Machine Learning with Applications 23 (2026) 100835

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Helicopter turboshaft modeling via mixtures of experts

Aurelio Raffa Ugolini [a,*], Francesco Aldo Tucci [b], Damiano Paniccia [b], Luigi Capone [b], Mara Tanelli [a]

[a] Dept. of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Via Giuseppe Ponzio 34, Milano, 20133, MI, Italy
[b] Leonardo Labs, Leonardo S.p.A., Via Raffaele Pieragostini 80, Genova, 16151, GE, Italy

ARTICLE INFO

Keywords:
Mixture of experts
Explainable machine learning
Software sensing
Helicopter turboshaft engine

ABSTRACT

A reliable and robust engine model is critical for helicopter design, operation, and maintenance, given the centrality of this sub-system. Several sources of uncertainty can limit the reliability and fidelity of first-principles models, necessitating data-driven solutions. Due to the relevant safety and security issues inherent to aircraft operation, however, fully black-box models may be unsuited to the challenge, due to their lack of explainability. In this work, we propose a multi-model approach to combine multiple physics-based descriptions, achieving a learning architecture that incorporates, in a data-driven setting, the existing knowledge of the engine's dynamics, maximizing interpretability and facilitating model validation and diagnostics. Enabled by recent advances in onboard data collection, we learn the model directly on realistic operating conditions by leveraging recorded flight information. The benefits include a high degree of local interpretability as well as minimal requirements in terms of input signals, as empirically demonstrated in a real-world use case. We compare our technique on a real helicopter dataset against the SINDy technique, showcasing the advantages of our approach against the well-known interpretable approach to Nonlinear System Identification.

1. Introduction

The importance of a robust engine model cannot be overstated, particularly in the context of helicopter design, operation (e.g., control), monitoring, and maintenance. In fact, dynamic interface problems between engine/fuel and aircraft/flight control systems have been known for a long time (see, e.g., Mihaloew et al., 1988), with important effects on handling and performance. These include, among others, torque and fuel supply oscillations, rotor speed excursions during maneuvers, and excessive vibrations, which can introduce additional overhead to pilots unless accounted for, especially during maneuvers that demand precise control over engine torque and rotor speed. Understanding and quantifying the coupling between the dynamics of the engine, fuel control, airframe, and flight controls has been regarded as essential for successful exploitation of control systems (see Corliss, 1982). Furthermore, effective engine modeling is instrumental to improve the overall fuel efficiency of helicopters. For example, thermodynamical models help to assess the role of operating conditions on engine performance, which is a significant factor in both operational cost and environmental impact.

High-fidelity, physics-based turboshaft models. Mathematical descriptions of turbo engines have been around for decades, finding applications in all stages of the machinery's lifetime. In Ballin (1988), a first-principles nonlinear mathematical model for a T700 turboshaft engine is introduced, based on the thermodynamic laws involved in the various stages along the gas path. For this reason, the model involves nonlinear relationships among the gas mass flows, temperatures, and pressures at the various stations of the engine (see Fig. 1), some of which are expressed as static maps computed offline. Similar models have been used in the literature for control purposes, such as in Castiglione et al. (2023), Gu et al. (2022), Zheng et al. (2018). Reliable engine simulations are also vital for the ongoing monitoring of engine deterioration. Comparing simulated and actual outputs allows for the detection of signs of wear and tear on engine components. Approaches based on thermodynamics, used for control and monitoring, can also be found in Ekinci and Yavrucuk (2020), Mao et al. (2019), Sheng et al. (2016), Yazar (2018), Zheng et al. (2018).

Thermodynamical models, however, require adjustments to account for different external conditions, e.g., air intake temperature and pressure, or the type of engine operation. In particular, the engine response

* Corresponding author.
E-mail addresses: aurelio.raffa@polimi.it (A. Raffa Ugolini), francesco.tucci.ext@leonardo.com (F.A. Tucci), damiano.paniccia@leonardo.com (D. Paniccia), luigi.capone@leonardo.com (L. Capone), mara.tanelli@polimi.it (M. Tanelli).

https://doi.org/10.1016/j.mlwa.2025.100835
Received 30 July 2025; Received in revised form 17 December 2025; Accepted 30 December 2025
Available online 4 January 2026

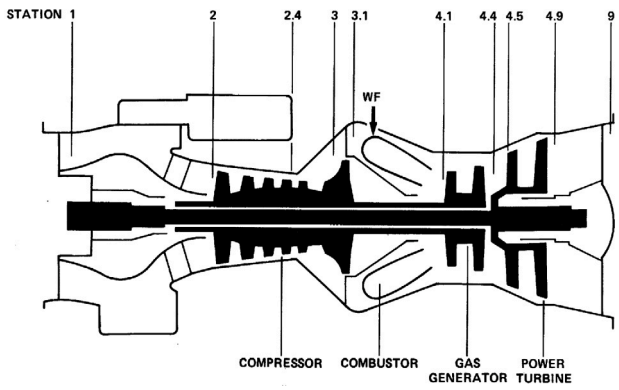


Fig. 1. Schematics of a small turboshaft engine.
Source: Adapted from Ballin (1988).

varies with the ambient conditions (see Volponi, 1998), which makes the characterization of the engine dynamics over the whole flight envelope a non-trivial problem. The same argument applies to performance maps obtained by rig tests, as differences may arise as a result of off-nominal operating conditions. In Volponi (1998), several strategies are presented to correct the engine variables, based on dimensional analysis, to their equivalent in *standard day conditions*. Instead, in Sheng et al. (2016), a method for obtaining component characteristics valid also during the engine start-up process is discussed, based on above-idle experimental data. Even then, multiple sources of uncertainty and disturbances still affect the operation of gas turbine engines. Indeed, the propulsion demand can vary greatly, which makes it extremely difficult for a model to be valid over the full operating envelope. In addition, unmeasured engine-specific differences are often present, such as various stages of degradation. These differences affect the engine's operation, which can be observed in practice as a drift from the ideal descriptions of the aforementioned models, with possible adverse effects on the engine's performance.

Data-driven methods. In recent years, data-driven (see, e.g., Montero-Jiménez & Vingerhoeds, 2018) as well as hybrid approaches (i.e., jointly incorporating data-driven and knowledge- or physics-based strategies, as discussed in Montero Jimenez et al., 2020) have appeared in the context of engine simulation and diagnostics. In hybrid approaches, the main motivation is to weave (partial) knowledge of the phenomenon (e.g., as physical constraints or similarity to recorded instances) into the insights directly inferred from the collected data.

Due to the inherent risks present in Aeronautical applications, however, Interpretability is a paramount concern for data-driven solutions. Indeed, helicopters are expensive machines that are often used in dangerous environments (e.g., search and rescue operations), leaving no room for doubts concerning the reliability of the instrumentation, be it hardware or software. Furthermore, as is common in aviation, helicopters are placed under strict regulatory assessments, which require careful evaluation of all components.

The preservation of physical interpretability represents a crucial point, e.g., in engine diagnostics, where a deviation from the nominal "ideal" engine behavior is expected. This fact has been addressed through a combination of methodologies among which: *(i)* the correction of the relevant variables to match the observed behavior (e.g., as done in Volponi, 1998), *(ii)* the adaptation of the model to account for degradation (e.g., using thresholds as in Litt et al., 1995, or self-tuning approaches as in Brotherton et al., 2003), *(iii)* the use of statistical inference methodologies (see Consumi & Agostino, 1998). Many data-driven approaches use data either to adapt the simplified descriptions to the observed condition (see, e.g., Castiglione et al., 2023; Gu et al., 2022; Wei et al., 2021) or to maintain physical interpretability, both in the context of engine control and monitoring. For example, in Duyar et al. (1994), a model-based fault detection and diagnosis scheme is developed for a liquid propulsion rocket engine. In Litt et al. (1995), a similar concept is developed for a T700 turboshaft engine, with the goal of designing an intelligent control system capable of fault accommodation.

As a particular case of the adaptive paradigm, several Parameter-Varying approaches have been proposed to model Turbo Engines (see, e.g., Gu et al., 2022; Yu et al., 2021). Linear Parameter Varying (LPV) and Quasi Linear Parameter Varying (Quasi-LPV) formalisms involve a single base model that is adapted continuously through functions of observed inputs, called *schedulers* (including, in the case of Quasi-LPV, functions of the system's states). Often, when the model is to be estimated from data, said functions are not explicitly defined but rather empirically obtained on a grid of scheduler values.

This is usually a complex problem that can be mitigated if fixed-scheduler experiments are available. However, in realistic scenarios where data is collected during operation, all flight parameters change simultaneously. An alternative is then to parameterize the scheduling functions explicitly, e.g., as interpolators between a finite set of local submodels. Approaches involving distinct models, possibly falling outside the family of LPV models, have indeed been applied to simulation problems, as shown in Fishwick (1993), Fishwick et al. (1994), Vangheluwe et al. (2002). The issue we face here is the lack of prior knowledge on the specific submodels that are required to effectively simulate the engine's outputs. This modeling uncertainty often collides with the need for reliability in data-driven solutions, characteristic of this sector.

What makes a model interpretable?

As mentioned, Interpretability is fundamental in safety-critical applications. Unfortunately, there is currently no consensus on a definition of Interpretability for ML Models. To circumvent deeper philosophical questions, we rely on the following notion, adapted from Doshi-Velez and Kim (2017).

Informal Definition (*Model Interpretability*)**.** We characterize the interpretability of (some aspects of) an ML Model as the "ability to explain or to present [said aspects] in understandable terms to a human" (Doshi-Velez & Kim, 2017).

Interpreting model predictions here is thought of as a tool to validate and diagnose a model. We need to be able to thoroughly understand the logic through which predictions are generated, and, perhaps more importantly, pinpoint why the model fails to yield realistic predictions if that happens. A potential ground for interpretability lies in relating the fitted model to known interpretable structures or patterns, such as first-principle physical models. Consequently, *constraining* a model to only achieve an interpretable structure, when applicable, is also an option.

From the "Informal Definition", we can see how interpretability does not depend exclusively on the Model, but also on the method through which we can provide *explanations*. We might want to explain a model "globally", i.e., by providing insight into how features affect predictions throughout the input domain, or "locally", e.g., conveying the effects of feature values on a specific prediction. In the following, we will compare both approaches to explaining data-driven models to be employed in the simulation of a turboshaft engine.

Prior to delving deeper into the details of the employed techniques, we stress the fact that, since "Informal Definition" has a qualitative nature, we are unable to *quantify* how interpretable a model is. Worse still, interpretability is often at odds with model complexity, which means that the most flexible and powerful models we can achieve are typically the most obscure. The result is that, in a standard ML Model Selection process where different learners are compared, the most interpretable candidates can be disproportionately penalized (Molnar et al., 2020).

2. Methodology

As mentioned, an abundance of mathematical work has been made available over the years concerning the modeling of turboshaft engines. The key question is then how to conjugate previous physical knowledge and data-driven insights to mitigate the aforementioned risks, while retaining interpretability.

To address this need, we propose a strategy that leverages (*i*) the well-known simplified description of the engine's dynamics (e.g., from Dreier, 2017), treated as grey-box models, as well as (*ii*) a multi-model approach which can incorporate flexibly a variety of operating-condition-specific behaviors. In order to link operating conditions to the most appropriate local submodel, we employ a version of the Mixture of Experts (MoE) architecture proposed by Jacobs et al. (1991), whereby the grey-boxes are treated as experts. This is in line with works such as Montero-Jiménez and Vingerhoeds (2018), where the importance of operating conditions is acknowledged, as well as multi-model approaches to simulation and diagnostics (see, e.g., Castiglione et al., 2023; Duyar et al., 1995; Montero Jimenez et al., 2020).

