scaling, not as aerodynamic validation against measured data. It is useful because the desired lag, hysteresis, and rate effects are known in advance, and can be checked independently of measurement noise or facility effects.

The generator maps (α, flap, q) to (C_D, C_L, C_m). Lift and moment respond to a first-order-lagged *effective* angle of attack, so that oscillating α produces hysteresis loops — the memory the biquad bank exists to capture; lift saturates through a tanh (stall); drag follows a quadratic polar $C_D = C_{D0} + kC_L^2$; and the pitching moment includes a pitch-rate damping term. The model dimensions and held-out fit are in Table 1. The small standardized error does not by itself prove the model superior to other surrogate classes; the verification criteria for this first case are that the learned model remains stable, carries temporal memory through its filter state, reproduces the sign and qualitative shape of the aerodynamic responses, and preserves the exact dimensional scaling from coefficients to forces and moments. Driven in a real-time simulation loop with an angle-of-attack sweep and an airspeed step $40 \to 60$ m/s, the model reproduces the expected behavior: C_L rising with α, a nose-down (restoring) C_m, drag following the polar, the pitch-rate term shifting C_m, and all dimensional loads scaling by the expected factor $(60/40)^2 \approx 2.25$ with dynamic pressure — with the pitch torque equal to $\bar{q}ScC_m$ to machine precision.

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 inputs is learned. Held-out error rises by more than an order of magnitude — from 2.0×10^{-3} to 1.3×10^{-1} (Table 1) — even though the static model has only about 16% fewer parameters. The memory supplied by the biquad bank, not the MLP nonlinearity, is therefore what captures the lag, hysteresis, and rate effects; this is exactly the regime in which a quasi-static lookup table or a memoryless network is structurally unable to fit the data.

Figure 2 makes this concrete. Driving the trained model through a clean sinusoidal angle-of-attack sweep, with the flap and pitch-rate channels held at zero so the only source of a loop is the

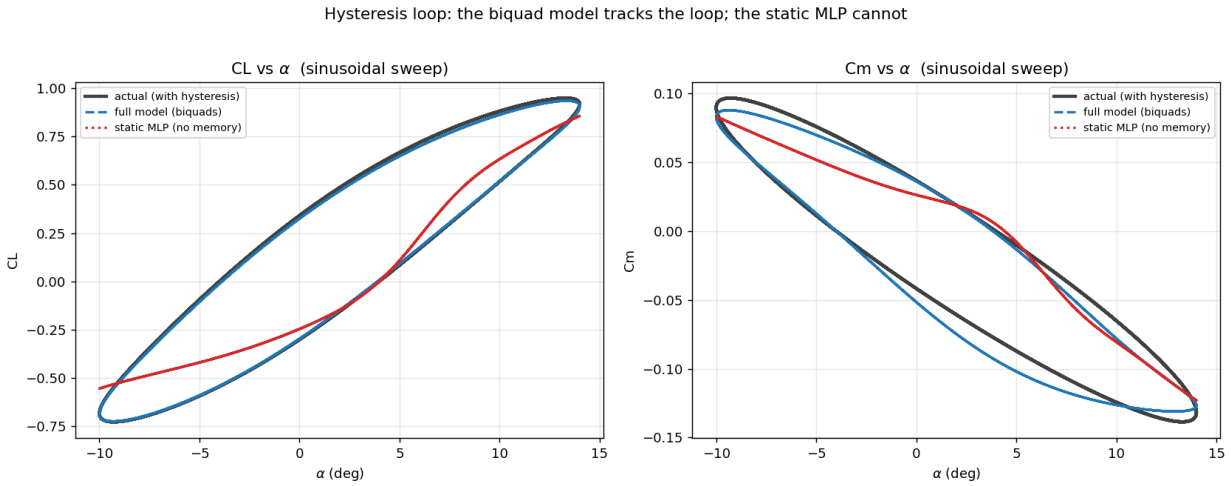

Figure 2: Synthetic unsteady-airfoil verification: predicted versus true aerodynamic-coefficient hysteresis on a sinusoidal α sweep (flap and pitch rate held at zero, so the loop is carried entirely by the lagged effective angle of attack). The full biquad-Wiener model (dashed) reproduces both the C_L and C_m loops; the memoryless static-MLP ablation (dotted) cannot represent a loop and collapses to a single-valued curve. The explicit linear path and input standardization are what let the C_m loop be recovered, despite C_m's variance being dominated by the (here zero) pitch-rate term (see text).

lagged effective angle of attack, traces out the C_L–α and C_m–α hysteresis loops. The full model reproduces both loops closely, while the static-MLP ablation, lacking internal memory, collapses each to a single-valued curve.