Such an adaptive framework, capable of blending physical constraints with data while respecting interpretability goals, is primarily directed towards simulation, e.g., for control and planning purposes. As described by Mihaloew et al. (1988), synergizing engine and flight controls has advantages in key areas of interest to helicopter manufacturers, such as reducing cost and time in development phases, increasing maneuverability and flight quality, as well as improving fuel efficiency and safety. However, additional uses could be envisioned for diagnostics and prognostics purposes, as the goal is to provide high-quality models for the nominal behaviors of a fleet or a single machine. Delimitations of the approach imposed by the experimental data collections are related to non-intrusiveness: minimal invasivity (i.e., only relying on flight parameters that are collected by the AMMCs during operation) as well as passive observation (i.e., without altering the control logic, which is acting in closed loop).

2.1. Simplified turboshaft models

Our goal is to learn a continuous-time dynamical system model for the engine output Torque (*TRQ*), given the effective control input variables to the system (namely Rotor Speed, N_R, and Collective Pitch, *COL*), and a set of non-controlled inputs measured during operation. As starting point, an approximation of the Torque response *TRQ* given the Fuel Flow (described in terms of a mass flow, *WF*) as input, obtained by assuming instantaneous changes in temperature along the gas path (i.e., disregarding the thermal inertia of the machinery), is found in Dreier (2017) as a time-constant model of the form:

$$(1+\tau s)TRQ = \mu WF, \tag{1}$$

with $\tau > 0$ so as to guarantee stability. The gain $\mu > 0$ is usually obtained by static performance results. The Fuel Flow is, however, not the true input of the system, as during operation it is automatically regulated from the Collective Pitch *COL* and Rotor Speed N_R to avoid drops in the latter. As per the previously cited work, the Fuel Flow control law is ideally described in the frequency domain via:

$$WF = \left(K_P + \frac{K_I}{s} + sK_D\right)\delta N_R + \frac{\tau_1 s + k_c}{\tau_2 s + 1} COL, \tag{2}$$

i.e., the sum of a PID controller in δN_R (defined as the offset of the rotor speed from the nominal value N_R^{ref}), and a lead–lag compensator (with $\tau_2 > 0$ guaranteeing stability) in the collective lever position *COL*. In practice, the parameters of Eqs. (1) and (2) are usually dynamically adjusted (i.e., *scheduled*) as a function of other measured flight parameters. This is a relevant point as we operate under the assumption of not having direct access to Fuel Flow measurements, but rather having to reconstruct it from the Collective position and Rotor Speed. In addition, both models represent only (local) approximations of the true dynamics, as differences in external conditions (e.g., air intake temperature and speed) and unmodeled components (e.g., thermal inertias) affect the fidelity of the models.

Because of the availability of the internal variables *WF*, N_G, and N_P limited to the learning phase, we can envision breaking down the learning problem into three separate stages: *first*, devise a surrogate for the signals unmeasured at runtime (N_G, N_P); *second*, learn a model for the *WF* based on the control inputs (N_R, *COL*) as well as the aforementioned estimates, modulated by the remaining observed quantities; *last*, repeat the approach for the *TRQ* where the reconstructed *WF* becomes the (explicit) control input and the remaining variables are used only to condition the response function. In both cases, reliance on (simplified) first-principle models is paramount to respect the physicality of the predictions, as well as enhance generalizability.

2.2. Multi-model approaches

The maps between flight parameters and the coefficients in Eqs. (1) and (2) are usually obtained by means of specific experiments (e.g., bench tests), and subsequently adjusted to compensate for different pressures and temperatures compared to those experienced when the tests were performed (see, e.g., Volponi, 1998). This generally prevents calibrating the maps using operational data and introduces systematic errors due to the applied correction. While data-driven approaches to estimate turbine compressor maps have been investigated (see, e.g., Yazar et al., 2024), our goal is to formulate a framework that is easily adaptable to new platforms and even individual machines under minimal data requirements.

A natural way to incorporate the changes in parameters necessary to apply Eqs. (1) and (2) in practice, is to rely on Parameter-Varying approaches (Gu et al., 2022; Yu et al., 2021).

The mixture of expert architecture. In order to circumvent an explicit mapping of the schedulers onto the governing equations' parameters, we rely on an automated strategy based on the Mixture of Experts (MoE) architecture introduced in Jacobs et al. (1991). The MoE, originally introduced in Jacobs et al. (1991), is an associative competitive framework in which several base learners (*experts*) are combined by a *gate* to solve supervised learning problems. In its inception (see Fig. 2), the MoE is a feed-forward network composed of different modules, each usually receiving the same inputs. The final output is the result of a *selector*, i.e., a stochastic "switch" gate, controlled by a set of allocation probabilities. This pooling operation is commonly replaced by a weighted sum of the experts' predictions, wherein weights are provided by the gating. This design makes the MoE fully deterministic, and can be interpreted as eliminating the stochasticity of the expert selection by predicting the output *in expectation*.

A Mixture of Experts is thus a special ensemble model, where the predictions from each learner are pooled together in a context-aware fashion by an additional learner, called "gating". A key feature of the MoEs is the ability to perform a soft partition of the input space via the gating: each point is assigned a different mixture of weights relating to the contribution of each expert, which can range from a single expert being "tasked" to perform the prediction to the importance being spread uniformly across the experts' pool (as in bagging). The outputs from the gating module thus have a dual interpretation: they encode the degree to which each expert can describe the output, as well as a degree of uncertainty on whether any one expert is suitable for a given example. Furthermore, because of the "divide et impera" approach, the MoE architecture is particularly indicated for problems where different "regimes" in the input–output dependency are assumed, but for which sharp boundaries are too limiting. Different regimes can correspond, for example, to different operating conditions. Thus, the various regions of expert specialization can be used to "inspect" the decision-making process of the overall algorithm, making it locally interpretable.

Many different conceptualizations of the MoE architecture have been introduced (see, e.g. Masoudnia & Ebrahimpour, 2014; Yuksel

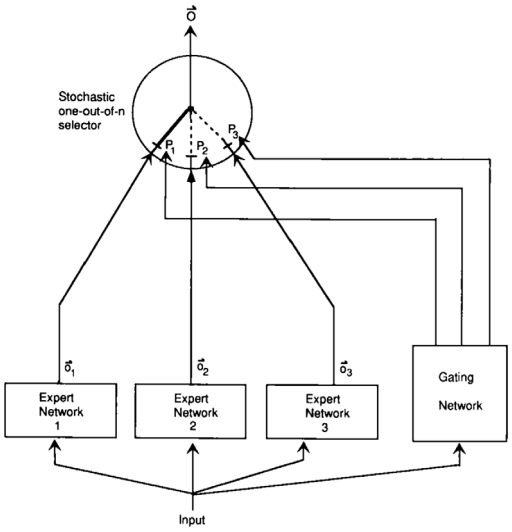


Fig. 2. Schematic Representation of the MoE.
Source: Adapted from Jacobs et al. (1991)

et al., 2012 for an organic review). From a methodological standpoint, developments have addressed hierarchies of experts (see Ahn & Sentis, 2021; Jordan & Jacobs, 1993), probabilistic extensions (see Bishop & Svensen, 2012; Etienam et al., 2024; Härkönen et al., 2022; Xia et al., 2020; Zhang et al., 2022), and differentiable formulations compatible with modern Machine Learning approaches (e.g., Hazimeh et al., 2021). MoEs have found successful application in machine learning problems characterized by natural task multimodality, such as Multi-Task Learning (MTL) (see, e.g. Hazimeh et al., 2021), and Natural Language Processing (NLP) (see, e.g., Fedus et al., 2022; Royer et al., 2023; Shazeer et al., 2017; Zhao et al., 2023). More closely related to the topic of this paper, MoEs and similar architectures have also been employed for the modeling and identification of dynamical systems (see Baldacchino et al., 2016, 2017; Bischof, 2022; Johansen & Babuska, 2003; Leoni et al., 2024), as well as in predictive maintenance (e.g., in Hassberger & Lee, 1989).

Our take on the MoE. The MoE framework enhances the expressiveness of the overall model beyond what individual experts can achieve. Even when using simple linear submodels, the architecture can capture nonlinear relationships by selecting different expert combinations across various regions of the input space. We exploit this idea to represent a Nonlinear System as an input-modulated combination of linear or weakly nonlinear systems. Besides the possibility of mirroring the presence of multiple "operating modes", this strategy allows for including physical knowledge on (parts of) the system, encoded, e.g., via the experts.

To learn a continuous-time dynamical system, we minimize a loss on the time derivative (e.g., $\boldsymbol{y}_t = \dot{\boldsymbol{z}}_t$ or $\boldsymbol{y}_t = \dot{\boldsymbol{z}}_t$) of the system's state $\boldsymbol{z}_t \in \mathbb{R}^d$, given the control variables $\boldsymbol{u}_t \in \mathbb{R}^u$, and possibly a set of additional "scheduler" variables $\boldsymbol{v}_t \in \mathbb{R}^v$. The result is a model in the form:

$$\boldsymbol{y}_t = f_{\text{MoE}}(\boldsymbol{x}_t; \boldsymbol{\theta}) = \sum_{i=1}^{M} \alpha_{ti} f_i(\boldsymbol{x}_t; \boldsymbol{\theta}_i), \tag{3}$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the concatenation of $\boldsymbol{z}_t$, $\boldsymbol{u}_t$, and $\boldsymbol{v}_t$. Observe that the coefficients α_{ti} themselves depend on the covariates and a subset of tunable parameters $\boldsymbol{\theta}_{\mathcal{G}}$, i.e., $\boldsymbol{\alpha}_t = f_{\mathcal{G}}(\boldsymbol{x}_t; \boldsymbol{\theta}_{\mathcal{G}})$.

Learning both the experts and the gating function simultaneously from data presents itself as a challenging optimization problem. The algorithm must balance two competing objectives: local accuracy (encoded by the experts) and global prediction quality (modeled by the overall MoE output). This reflects a trade-off between "competition" (experts specializing in different regions of the operating space) and "collaboration" (blending predictions to match the target, especially near regime boundaries). To address this, we propose a composite loss function that combines a collaborative term (squared error of the MoE prediction) and a competitive term (sum of the individual expert errors, weighted by their contributions):

$$\ell(\boldsymbol{x}_t, \boldsymbol{y}_t; \boldsymbol{\theta}, \boldsymbol{\beta}) = \sum_{t=1}^{N} \beta_1 \ell_{\text{coll}}(\boldsymbol{x}_t, \boldsymbol{y}_t; \boldsymbol{\theta}, \boldsymbol{\beta}) + \beta_2 \ell_{\text{comp}}(\boldsymbol{x}_t, \boldsymbol{y}_t; \boldsymbol{\theta}, \boldsymbol{\beta}), \tag{4a}$$

$$\ell_{\text{coll}}(\boldsymbol{x}_t, \boldsymbol{y}_t; \boldsymbol{\theta}, \boldsymbol{\beta}) = \left\| \boldsymbol{y}_t - \sum_{i=1}^{M} \alpha_{ti} f_i(\boldsymbol{x}_t; \boldsymbol{\theta}_i) \right\|_2^2, \tag{4b}$$

$$\ell_{\text{comp}}(\boldsymbol{x}_t, \boldsymbol{y}_t; \boldsymbol{\theta}, \boldsymbol{\beta}) = \sum_{i=1}^{M} \alpha_{ti}^2 \left\| \boldsymbol{y}_t - f_i(\boldsymbol{x}_t; \boldsymbol{\theta}_i) \right\|_2^2, \tag{4c}$$

where $\boldsymbol{\theta}_i$ are the parameters of expert i's prediction function f_i, and $\boldsymbol{A} = [\alpha_{ti}]$ is the $N \times M$ matrix of non-negative weights (rows summing to 1). The balance between collaboration and competition is controlled by hyperparameters $\boldsymbol{\beta} = (\beta_1, \beta_2)$, which can be tuned via cross-validation or specified a priori. To ensure the aforementioned condition on $\boldsymbol{A}$, we parameterize $f_{\mathcal{G}}(\bullet; \boldsymbol{\theta}_{\mathcal{G}})$ as a Multi-Layer Perceptron (MLP) with SoftMax activation in the last layer.

Choice of experts. A key advantage of the MoE architecture lies in its ability to embed physical domain knowledge. We do so by enforcing structural constraints on the experts, such that each one locally behaves like a simple (e.g., linear) continuous-time model informed by the set of schedulers $\boldsymbol{v}_t$.

Specifically, each expert is parameterized as:

$$f_i\left(\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{v}_t; \boldsymbol{\theta}_i\right) = \boldsymbol{g}_i\left(\boldsymbol{v}_t; \boldsymbol{\theta}_i\right)^{\top} \begin{bmatrix} \boldsymbol{z}_t \\ \boldsymbol{u}_t \end{bmatrix}, \tag{5}$$

where $\boldsymbol{g}_i = \boldsymbol{g}(\bullet; \boldsymbol{\theta}_i) : \mathbb{R}^v \mapsto \mathbb{R}^{d+u}$ is an MLP parameterized vector-valued function, whose role is to provide the linear parameters of f_i (with respect to $\boldsymbol{z}_t$, $\boldsymbol{u}_t$), given schedulers $\boldsymbol{v}_t \in \mathbb{R}^v$. Through Eqs. (5), each individual expert is itself an LPV model. Eq. (5) is essential to our explanation strategy because it allows a local interpretation of the model (i.e., sufficiently close to each query point $(\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{v}_t)$) as a linear system with coefficients $\boldsymbol{g} = \sum_i \alpha_i \boldsymbol{g}_i$. Notice that, in general, the schedulers $\boldsymbol{v}_t$ might also include control variables. We do not consider the Quasi-LPV case, where schedulers are allowed to contain state information.

Eq. (5) does not include any physical *bias* on the system, aside from the linear structure if $\boldsymbol{v}_t$ is constant. Yet, we might be interested in enforcing the *signs* for (some of) the mapped parameters, e.g., to respect physicality or some stability conditions on the local representation. This is achieved via an activation function of the form:

$$h(x) = \pm \ln\left(1 + e^{kx}\right)/k, \tag{6}$$

where $k > 0$ is a hyperparameter. Function (6) enforces positivity (respectively, negativity) for the mapped output, approaching zero for large negative x, and asymptotically approaching x ($-x$) for $x \to \infty$.

We see the fact that we can enforce a desired structure for the model's dynamical equations via parameterization (5), coupled with sign constraints (6), as an inherent interpretability feature. Indeed, given a value for the schedulers $\boldsymbol{v}_t$, we can always relate the (local) representation (5) to a linearized system of the same structure as a "reference". Since linear systems have been extensively analyzed, many properties can be easily attached to the system. A few considerations on the properties of the so-parameterized MoE architecture are discussed in Appendix.

Ultimately, we will extensively exploit the ability to impose a desired structure throughout the experimental section, where it will help us relate the MoE to well-known simplified equations of the turboshaft system.

Fitting the MoE architecture. Since the loss described in (4) features coupling between the parameters of each expert and the gating, we use Automatic Differentiation (AD) and a gradient-based optimizer (the *Adam* algorithm from Kingma & Ba, 2017). Once a model is trained, an open-loop simulation can be performed using any ODE solver.

2.3. A brief overview of SINDy

In order to benchmark our technique with a popular (global) interpretability approach to Nonlinear System Modeling, we compare it against the SINDy method (see Brunton et al., 2016). SINDy attempts to identify a nonlinear, continuous-time system of the form

$$\dot{\boldsymbol{x}}_t = f(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{7}$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the system's state ($\boldsymbol{x}_0$ is assumedly known), $\boldsymbol{u}_t \in \mathbb{R}^u$ is an external, controllable input that can is fed to the system and $f : \mathbb{R}^n \times \mathbb{R}^u \to \mathbb{R}^n$ is the (unknown) function dictating the system dynamics. SINDy aims to identify a model for (7) from state and input measurements, exploiting a *library* $\phi(\boldsymbol{x}_t, \boldsymbol{u}_t) \in \mathbb{R}^{n_\phi \times 1}$ of *predetermined* basis functions to yield an approximation of the map f as

$$\dot{\boldsymbol{x}}_t \approx \boldsymbol{\Theta}^\top \phi(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{8}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{n_\phi \times n}$ is the *sparse* matrix of coefficients associated with the basis functions, usually either linear or (smooth) nonlinear in the state, inputs or both of them.

Given a dataset $D = \{\boldsymbol{x}_t, \dot{\boldsymbol{x}}_t, \boldsymbol{u}_t\}_{t=1}^T$, where $\boldsymbol{x}_t$, $\dot{\boldsymbol{x}}_t$ and $\boldsymbol{u}_t$ denote the available samples of the state, its derivative and input trajectory, upon introducing matrices

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{\boldsymbol{x}}_1, & \dot{\boldsymbol{x}}_2, & \dots, & \dot{\boldsymbol{x}}_T \end{bmatrix}^\top, \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1, & \boldsymbol{x}_2, & \dots, & \boldsymbol{x}_T \end{bmatrix}^\top,$$
$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1, & \boldsymbol{u}_2, & \dots, & \boldsymbol{u}_T \end{bmatrix}^\top, \quad \boldsymbol{\Theta} = \begin{bmatrix} \theta_1, & \theta_2, & \cdots, & \theta n \end{bmatrix},$$

the SINDy algorithm learns a sparse representation of (7) in the candidate terms (8) by solving a sparse optimization problem of the form

$$\min_{\boldsymbol{\Theta}} \left\| \dot{\boldsymbol{X}} - \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{U})\boldsymbol{\Theta} \right\|_2^2 + \lambda^2 \sum_{i=1}^n \left\| \boldsymbol{\Theta}_i \right\|_0, \tag{9}$$

where $\boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{U})$ stacks (row-wise) the values of $\phi(\boldsymbol{x}_t, \boldsymbol{u}_t)$ computed for each input/state pair. Note that this problem is separable with respect to the columns of $\boldsymbol{\Theta}$ and, thus, its solution can be solved by tackling the resulting n (sparse) regression problems in parallel.

Notably, problem (9) features an ℓ_0-norm regularization, which makes the overall problem non-convex. A possible approach to handle this class of problems approximately is the *Sequential Thresholded Least Squares* (STLS) algorithm (Brunton et al., 2016), whose iterative step j can be summarized as:

$$S_h^j \leftarrow \left\{ \boldsymbol{\theta} \in \mathbb{R}^{n_\phi} : \theta_i = 0 \; \forall i : \left| \theta_{hi}^{(j)} \right| < \lambda \right\}, \tag{10a}$$

$$\boldsymbol{\theta}_h^{(j+1)} \leftarrow \arg\min_{\theta_h \in S_h^j} \frac{1}{2} \| \dot{X}_h - \boldsymbol{\Phi}(X, U)\Theta_h \|_2^2, \tag{10b}$$

for each column of $\boldsymbol{\Theta}$ and, thus, $h = 1, \dots, n$. When $\boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{U})$ is full rank, the scheme in (10) is known to converge to a fixed point in a finite number of steps that is upper-bounded by the dimension n_ϕ of the library, that is also a local minimizer of (9) (see Zhang & Schaeffer, 2018). Notwithstanding the apparent simplicity of SINDy, it has been applied to control problems (see, e.g., Fasel et al., 2022; Kaiser et al., 2018).

By looking at the fitting problem in (9), it is clear that SINDy requires the user to select the library of basis functions. The choice of an appropriate basis function heavily depends on the users' domain knowledge. Further, the interpretability of the SINDy technique thus rests on the delicate assumption that a good approximation can be obtained in terms of few terms within the candidate libraries. Equivalent sets of features may thus lead to different levels of sparsity in the solution. As such, when deep domain knowledge is insufficient, lengthy trial-and-error procedures are required to select a "good" set of basis functions, for which sparsity is not guaranteed in general. Although SINDy is often viewed as an "interpretable-by-default" tool, this entails that the possibility of making sense of the final model is at risk if a sparse solution cannot be recovered.

2.4. A purely data-driven alternative

As a baseline for all the other models discussed, we consider a purely data-driven black-box approach. More specifically, given the available input variables (N_R, COL, v_{AIR}, P_0, and T_1), we consider a nonlinear predictor for the target variable y_t (either N_G, WF, or TRQ at time t) described as

$$y_t \approx f_{\text{NFIR}}\left(\boldsymbol{x}_t^{N_R}, \boldsymbol{x}_t^{COL}, \boldsymbol{x}_t^{v_{AIR}}, \boldsymbol{x}_t^{P_0}, \boldsymbol{x}_t^{T_1} \right) \tag{11}$$

where each feature vector $\boldsymbol{x}_t^{\bullet}$ is a collection of past values of the corresponding variable over a given horizon, and the (nonlinear) map f_{NFIR} has to be obtained from data. This class of models is called "Nonlinear Finite Impulse Response" (NFIR, see Qin & McAvoy, 1996). The reason for this choice is that the class of models (11) is always Bounded-Input, Bounded-Output (BIBO) stable, and does not involve expensive continuous-time simulations but rather pointwise predictions. In addition, we can adopt a non-parametric approach to the estimation of f_{NFIR}, e.g., via Kernel Methods involving Radial Basis Functions.

2.5. Model evaluation

As a measure of the quality of the models, we consider the average Mean Absolute Error (MAE) over individual simulated trajectories in the validation/test set. For a single trajectory $\boldsymbol{y}^{(j)}$ (indexed by an integer j), given the corresponding open-loop simulation $\hat{\boldsymbol{y}}^{(j)}$, the MAE is defined as

$$MAE\left(\boldsymbol{y}^{(j)}, \hat{\boldsymbol{y}}^{(j)} \right) = \sum_{t=1}^{N_j} \left| y_t^{(j)} - \hat{y}_t^{(j)} \right|. \tag{12}$$

Since different trajectories in the sets may have vastly different numbers of samples $N_j, j = 1 \dots J$, we define the score on the entire set as a simple average:

$$MAE_{set} = \frac{1}{J} \sum_{j=1}^{J} MAE\left(\boldsymbol{y}^{(j)}, \hat{\boldsymbol{y}}^{(j)} \right); \tag{13}$$

this is used to avoid biasing the MAE_{set} metric in favor of long trajectories, and, consequently, disregarding fit on short ones. Without loss of generality, we rescale (12) and (13) by the average absolute output value over training

$$\overline{y} = \frac{1}{N} \sum_{j=1}^{J} \sum_{t=1}^{T_j} \left| y_t^{(j)} \right|, \quad N = \sum_{j=1}^{J} T_j \tag{14}$$

yielding the "Relative" MAE on trajectories and on the entire set, as

$$RMAE\left(\boldsymbol{y}^{(j)}, \hat{\boldsymbol{y}}^{(j)} \right) = MAE\left(\boldsymbol{y}^{(j)}, \hat{\boldsymbol{y}}^{(j)} \right) \; / \; \overline{y}, \tag{15a}$$

$$RMAE_{set} = MAE_{set} \; / \; \overline{y}. \tag{15b}$$

Hyperparameter tuning. Our MoE formulation includes several hyperparameters crucial to performance: the number of experts, M, is of course the most notable, as it directly controls the trade-off between flexibility and complexity; the combination weights, β_1, β_2 in Eq. (4), are also relevant in determining the degree of expert specialization, as discussed, whereas the value of k from activation functions of the form (6) determine the degree of penalty towards (near) non-physical solutions. In addition, the complexity of the gating module (i.e., number of layers, neurons per layer) has to be tuned empirically. We use a Bayesian optimization algorithm (BayesOpt, see Martinez-Cantin, 2014) to automatically search for a good hyperparameter configuration based on the Validation $RMAE_{set}$ (15b). We also apply a similar procedure to tune the hyperparameters (e.g., the sparsity penalty $\lambda > 0$ for the STLS Algorithm, and polynomial degree p for the candidate library) to fit SINDy, and once again select the best model via Validation $RMAE_{set}$.