The pitching moment is the demanding case, and it is where the linear path and input standardization earn their place. In this generator the pitch-rate term dominates C_m: across the training set the lagged-angle-of-attack contribution accounts for only about 4% of the variation in C_m (the rate term for essentially all the rest), whereas for C_L the lagged-α term is the dominant signal. A single shared MLP fit by a standardized loss would therefore spend almost all of its C_m budget on the rate dependence and barely constrain the small lagged-α component — precisely the part that remains when the evaluation sweep sets the rate to zero, collapsing the C_m loop. The explicit linear path absorbs the near-linear rate term so the MLP can fit the residual history-dependent part, and input standardization keeps the small lagged-α feature from being swamped in the gradient. With both in place the C_m loop is recovered (Fig. 2), confirming that the design choices, not merely the presence of the filter bank, are what make a small history-driven coefficient fittable alongside a large rate-driven one.

7.1 Validation on measured wind-tunnel data

As a first check against *measured* aerodynamics rather than a synthetic generator, the model is fit to the static section data of the S809 wind-turbine airfoil measured in the Ohio State University 3×5 subsonic wind tunnel at a Reynolds number of 0.75×10^6, as reported by Ramsay, Hoffmann and Gregorek [11]. A single input (α) drives the three coefficients (C_D, C_L, C_m). Only the genuinely measured incidence range ($|\alpha| \leq 20.5°$) is used; the deep-stall continuation in the distributed file is a flat-plate (Viterna) extrapolation and is excluded. Every fourth point in angle of attack is held out as an interpolation test set. A static polar carries no time ordering, so here the model is used

Table 1: Synthetic unsteady-airfoil experiment ($N = 3$ inputs, $K = 3$ coefficients, 4000 samples). MSE is standardized and averaged over the three coefficients 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 inputs; the proposed model also trains the biquad bank. The analytic gradients of the proposed model match central finite differences to 7.5×10^{-11}.

Configuration	Biquads	Trainable params	Held-out MSE
Static MLP (no memory)	0 (pass-through)	316	1.3×10^{-1}
Proposed (biquad dynamics)	12	376	2.0×10^{-3}

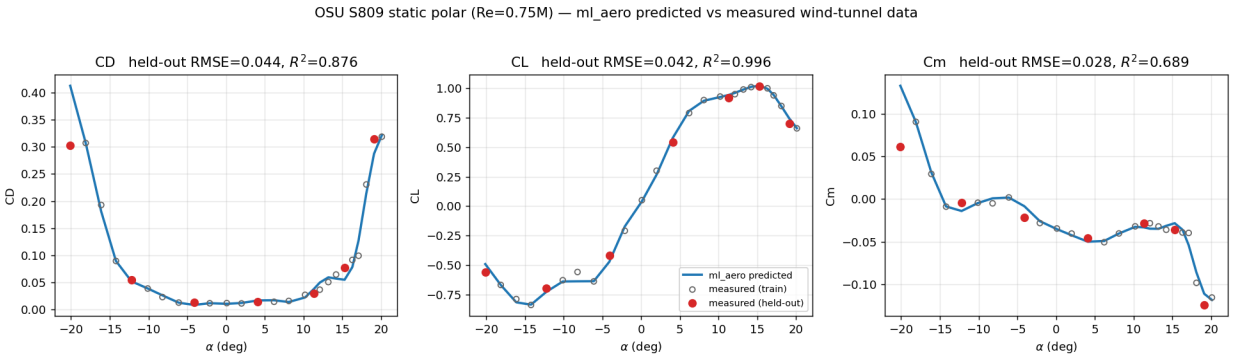

Figure 3: Validation on measured data: predicted versus measured C_D, C_L and C_m for the S809 airfoil in the OSU 3×5 tunnel at $\mathrm{Re} = 0.75 \times 10^6$ [11]. Open circles are training points, filled circles the held-out interpolation test, and the curve is the model. The held-out lift fit is $R^2 = 0.996$.

in its static-readout configuration — the biquads reduce to unit pass-through and the learned map is the MLP acting on the instantaneous incidence.