Table 1
Variables in the Turboshaft Dataset and their Runtime Availability.

Variable	Category	Symbol	Avail.
Collective Pitch	Control Input	COL	✓
Rotor Speed	Control Input	N_R	✓
Intake Air Velocity	Uncontrolled Input	v_{AIR}	✓
Engine Air Intake Pressure	Uncontrolled Input	P_0	✓
Engine Air Intake Temperature	Uncontrolled Input	T_1	✓
Gas Producer Turbine Speed	Internal Engine Variable	N_G	✗
Power Turbine Speed	Internal Engine Variable	N_P	✗
Fuel Flow	Intermediate Output	WF	✗
Engine Output Torque	Engine Output	TRQ	✗

3. Experimental results

Our goal is to recover the torque output of the helicopter's turboshaft engines from a minimal set of observables, in an attempt to enhance the applicability of the technique compared to more invasive approaches. Thus, we attempt to leverage as few sensors as possible; this is not only to restrict the complexity of the models and introduce fewer sources of uncertainty (e.g., modeling errors and potential sensor malfunction, as discussed in Volponi, 2014) but also to disencumber the methodology from signals that may not be monitored by all helicopter's onboard Automated Mission Management Computer (AMMC).

The dataset contains, for each turboshaft of a twin-engine helicopter, the variables reported in Table 1 over a total of 13 h of flight, subdivided into different missions where specific maneuvers were performed and analyzed. Notice that the data, collected during actual operation, have a closed-loop Fuel Flow actuation as described in Kim (1992). Among the recorded variables, only the Collective Pitch, COL, is directly controlled by the pilot, whereas we consider the Rotor Speed, N_R, as input to the engine, although it is not directly manipulated; indeed, part of the Engine's onboard automatic control is aimed at keeping the N_R constant. The external Air Speed (v_{AIR}), intake pressure (P_0), and temperature (T_1) are measured and available for prediction, as they affect engine performance, but represent external conditions that cannot be controlled. The Gas Producer Speed, N_G, and Power Turbine Speed, N_P, are essential engine variables that are recorded in this dataset, which are, however, unavailable during routine flights. All of the aforementioned variables are recorded at a 12 Hz frequency during the flights contained in the dataset on both engines via two sets of independent FADECs and AMMCs for redundancy. Thus, the first step in our preprocessing is a correlation analysis: we only keep the flight data where the signals are coherent between AMMCs, resulting in a dataset where each engine descriptor is corroborated by the sensors. The helicopter under study, moreover, features two twin turboshaft engines. To augment the data, we treat signals recorded on different engines as separate datapoints, with attention to keeping data from the same flights within the same set, to avoid leaking information. The models presented in this work are specific to the helicopter type they were trained for, whereas adaptation to other machines might require training new models. We leave more advanced alternatives, such as Transfer Learning, as future work.

Based on the literature concerning turboshaft engines, we know that key internal quantities to describe the engine state are the temperatures, pressures, and mass flows between each engine station along the gas path (see, e.g., Ballin, 1988; Green et al., 1997). Unfortunately, these internal engine variables are not recorded either during operation or as part of the flight dataset object of this analysis. Instead, the only variables in the set suitable to characterize the engine state are the air intake pressure (P_0) and temperature (T_1), the Fuel Mass Flow to the combustion chamber (WF), the Gas Producer Speed (N_G) and the Power Turbine Speed (N_P), upon which we know from previous works (see Gu et al., 2022; Zheng et al., 2018) that an approximate engine model can in principle be constructed.

Given the available data and observed variables, our goal is to realistically and interpretably simulate the Torque output (and the intermediate Fuel Flow output). More specifically, we aim at learning two cascaded continuous-time models, one for the Fuel Flow (WF), given the effective control input variables to the system (namely Rotor Speed, N_R, and Collective Pitch, COL), and a second one for the engine output Torque (TRQ), given the Fuel Flow. Both models should also rely on the set of non-controlled inputs available, and on a suitable *proxy* for the internal engine variables (see Table 1), to enhance performance, acting as schedulers.

Because of the availability of the internal variables N_G, and N_P limitedly to the learning phase (see Table 1), we start by reconstructing signals that are not available during operation, so that downstream models can respect the physical dependencies of the true system. The data comes from a free-turbine turboshaft engine, which entails that the Gas Turbine and Power Turbine are not directly linked via a mechanical transmission. Most flight conditions do not involve significant differences between Main and Tail Rotor Torque demands or abrupt Collective Pitch Changes, which keeps the Main Rotor and Power Turbine Speeds nearly proportional. As such, we can easily surrogate the internal state N_P with the measured input N_R. Since, instead, the N_G variable is non-trivial to surrogate, we use the NFIR approach described in Section 2.4 to build a causal estimate based on the available observable inputs. Following the approach described in Kurzke (2009), Volponi (1998), we correct signals to their Standard Day Conditions equivalent, and subsequently normalize them by standard scaling. Since the approaches employed in the analysis involve fitting models on the derivatives of the state variables, we perform an estimation of the gradients with denoising by adopting Total Variation Regularization (see Selesnick et al., 2020; Van Breugel et al., 2020).

Moe strategy. We train the two models for the TRQ and WF independently under loss (4), i.e., directly using gradients ($\dot{TRQ}$ and $\dot{WF}$) estimated from the available trajectory data.

Notice that, if one neglects the integral term in Eq. (2), both models can be rewritten in the time domain as first-order linear ODEs, namely:

$$\dot{TRQ} = -\frac{1}{\tau} TRQ + \frac{\mu}{\tau} WF, \tag{16a}$$

$$\dot{WF} = -\frac{1}{\tau_2} WF + \frac{K_P \delta \dot{N}_R + (K_D + \tau_2 K_P)\delta \ddot{N}_R + \tau_2 K_D \delta \dddot{N}_R + k_c \dot{COL} + \tau_2 \ddot{COL}}{\tau_2}. \tag{16b}$$

In other words, $y_t^{TRQ} = \dot{TRQ}$ should be, given any possible choice of schedulers $\boldsymbol{\nu}_t^{TRQ}$, a (varying) linear combination of the state $z_t^{TRQ} = WF$ and control input $u_t^{TRQ} = WF$. Likewise, $y_t^{WF} = \dot{WF}$ should be described by a linear combination of the state $z_t^{WF} = WF$, COL and its first derivative, δN_R and its first two derivatives[1] (i.e., the inputs $\boldsymbol{u}_t^{WF}$), for any choice of schedulers $\boldsymbol{\nu}_t^{WF}$. Additionally, the multiplicative coefficients of TRQ in (16a) and WF in (16b) should be negative, whereas that WF in (16a) and of $\dot{COL}$ in (16b) should be positive, to guarantee physicality and stability of the linearized systems.

Thus, we can set $z_t^{WF} = WF$,

$$\boldsymbol{u}_t^{WF} = \left[\delta N_R, \delta \dot{N}_R, \delta \ddot{N}_R, COL, \dot{COL}\right]^\top,$$

and include all additional schedulers in $\boldsymbol{\nu}_t^{WF}$, ensuring that $\boldsymbol{g}_i^{WF}$ fulfills $g_{i1}^{WF} < 0$ and $g_{i6}^{WF} > 0$, for any choice of schedulers $\boldsymbol{\nu}_t^{WF}$ by means of the activation (6). In the case of Eq. (1), instead, $\dot{TRQ}$ appears as a (varying) linear combination of TRQ and WF, thus we set $z_t^{TRQ} = TRQ$, $u_t^{TRQ} = WF$, with all additional schedulers in $\boldsymbol{\nu}_t^{TRQ}$, and ensure that $g_{i1}^{TRQ} < 0$ and $g_{i2}^{TRQ} > 0$ once more via (6). Hyperparameters are tuned via the *RMAE* (15b) on the Validation set. For the downstream TRQ model only, we compare MoEs featuring different inputs:

1. The true WF signal;

[1] Since a pure integrator is not stable, we cannot define the integral of δN_R along flight trajectories as input to the MoE; instead, we neglect the integral dependency (which amounts to either assuming $\delta N_R \approx 0$ or $K_I \approx 0$).

2. The NFIR-reconstructed *WF* surrogate, i.e., a black-box but BIBO-stable and finite-memory estimate for the actual system's input;
3. The simulated *WF* from the upstream MoE model, i.e., an interpretable surrogate.

The first option has the purpose of establishing an "ideal" baseline for the MoE *TRQ* model, exempt from reconstruction errors affecting the inputs. In practice, however, only the last two options are viable, since *WF* is unavailable at runtime.

SINDy strategy. Via SINDy, we tackle directly the models for the *WF* and *TRQ* via

$$y_t^{(j)} = \dot{z}_t^{(j)} = \boldsymbol{\phi}^{(j)}\left(z_t^{(j)}, \boldsymbol{u}_t^{(j)}, \boldsymbol{v}_t^{(j)}\right)^\top \boldsymbol{\theta}, \tag{17}$$

where the superscript $^{(j)}$ denotes the model (i.e., *WF* or *TRQ*), and $\boldsymbol{v}^{(j)}$ the set of schedulers. Notice that, here, we cannot differentiate the role of state, input, or scheduler information within the model. In either case, we select a polynomial candidate library $\boldsymbol{\phi}^{(j)}$. To tune the hyperparameters of the model (sparsity penalty $\lambda \in [10^{-2}, 10^{3}]$, polynomial degree $1 \le p \le 3$, STLSQ Algorithm's regularization parameter $\alpha \in [10^{-4}, 10^{-1}]$), we employ the *RMAE* (15b) computed on the Validation set. Of course, the *RMAE* does not penalize overly complex models, as other criteria such as the AIC or BIC would, but instead reflects a standard model selection pipeline focused on performance and requiring the least amount of human intervention.

Similarly to the MoE case, we compare different choice inputs for the *TRQ* model:

1. The true *WF* signal;
2. The NFIR-reconstructed *WF* surrogate.

Remark 1 (*Note on the Figures*)**.** Figs. 3–6 portray simulations over different trajectories in the datasets. A flight contains a variable number of continuous recorded trajectories $\boldsymbol{y}^{(j)}$, as described in Section 2.5. For each trajectory presented, a 3×1 grid plot reports:

Top – the simulation of the output (either *WF* or *TRQ*) via different models (MoE, SINDy or NFIR) against the ground truth. For clarity, each signal is rescaled by the average absolute output value over training, $\bar{y}$.

Middle – the inputs to the system (and, where applicable, their surrogates) and some schedulers (e.g., δN_R and *COL* for the Fuel Flow, and *WF*, δN_R, *COL* for the Torque). Derivatives of the inputs are not represented for clarity.

Bottom – the outputs of the gating computed over the simulated trajectory (for the MoE model only); these are represented as stacked color-coded vertical bands, where the height of each segment equals the gating output for any particular expert.

Although the models presented in the following are not optimized for real-time inference, and usage for control requires very low computational overhead, especially if deployed on edge devices with limited memory or computational power, both the SINDy and MoE models presented in this work support real-time simulations at the aforementioned 12 Hz polling frequency on a laptop.[2]

3.1. Intermediate output: Fuel flow

The performance indices for the different approaches can be found in Table 2.

The NFIR Model consists of a simple nonlinear regression for the outputs (*WF* in this case), in a set of time-shifted input coordinates. The time-discretization in the NFIR model mitigates issues related to the estimation of time derivatives for δN_R and *COL*, limiting the noise fed to the model. While the choice of predictors in (11) automatically make it BIBO-stable, it also prevents the model from considering past outputs as features in an autoregressive fashion and, therefore, modeling transients that might affect the *WF*. This is clear from the plots in Fig. 3, wherein the NFIR signal does exhibit offsets with respect to the true *WF* signal. The results of the MoE technique can be seen in Table 2 and Fig. 3, where we can observe a better reconstruction of the *WF* output compared to the NFIR model, albeit some *drifts* can be observed towards the end of trajectories.

A surprising result, instead, concerns the SINDy model for the *WF*. For starters, the best performing candidate under our model selection procedure (with polynomials up to the third degree) contains 398 terms:

$$\dot{WF} = -6.95\ \dot{COL} - 12.58\ \delta N_R + 10802.22\ \delta\dot{N}_R + \cdots + 55.46\ T_1^2\ \dot{N}_G. \tag{18}$$

This highlights a key limitation of the basic SINDy algorithm, i.e., that sparsity depends on the particular choice of coordinates used to describe the system's dynamics. Furthermore, the sheer number of terms in the equation renders explanations virtually impossible.

While the SINDy model offers the best performance on Validation and in the median Test case (see Table 2), it results in some unstable dynamics over the Test set (see, e.g., Fig. 3b). This is a critical limitation of SINDy, as we cannot easily ensure stability for arbitrary choices of candidate libraries.[3]

Table 2
Fuel Flow (*WF*) Reconstruction Performances (Relative MAE [%], Best in Bold) on the Validation and Test Flights.

RMAE [%]	Validation			Test		
Model	Mean ± S.D	Median	IQR	Mean ± S.D	Median	IQR
NFIR	3.10 ± 1.72	2.63	1.24	3.47 ± 1.86	2.92	2.40
MoE	2.08 ± 1.16	1.79	1.29	**2.18** ± 1.58	1.64	0.98
SINDy	**1.47** ± 1.07	**1.02**	**1.06**	(1 ± 8) $\times 10^8$	**1.22**	**0.81**

Table 3
Torque (*TRQ*) Reconstruction Performances (Relative MAE [%], Best in Bold) on the Validation and Test Flights. Rows in *italic* are reported for comparison purposes only, as they are based on the true *WF* signal (unavailable at runtime).

RMAE [%]	Validation			Test		
Model	Mean ± S.D.	Median	IQR	Mean ± S.D.	Median	IQR
NFIR	4.07 ± 2.17	3.47	1.46	4.41 ± 2.61	3.64	2.93
MoE [Ideal]	*1.73* ± 0.59	*1.69*	*0.80*	*1.79* ± 0.70	*1.60*	*0.99*
SINDy [Ideal]	*0.85* ± 0.35	*0.79*	*0.51*	*1.30* ± 1.69	*0.88*	*0.57*
MoE-NFIR	2.41 ± 0.67	2.37	**0.77**	2.69 ± 1.37	2.54	1.16
SINDy-NFIR	**1.96** ± 1.07	**1.53**	1.05	**2.34** ± 2.36	**1.55**	**0.85**
MoE-MoE	2.77 ± 1.13	2.36	1.59	2.91 ± 2.38	2.29	1.17

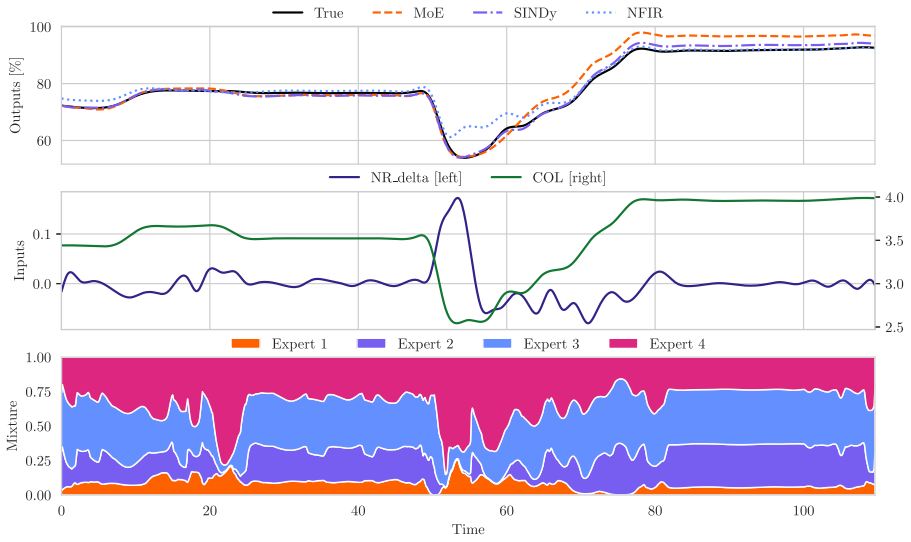
3.2. Final output: Torque

We set a baseline for modeling in the *TRQ* case too by considering the same NFIR model class as discussed in Section 2.4. Compared to the *WF* case, we can see from Table 3 as well as from Fig. 5 (light blue dotted line) that the NFIR model performs worse, likely due to the fact that it ignores not only the dependency from past outputs but also from the true *WF* signal. Nevertheless, it acts as a valid benchmark for the simulation approaches.

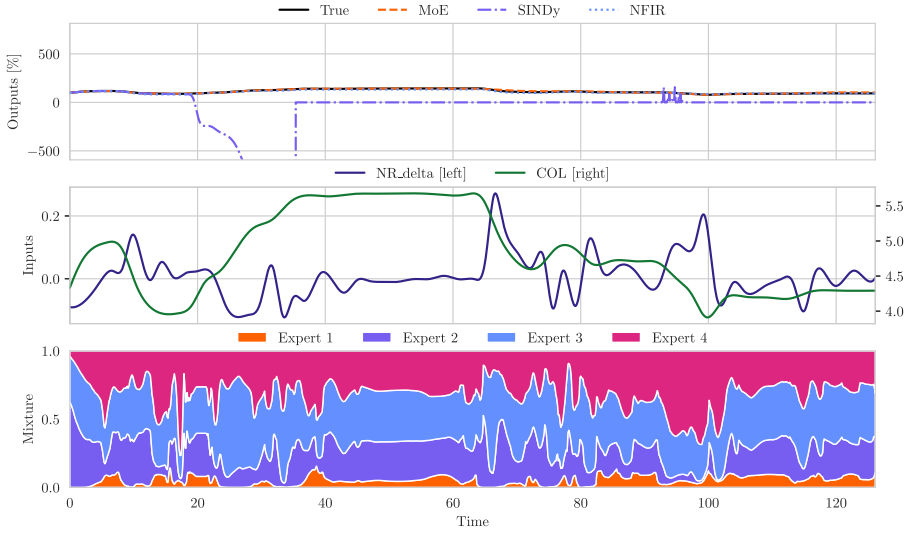
We further establish two "ideal" baselines to test the models: the *MoE [Ideal]* and *SINDy [Ideal]*, with the True Fuel Flow Input (*WF*) used for both Training and Simulation. These experiments are intended

[2] MacBook Pro M3, late 2022.

[3] In Kaptanoglu et al. (2021), a variant of the base SINDy algorithm with global stability guarantees ("Trapping" SINDy) is proposed, with the limitation of only considering quadratic nonlinearities.



(a) Validation Flight



(b) Test Flight

Fig. 3. Fuel-Flow via MoE, SINDy, and NFIR on Validation (a) and Test (b) Trajectories.

to show the extent to which the model types can reconstruct the *TRQ*, based on a theoretical uncorrupted reconstruction of the *WF*. Previous work follows a similar approach concerning a SINDy *TRQ* model (Paniccia et al., 2025), based on candidates chosen from a subset of the signals in Table 1. The aforementioned work demonstrates that SINDy can successfully recover equations akin to (16a), provided that manual model selection is employed to maximize model sparsity.

By the aforementioned Hyperparameter Tuning strategy, we hereby focus on model accuracy to select a SINDy model. This results in a *SINDy [Ideal]* model that is superior to the MoE counterpart, although lacking sparsity once again. The *SINDy [Ideal]* model features 821 terms:

$$\dot{TRQ} = -36.40TRQ + 31.00WF + 107.27\dot{WF} + 2.57C\dot{O}L + \cdots - 938.28{T_1}^3$$

Further, while the *MoE [Ideal]* shows consistent metrics across Validation and Testing, we can observe a significant deterioration of the *SINDy [Ideal]*'s Mean *RMAE* during Testing. Upon closer inspection, we can see that this phenomenon is due to some simulations yielding non-physical results for the *TRQ*, as visible in Fig. 4b.

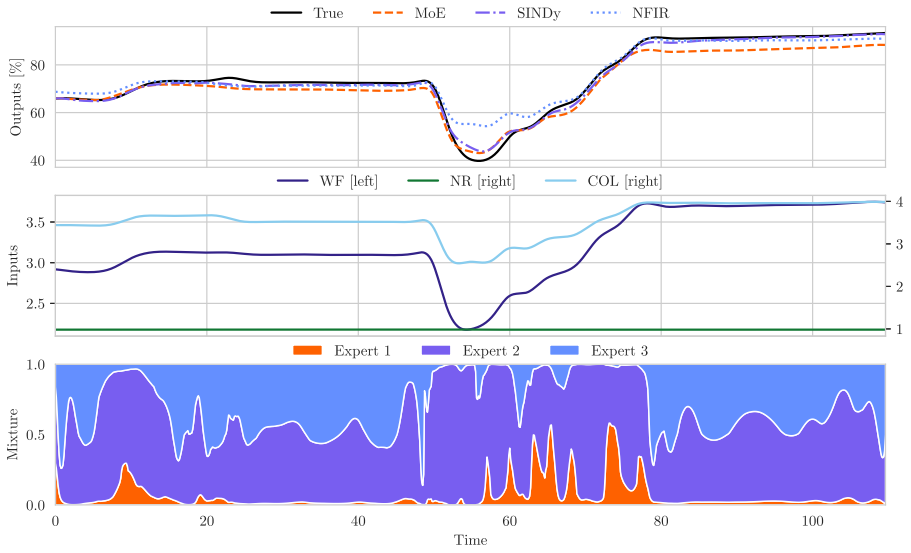
This fact is worrisome, as such anomalies were absent throughout the Validation trajectories, which could have misled us during model selection.

NFIR-based WF surrogate. The next two models we compare are instances of the MoE and SINDy, this time fitted on the NFIR-based surrogate to the *WF* signal. We can see, from Table 3, that the SINDy model (SINDy-NFIR) once again surpasses the MoE (MoE-NFIR) on almost all metrics (in particular, a paired t-test on the null hypothesis of the models having identical average RMAE on the Test flights leads to rejection with a p-value of 9.04×10^{-6}). The SINDy model in this case contains 611 terms:

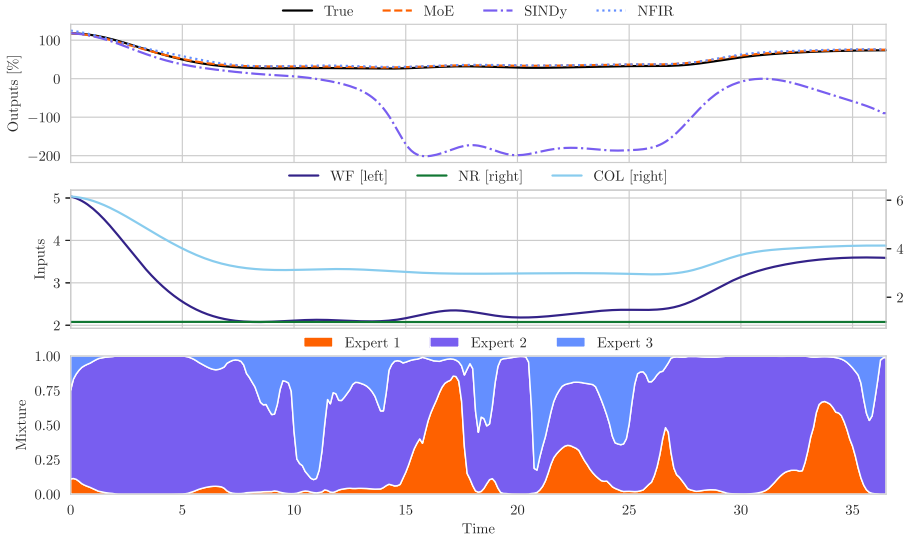
$$\dot{TRQ} = 90.86 - 0.01TRQ - 0.61WF - 34.19\dot{WF} + 3.38COL + \cdots - 1.82P_0^3$$

We can once again detect a discrepancy between the Mean and Median Test performance, which is explained by non-physical behaviors such as the one visible in Fig. 5b.