Figure 3 shows the predicted coefficients against the measurements. The model reproduces the lift curve through the stall break near 16°, the drag bucket, and the pitching-moment break. On the held-out points the lift fit is excellent ($R^2 = 0.996$); the drag and moment coefficients, whose measured variance over this range is small, are captured well (C_D $R^2 = 0.88$, C_m $R^2 = 0.69$). This validates the nonlinear *static* readout against real data; it does not exercise the unsteady filter bank, for which time-resolved oscillating-airfoil measurements (e.g. the pitch-oscillation cases in the same OSU campaign [11], or the dynamic-stall data of McCroskey [12]) are the appropriate next dataset.

8 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 lag — Reynolds- or Mach-dependent dynamic-stall time constants, or configuration-dependent spool-up — is 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 [2]. The model also inherits the usual limitation of lookup tables and neural surrogates: extrapolation outside the training envelope should be guarded. A production simulator should therefore flag or reject out-of-envelope inputs, or blend to a conservative fallback model. Finally, the unsteady results here are on synthetic data with a static-MLP ablation

as the only baseline, and the measured-data check of Section 7.1 exercises only the static readout. A full evaluation should add quantitative comparison against rational-function approximation, recurrent-neural-network reduced-order models, and high-resolution lookup tables, on time-resolved oscillating-airfoil or flight data, and should test whether the learned poles recover physically meaningful aerodynamic time scales (e.g. indicial or dynamic-stall time constants). That validation is the subject of ongoing work and is the prerequisite for using the model in a certification-relevant setting.

9 Conclusion

We presented a compact differentiable-biquad Wiener model for gray-box prediction of unsteady aerodynamic coefficients in real-time simulation. The model borrows stable differentiable-IIR machinery from signal processing and interprets it through an aeronautical lens as a nonlinear, trainable extension of classical rational-function unsteady-aerodynamic models. The resulting surrogate has fixed per-step cost, a small internal state, explicit pole-stability constraints, and a direct path from dimensionless coefficients to dimensional loads. A synthetic unsteady-airfoil case verifies the training and runtime procedure, including hysteresis, rate effects, and dynamic-pressure scaling, and the nonlinear static readout is validated against measured S809 wind-tunnel section data. The next step is to extend the measured-data validation to the unsteady regime using time-resolved oscillating-airfoil or flight-test data, and to add a direct comparison with lookup-table, recurrent-neural-network, and rational-function baselines.

References

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

[2] 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.

[3] 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.

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

[5] T. Theodorsen, "General theory of aerodynamic instability and the mechanism of flutter," NACA Report 496, 1935.

[6] K. L. Roger, "Airplane math modeling methods for active control design," AGARD-CP-228, 1977.

[7] M. Karpel, "Design for active flutter suppression and gust alleviation using state-space aeroelastic modeling," *J. Aircraft*, vol. 19, no. 3, 1982.

[8] J. G. Leishman and T. S. Beddoes, "A semi-empirical model for dynamic stall," *J. American Helicopter Society*, vol. 34, no. 3, 1989.

[9] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 52, 2020.

[10] J. Kou and W. Zhang, "Data-driven modeling for unsteady aerodynamics and aeroelasticity," *Progress in Aerospace Sciences*, vol. 125, 2021.

[11] R. R. Ramsay, M. J. Hoffmann, and G. M. Gregorek, "Effects of grit roughness and pitch oscillations on the S809 airfoil," National Renewable Energy Laboratory, Golden, CO, Tech. Rep. NREL/TP-442-7817, 1995.

[12] W. J. McCroskey, K. W. McAlister, L. W. Carr, and S. L. Pucci, "An experimental study of dynamic stall on advanced airfoil sections," NASA Technical Memorandum 84245, 1982.

Declarations

Funding. No external funding was received for this work.

Competing interests. The author declares no competing interests.

Data availability. The unsteady verification case (Section 7) uses synthetic data generated by the equations described there. The measured-data validation (Section 7.1) uses the publicly available S809 airfoil section measurements from the Ohio State University wind-tunnel campaign reported in NREL Technical Report NREL/TP-442-7817 [11].

Code availability. The prototype implementation is not publicly available. The inference procedure is fully specified by Eqs. (1)–(4).

Author contributions. The sole author conceived the method, implemented the prototype, performed the numerical experiment, 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.