MoE-based WF surrogate. We also include in the comparison a downstream model fitted on the MoE-based *WF* simulations. The reason for this choice is that we want to assess the capabilities of an end-to-end



(a) Validation Flight



(b) Test Flight

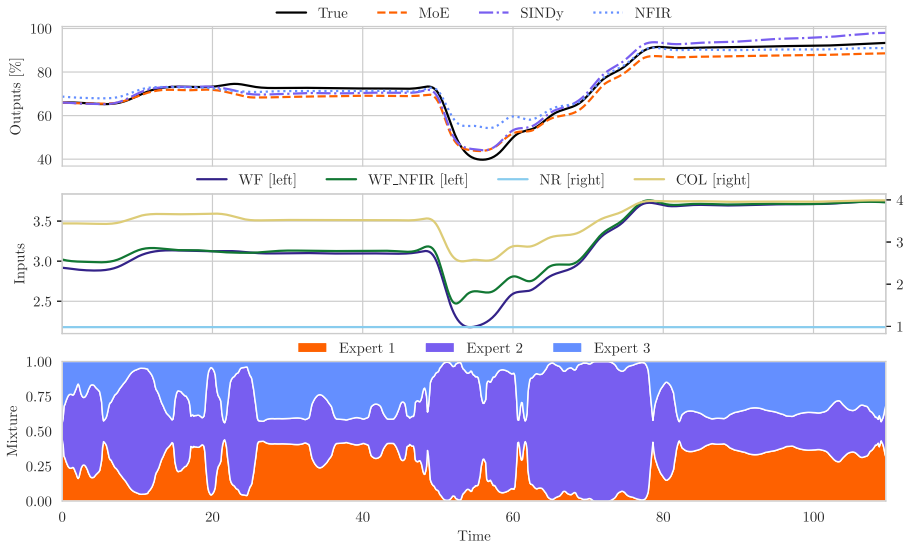
Fig. 4. Torque (True *WF* inputs) via MoE, SINDy, and NFIR on Validation (a) and Test (b) Trajectories.

MoE model for the simulation of the *TRQ* output. In doing so, we might be tempted to directly use the *MoE [Ideal]* model on the MoE-simulated *WF* signals, but we have observed that this leads to a significant deterioration of the performance. Instead, we train a new MoE model to account for modeling errors in the upstream *WF* reconstruction. The result (MoE-MoE, see Table 3) is a much less parsimonious model ($M = 6$ experts, compared to the 3 of the other approaches) which fails to compete with the MoE-NFIR alternative. Examples of simulations performed with this model are visible in Fig. 6.

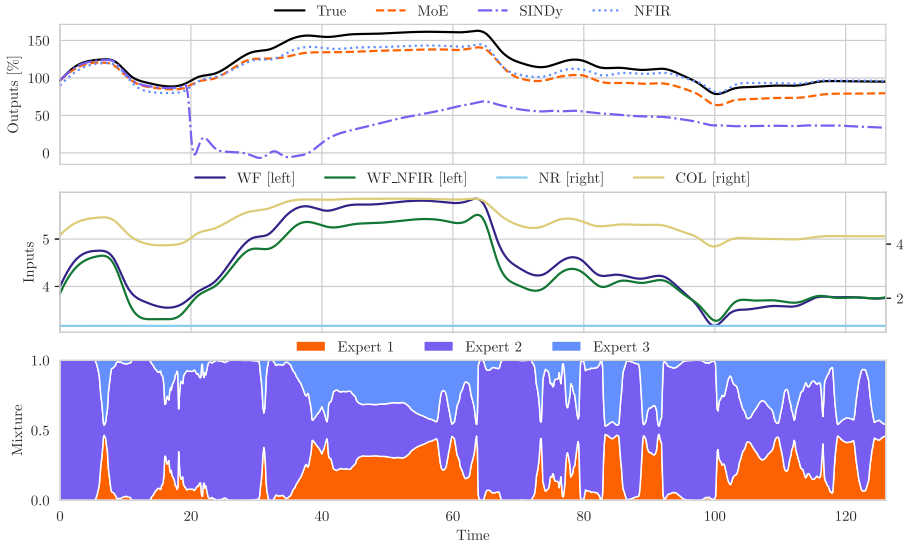
4. Discussion

By comparing the MoE framework against baseline approaches (the NFIR method and SINDy technique), we have shown that the former can effectively compete with black-box alternatives for the prediction of the Torque signal while remaining locally interpretable. In particular, we observed how the reconstruction is accurate when very informative signals are provided, as in the case of the MoE based on the true *WF* measurements. A key qualitative advantage of our technique is the ability to provide a simplified local explanation for the predictions, in the form of a combination of the expert submodels, which respects some underlying physical priors. As mentioned, this characteristic is, in our opinion, an interpretability feature, since it allows us to relate a specific local realization of the model to a reference structure on which we possess domain knowledge.

Meanwhile, SINDy is also portrayed as an appealing strategy for pursuing physics-based learning. The goal of SINDy is to return *minimal* nonlinear models, which can be then inspected and studied by means of many standard techniques (e.g., linearization, perturbation, or bifurcation analysis). The possibility of successfully employing these approaches, however, it is dependent on the kinds of nonlinearities and the scale of the system, which means it is not guaranteed for any SINDy model. Our analysis highlighted difficulties in achieving sparse, intelligible representations for the systems' dynamics and ensuring stability. Indeed, our main takeaway is that it is difficult to automatically select a SINDy model that is simultaneously interpretable (i.e., only considers a few candidates), accurate (in terms of simulation performance), and robust to extrapolation. By adopting a hyperparameter tuning strategy focused on simulation performance, we have been able to select impressively accurate (on most trajectories) yet highly



(a) Validation Flight



(b) Test Flight

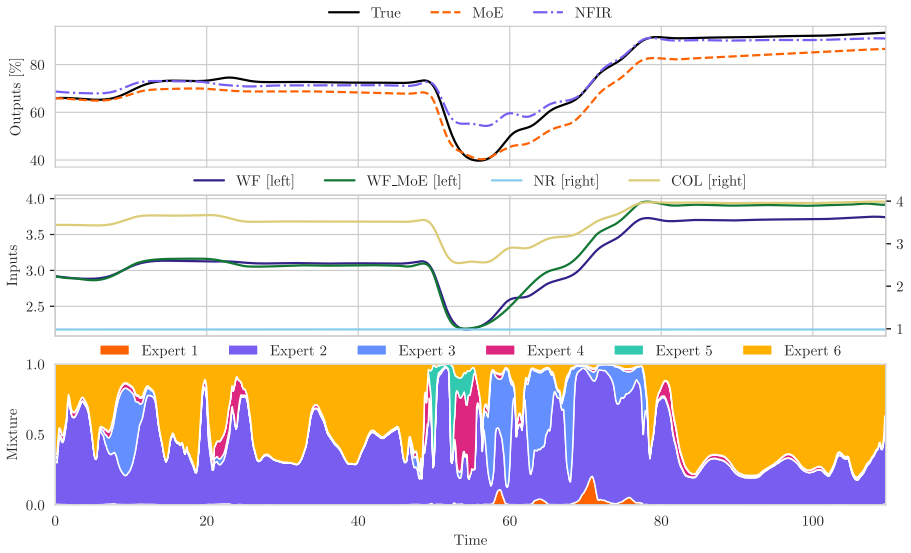
Fig. 5. Torque (NFIR-reconstructed *WF*) via MoE, SINDy, and NFIR on Validation (a) and Test (b) Trajectories.

complex models, i.e., sacrificing interpretability as well as robustness. In stark contrast with classical ML applications, where the fitting and inference tasks are usually aligned, the dynamical context introduces simulation, entailing challenges such as unforeseen instabilities and violation of the physical domain of variables. With complex polynomial models such as SINDy's, the chance of inadvertently including unstable modes or non-physical equilibria is very high. At the same time, the very tools that are effective in ascertaining such aspects become less and less effective as the model's complexity increases.

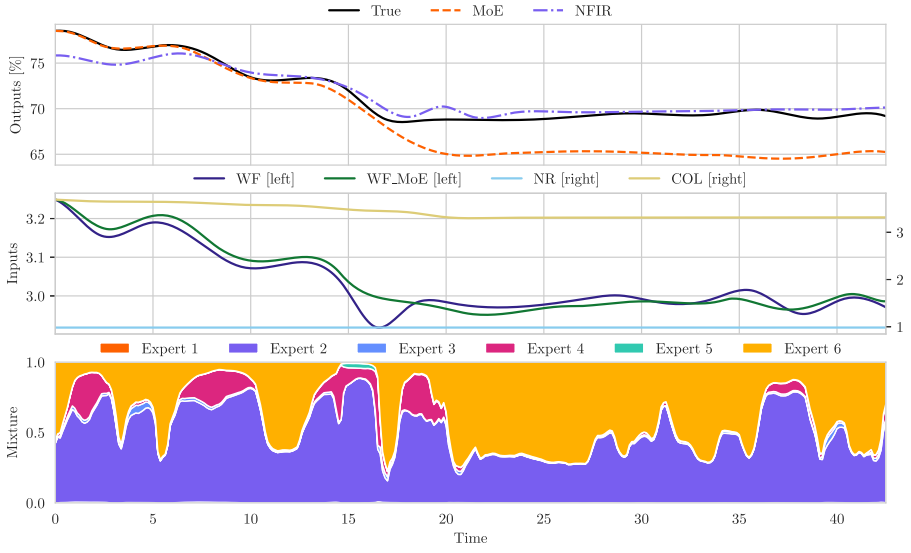
In our opinion, this fact alone should act as a cautionary tale to anyone considering SINDy for critical applications. Indeed, with complex, nonlinear systems such as in this case, ensuring that data collected during operation (i.e., at closed loop) conveys all relevant operating modes is practically infeasible. Even worse, the fact that the trained SINDy models perform extremely well in Validation but degraded quickly at test time indicates that there is a possibility of being misled during model selection. Thus, the models we employ should be extremely robust even in extrapolation regimes, e.g., by guaranteeing key properties such as stability, which is extremely difficult to ensure on models such as the ones returned by SINDy in the general case.

The high-level insight is that interpretable and robust models can be achieved via SINDy, but (at least in this application) only through extensive human intervention. The multi-model perspective supporting the MoE, meanwhile, appears to be more robust, since it directly tackles local approximations of the dynamics, in a way that remains interpretable regardless of underlying complexity (e.g., of the gate).

Moreover, while we did not delve into explaining the specific models that we obtained by fitting the MoE technique, the ability to "disassemble" their prediction logic into a gated superimposition of simpler submodels can be a very important tool. Indeed, it allows us to attribute varying degrees of *responsibility* among the pool of experts to specific predictions. Since experts are manageable models by design, they can also be further explained by other techniques, which is, however, out of the scope of this work. Nevertheless, the attribution of regions of the input space to experts can assist in data collection campaigns, e.g., to determine which models need to be improved, as well as validating the model on specific challenging scenarios. The MoE also guarantees a stronger regularity of the dynamical system, in light of the possibility of easily enforcing some notions of stability



(a) Validation Flight



(b) Test Flight

Fig. 6. Torque (MoE-reconstructed *WF*) via MoE on Validation (a) and Test (b) Trajectories.

(see Appendix), and interpretability by including a desired structure for the linearized system.

5. Conclusions

In this work, we have addressed the problem of modeling the Torque output of a turboshaft engine from operational data, highlighting the importance of inspecting the decision-making logic of the model. We then proceeded to discuss what simplified representations, as well as adjustment strategies, are eligible under a minimal set of available input signals. Based on our research, we found that multi-model paradigms such as MoEs are particularly promising for this task, as they can very naturally incorporate physical knowledge, in terms of structure and stability. More specifically, we find that an approach treating complexity in physical phenomena through the lenses of parsimonious simplified descriptions, e.g., in the form of linear submodels, is easier to justify and manage compared to finding a few meaningful nonlinear terms within a pool of candidates. This intuition is not necessarily in contrast with the ideas at the core of SINDy, as shown, e.g., in Callaham et al. (2021). However, our experiments and analyses against both a black-box baseline and SINDy show that the MoE architecture is a valid tool for the intended applications, which does not appear to suffer from the same pathologies as its competitor with respect to extrapolation. Indeed, although SINDy achieves high accuracy on average, the occasional errors manifest themselves as dangerous, non-physical scenarios (e.g., as shown in Fig. 4). Notwithstanding the specific application presented in this work, the introduced methodology is much more general, with potential use in many other engineering domains which make use of simplified models. Moreover, the "explainable" structure of the MoE proposed here enables a scalable model for large-scale problems, without sacrificing the possibility of analyzing the individual submodels, for example by assessing their stability. In our future work, we intend to consider a different learning paradigm, which attempts to exploit as much information as possible from the entirety of the time histories. Further, we propose to tune cascaded models simultaneously, i.e., on both the final outputs and intermediate signals.

During the preparation of this work the author(s) used Grammarly and Writefull in order to enhance readability. After using these tools/services, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

CRediT authorship contribution statement

Aurelio Raffa Ugolini: Conceptualization, Investigation, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing. **Francesco Aldo Tucci:** Conceptualization, Validation, Writing – review & editing. **Damiano Paniccia:** Conceptualization, Validation, Writing – review & editing. **Luigi Capone:** Resources, Supervision. **Mara Tanelli:** Supervision, Validation, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Frozen and robust stability of the MoE

The stability of the linearizations of an LPV system is sometimes referred to as "frozen stability" of the nonlinear system (see Briat (2015, Chapter 2, page 46)). Consider a scalar system of order d, i.e., $\boldsymbol{z}_t = \left[z_t, \dot{z}_t, \ddot{z}_t, \dots\right]^\top \in \mathbb{R}^d$ is the collection of a quantity $z_t \in \mathbb{R}$ and its first $d-1$ derivatives. In the following, we shall assume that the gate and $\boldsymbol{g}_i$ share the same inputs, i.e., $\alpha_i = \alpha_i(\boldsymbol{v}_t;\boldsymbol{\theta})$, and that the schedulers $\boldsymbol{v}_t$ do not contain any state information.

The "frozen" linear system corresponding to (3) is the one achieved by keeping the schedulers constant, i.e., setting $\boldsymbol{v}_t \equiv \overline{\boldsymbol{v}}$ for all t. The system thus becomes:

$$\dot{\boldsymbol{z}}_t = \left[\dot{z}_t, \ddot{z}_t, \dots\right]^\top = \boldsymbol{A}\left(\overline{\boldsymbol{v}}\right)\boldsymbol{z}_t + \boldsymbol{B}\left(\overline{\boldsymbol{v}}\right)\boldsymbol{u}_t, \tag{A.1}$$

where the matrices $\boldsymbol{A}\left(\overline{\boldsymbol{v}}\right) \in \mathbb{R}^{d\times d}$ and $\boldsymbol{B}\left(\overline{\boldsymbol{v}}\right) \in \mathbb{R}^{u\times d}$ are given by

$$\boldsymbol{A}\left(\overline{\boldsymbol{v}}\right) = \left[\begin{array}{c|cccc} 0 & 1 & 0 & \dots & 0 \\ \vdots & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ \hline \sum_i \alpha_i\left(\overline{\boldsymbol{v}}\right)\boldsymbol{g}_{i,1:d}\left(\overline{\boldsymbol{v}}\right)^\top \end{array}\right], \quad \boldsymbol{B}\left(\overline{\boldsymbol{v}}\right) = \left[\begin{array}{ccc} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ \hline \sum_i \alpha_i \boldsymbol{g}_{i,(d+1):(d+u)}\left(\overline{\boldsymbol{v}}\right)^\top \end{array}\right]. \tag{A.2}$$

Observe that a necessary and sufficient condition for invertibility of $\boldsymbol{A}\left(\overline{\boldsymbol{v}}\right)$ is that

$$\sum_i \alpha_i\left(\overline{\boldsymbol{v}}\right) g_{i,1}\left(\overline{\boldsymbol{v}}\right) \neq 0, \tag{A.3}$$

since, by applying the Laplace expansion, we obtain

$$\det\left(\boldsymbol{A}\left(\overline{\boldsymbol{v}}\right)\right) = \pm\left(\sum_i \alpha_i\left(\overline{\boldsymbol{v}}\right) g_{i,1}\left(\overline{\boldsymbol{v}}\right)\right)\det\left(\boldsymbol{I}_{(d-1)\times(d-1)}\right) = \pm\sum_i \alpha_i\left(\overline{\boldsymbol{v}}\right) g_{i,1}\left(\overline{\boldsymbol{v}}\right). \tag{A.4}$$

Under condition (A.3), when $\boldsymbol{u}_t \equiv \overline{\boldsymbol{u}}$ for all t, the "frozen" system (A.1) always admits the unique equilibrium

$$z^\star = -\frac{\sum_i \alpha_i \boldsymbol{g}_{i,(d+1):(d+u)}\left(\overline{\boldsymbol{v}}\right)^\top \overline{\boldsymbol{u}}}{\sum_i \alpha_i\left(\overline{\boldsymbol{v}}\right) g_{i,1}\left(\overline{\boldsymbol{v}}\right)}. \tag{A.5}$$

Notice that (A.3) is fulfilled whenever all $g_{i,1}$ are either strictly positive or negative for all schedulers $\overline{\boldsymbol{v}}$, which is simple to encode via activation (6). Moreover, the stability of the frozen system (A.1) can be determined directly by $\boldsymbol{A}\left(\overline{\boldsymbol{v}}\right)$.

Example 1 (*Frozen Stability for Scalar First-Order Systems*)**.** Consider a linear, first-order, 1D, continuous-time dynamical system:

$$\dot{z}_t = a z_t + \boldsymbol{b}^\top \boldsymbol{u}_t.$$

The condition for Asymptotic Stability (AS) of said system is $a < 0$. Therefore, to achieve a frozen-AS model via MoE in this case is trivial, by simply setting $h(x) = -\ln\left(1+e^{kx}\right)/k$ on the first output unit for $\boldsymbol{g}_i$ for all experts, because of convexity of (3).

Proof. In this case, $A\left(\overline{\boldsymbol{v}}\right) = \sum_i \alpha_i\left(\overline{\boldsymbol{v}}\right) g_{i,1}\left(\overline{\boldsymbol{v}}\right)$, which coincides with its only eigenvalue. To ensure stability, we can simply set $g_{i,1} < 0$ for all experts. □

Example 2 (*Frozen Stability for Scalar Second-Order Systems*)**.** Consider a linear, second-order, 1D, continuous-time dynamical system:

$$\ddot{z}_t = a z_t + c\dot{z}_t + \boldsymbol{b}^\top \boldsymbol{u}_t.$$

The condition for Asymptotic Stability (AS) of said system is $a, c < 0$. Therefore, to achieve a frozen-AS model via MoE is still simple, only requiring two negative-sign constraints via (6).

Proof. Here, by setting $a\left(\overline{\boldsymbol{v}}\right) = \sum_i \alpha_i g_{i,1}\left(\overline{\boldsymbol{v}}\right) \in \mathbb{R}$ and $c\left(\overline{\boldsymbol{v}}\right) = \sum_i \alpha_i g_{i,2}\left(\overline{\boldsymbol{v}}\right) \in \mathbb{R}$, we get

$$\boldsymbol{A}\left(\overline{\boldsymbol{v}}\right) = \begin{bmatrix} 0 & 1 \\ a\left(\overline{\boldsymbol{v}}\right) & c\left(\overline{\boldsymbol{v}}\right) \end{bmatrix},$$

whose characteristic equation is

$$\lambda^2 - \lambda c\left(\overline{\boldsymbol{v}}\right) - a\left(\overline{\boldsymbol{v}}\right) = 0.$$

The roots of the polynomial are

$$\lambda = \frac{1}{2}c\left(\overline{\boldsymbol{v}}\right) \pm \frac{1}{2}\sqrt{c\left(\overline{\boldsymbol{v}}\right)^2 - 4a\left(\overline{\boldsymbol{v}}\right)}.$$

A sufficient condition for the roots to lie in the half-plane $\mathbb{C}_-$ is that $a\left(\overline{\boldsymbol{v}}\right), c\left(\overline{\boldsymbol{v}}\right) < 0$, which is easily achieved by setting $g_{i,j} < 0$ for all i and $j = 1, 2$ via (6). □

Frozen stability entails asymptotic stability of the linearized system, and by proxy of the full nonlinear system (3) when the schedulers are kept constant, i.e., if $\boldsymbol{v}_t \equiv \overline{\boldsymbol{v}}$ for all t. Frozen stability is, however, insufficient to guarantee stronger notions of stability, e.g., asymptotic stability in the Lyapunov sense, when schedulers and inputs follow arbitrary trajectories. The following result, however, ensures a weaker notion of stability for the autonomous nonlinear system (namely, the one obtained by setting $\boldsymbol{u}_t \equiv \boldsymbol{0}$ for all t), provided that the schedulers do not change too rapidly.

Definition 1 (*Robust Stability*)**.** An autonomous LPV system

$$\dot{\boldsymbol{z}}_t = \boldsymbol{A}\left(\boldsymbol{v}_t\right)\boldsymbol{z}_t \tag{A.6}$$

is said to be "robustly stable" if there exists a quadratic form

$$V\left(\boldsymbol{z}_t, \boldsymbol{v}_t\right) = \boldsymbol{z}_t^\top P(\boldsymbol{v}_t)\boldsymbol{z}_t, \quad P(\boldsymbol{v}_t) \succ 0, \tag{A.7}$$

which is a Lyapunov function for the system.

Proposition 1 (*Proposition 2.3.8 from Briat, 2015*)**.** *Consider an autonomous LPV system (A.6), with time-varying schedulers $\boldsymbol{v}_t$. Then, the following statements hold:*

1. *If system (A.6) is robustly stable, then the spectrum of $\boldsymbol{A}(\boldsymbol{v})$ is contained in the left half-plane $\mathbb{C}_-$ and bounded away from the imaginary axis for all $\boldsymbol{v}$; furthermore, the faster $\boldsymbol{v}_t$ varies, the farther are the eigenvalues of $\boldsymbol{A}$ from the imaginary axis;*
2. *If the spectrum of $\boldsymbol{A}(\boldsymbol{v})$ lies in the half-plane $\mathbb{C}_-$ and is bounded away from the imaginary axis for all $\boldsymbol{v}$, then the system (A.6) is robustly stable for sufficiently slow-varying $\boldsymbol{v}_t$.*

Proposition 1 ensures that, given constraints of the kinds of Examples 1–2, an MoE model always fulfills Robust Stability, provided that the schedulers $\boldsymbol{v}_t$ vary slowly enough.

Data availability

The data that has been used is confidential.

References

Ahn, J., & Sentis, L. (2021). Nested mixture of experts: Cooperative and competitive learning of hybrid dynamical system. (arXiv:2011.10605), http://dx.doi.org/10.48550/arXiv.2011.10605, arXiv.

Baldacchino, T., Cross, E. J., Worden, K., & Rowson, J. (2016). Variational Bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems. *Mechanical Systems and Signal Processing*, *66–67*, 178–200. http://dx.doi.org/10.1016/j.ymssp.2015.05.009.

Baldacchino, T., Worden, K., & Rowson, J. (2017). Robust nonlinear system identification: Bayesian mixture of experts using the t-distribution. *Mechanical Systems and Signal Processing*, *85*, 977–992. http://dx.doi.org/10.1016/j.ymssp.2016.08.045.

Ballin, M. G. (1988). *A High fidelity real-time simulation of a small turboshaft engine*: *Tech. Rep. No. NAS 1.15:100991*.

Bischof, R. (2022). Mixture-of-experts-ensemble meta-learning for physics-informed neural networks. M. A.

Bishop, C. M., & Svensen, M. (2012). Bayesian hierarchical mixtures of experts. (arXiv:1212.2447), http://dx.doi.org/10.48550/arXiv.1212.2447, arXiv.

Briat, C. (2015). Linear parameter-varying and time-delay systems: analysis, observation, filtering & control. In *Advances in delays and dynamics*: *Vol. 3*, Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-662-44050-6.

Brotherton, T., Volponi, A., Luppold, R., & Simon, D. (2003). eSTORM: enhanced self tuning on-board real-time engine model. In *2003 IEEE aerospace conference proceedings (cat. no.03TH8652)*: *Vol. 7*, (pp. 3075–3086). Big Sky, Montana, USA: IEEE, http://dx.doi.org/10.1109/AERO.2003.1234150.

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, *113*(15), 3932–3937. http://dx.doi.org/10.1073/pnas.1517384113.

Callaham, J. L., Koch, J. V., Brunton, B. W., Kutz, J. N., & Brunton, S. L. (2021). Learning dominant physical processes with data-driven balance models. *Nature Communications*, *12*(1), 1016. http://dx.doi.org/10.1038/s41467-021-21331-z.

Castiglione, T., Perrone, D., Strafella, L., Ficarella, A., & Bova, S. (2023). Linear model of a turboshaft aero-engine including components degradation for control-oriented applications. *Energies*, *16*(6), 2634. http://dx.doi.org/10.3390/en16062634.

Consumi, M., & Agostino, L. (1998). A statistical inference approach to gas path analysis of a turbofan. In *34th AIAA/ASME/SAE/ASEE joint propulsion conference and exhibit*. American Institute of Aeronautics and Astronautics, http://dx.doi.org/10.2514/6.1998-3551.

Corliss, D. (1982). *A helicopter handling-qualities study of the effects of engine response characteristics, height-control dynamics, and excess poweron nap-of-the-Earth operations*: *Tech. Rep.*, U.S. Army Research and Technology Laboratories (AVRADCOM), U.S. Army Aeromechanics Laboratory, NASA Ames Research Center.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. (arXiv:1702.08608), http://dx.doi.org/10.48550/arXiv.1702.08608, arXiv.

Dreier, M. E. (2017). Introduction to the flight simulation of helicopters and tiltrotors. In *AIAA education series*, *Introduction to helicopter and tiltrotor flight simulation* (2nd ed.). (pp. 1–18). American Institute of Aeronautics and Astronautics, Inc. http://dx.doi.org/10.2514/5.9781624105135.0001.0018.

Duyar, A., Eldem, V., Merrill, W., & Guo, T.-H. (1994). Fault detection and diagnosis in propulsion systems - A fault parameter estimation approach. *Journal of Guidance, Control, and Dynamics*, *17*(1), 104–108. http://dx.doi.org/10.2514/3.21165.

Duyar, A., Gu, Z., & Litt, J. S. (1995). A simplified dynamic model of the T700 turboshaft engine. *Journal of the American Helicopter Society*, *40*(4), 62–70. http://dx.doi.org/10.4050/JAHS.40.62.

Ekinci, S., & Yavrucuk, I. (2020). Fast engine model for FMU-less small turbojet engines. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, *234*(2), 416–427. http://dx.doi.org/10.1177/0954410019867013, (Accessed 2024-06-13).

Etienam, C., Law, K., Wade, S., & Zankin, V. (2024). Fast deep mixtures of Gaussian process experts. *Machine Learning*, *113*(3), 1483–1508. http://dx.doi.org/10.1007/s10994-023-06491-x.

Fasel, U., Kutz, J. N., Brunton, B. W., & Brunton, S. L. (2022). Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *478*(2260), Article 20210904. http://dx.doi.org/10.1098/rspa.2021.0904.

Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. (arXiv:2101.03961), http://dx.doi.org/10.48550/arXiv.2101.03961, arXiv.

Fishwick, P. A. (1993). A simulation environment for multimodeling. *Discrete Event Dynamic Systems: Theory and Applications*, *3*(2–3), 151–171. http://dx.doi.org/10.1007/BF01439847.

Fishwick, P., Narayanan, N., Sticklen, J., & Bonarini, A. (1994). A multimodel approach to reasoning and simulation. *IEEE Transactions on Systems, Man and Cybernetics*, *24*(10), 1433–1449. http://dx.doi.org/10.1109/21.310527.

Green, M. D., Duyar, A., & Litt, J. S. (1997). Model-based fault diagnosis for turboshaft engines. *IFAC Proceedings Volumes*, *30*(18), 73–78. http://dx.doi.org/10.1016/S1474-6670(17)42383-4.

Gu, Z., Pang, S., Zhou, W., Li, Y., & Li, Q. (2022). An online data-driven LPV modeling method for turbo-shaft engines. *Energies*, *15*(4), 1255. http://dx.doi.org/10.3390/en15041255.

Härkönen, T., Wade, S., Law, K., & Roininen, L. (2022). Mixtures of Gaussian process experts with SMC^2. (arXiv:2208.12830), arXiv.

Hassberger, J. A., & Lee, J. C. (1989). A simulation-based expert system for nuclear power plant diagnostics. *Nuclear Science and Engineering*, *102*(2), 153–171. http://dx.doi.org/10.13182/NSE89-A23640.

Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., & Chi, E. (2021). Dselect-k: differentiable selection in the mixture of experts with applications to multi-task learning. In *Advances in neural information processing systems*: *Vol 34*, (pp. 29335–29347). Curran Associates, Inc.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87. http://dx.doi.org/10.1162/neco.1991.3.1.79.

Johansen, T., & Babuska, R. (2003). Multiobjective identification of Takagi-Sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems*, *11*(6), 847–860. http://dx.doi.org/10.1109/TFUZZ.2003.819824.

Jordan, M. I., & Jacobs, R. A. (1993). Hierarchical mixtures of experts and the EM algorithm. In *Proceedings of 1993 international conference on neural networks*. Nagoya, Japan: IEEE, http://dx.doi.org/10.1109/IJCNN.1993.714125, i–xxxxiii.

Kaiser, E., Kutz, J. N., & Brunton, S. L. (2018). Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *474*(2219), Article 20180335.

Kaptanoglu, A. A., Callaham, J. L., Aravkin, A., Hansen, C. J., & Brunton, S. L. (2021). Promoting global stability in data-driven models of quadratic nonlinear dynamics. *Physical Review Fluids*, *6*(9), Article 094401. http://dx.doi.org/10.1103/PhysRevFluids.6.094401.

Kim, F. (1992). Analysis of propulsion system dynamics in the validation of a high-order state space model of the UH-60. In *Guidance, navigation, and control and co-located conferences*, *Flight simulation technologies conference*. American Institute of Aeronautics and Astronautics, http://dx.doi.org/10.2514/6.1992-4150.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. (arXiv:1412.6980), arXiv.

Kurzke, J. (2009). Model based gas turbine parameter corrections. In *ASME turbo expo 2003, collocated with the 2003 international joint power generation conference* (pp. 91–99). American Society of Mechanical Engineers Digital Collection, http://dx.doi.org/10.1115/GT2003-38234.

Leoni, J., Breschi, V., Formentin, S., & Tanelli, M. (2024). Explainable data-driven modeling via mixture of experts: Towards effective blending of grey and black-box models. (arXiv:2401.17118), arXiv.

Litt, J., Kurtkaya, M., & Duyar, A. (1995). Sensor fault detection and diagnosis for a T700 turboshaft engine. *Journal of Guidance, Control, and Dynamics*, *18*(3), 640–642. http://dx.doi.org/10.2514/3.21439.

Mao, H., Guo, Y., Li, R., & Lai, C. (2019). Versatile simulation platform for turboshaft engine control system. In *2019 Chinese control conference* (pp. 7211–7216). http://dx.doi.org/10.23919/ChiCC.2019.8865902.

Martinez-Cantin, R. (2014). Bayesopt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, *15*(115), 3915–3919.

Masoudnia, S., & Ebrahimpour, R. (2014). Mixture of experts: A literature survey. *Artificial Intelligence Review*, *42*(2), 275–293. http://dx.doi.org/10.1007/s10462-012-9338-y.

Mihaloew, J. R., Ballin, M. G., & Ruttledge, G. (1988). *Rotorcraft flight-propulsion control integration: An eclectic design concept*: *Technical Report No. 2815*, NASA.

Molnar, C., Casalicchio, G., & Bischl, B. (2020). Interpretable machine learning – A brief history, state-of-the-art and challenges. In I. Koprinska, M. Kamp, A. Appice, C. Loglisci, L. Antonie, A. Zimmermann, R. Guidotti, O. Ozgobek, R. P. Ribeiro, R. Gavaldà, J. a. Gama, L. Adilova, Y. Krishnamurthy, P. M. Ferreira, D. Malerba, I. Medeiros, M. Ceci, G. Manco, E. Masciari, Z. W. Ras, P. Christen, E. Ntoutsi, E. Schubert, A. Zimek, A. Monreale, P. Biecek, S. Rinzivillo, B. Kille, A. Lommatzsch, & J. A. Gulla (Eds.), *ECML pKDD 2020 workshops*: *Vol. 1323*, (pp. 417–431). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-65965-3_28.

Montero Jimenez, J. J., Schwartz, S., Vingerhoeds, R., Grabot, B., & Salaün, M. (2020). Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems*, *56*, 539–557. http://dx.doi.org/10.1016/j.jmsy.2020.07.008.

Montero-Jiménez, J. J., & Vingerhoeds, R. A. (2018). Enhancing operational fault diagnosis by assessing multiple operational modes. In *12e conférence internationale de modélisation, optimisation et sIMulation*.

Paniccia, D., Tucci, F., Guerrero, J., Capone, L., Sanguini, N., Benacchio, T., & Bottasso, L. (2025). A supervised machine-learning approach for turboshaft engine dynamic modeling under real flight conditions. *Aeronautical Journal*, 1–25. http://dx.doi.org/10.1017/aer.2025.10024.

Qin, S., & McAvoy, T. (1996). Nonlinear FIR modeling via a neural net PLS approach. *Computers & Chemical Engineering*, *20*(2), 147–159. http://dx.doi.org/10.1016/0098-1354(95)00011-P.

Royer, A., Karmanov, I., Skliar, A., Bejnordi, B. E., & Blankevoort, T. (2023). Revisiting single-gated mixtures of experts. (arXiv:2304.05497), arXiv.

Selesnick, I., Lanza, A., Morigi, S., & Sgallari, F. (2020). Non-convex total variation regularization for convex denoising of signals. *Journal of Mathematical Imaging and Vision*, *62*(6), 825–841. http://dx.doi.org/10.1007/s10851-019-00937-5.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: the sparsely-gated mixture-of-experts layer. (arXiv:1701.06538), http://dx.doi.org/10.48550/arXiv.1701.06538, arXiv.

Sheng, H., Zhang, T., & Jiang, W. (2016). Full-range mathematical modeling of turboshaft engine in aerospace. *International Journal of Turbo & Jet-Engines*, *33*(4), 309–317. http://dx.doi.org/10.1515/tjj-2015-0033.

Van Breugel, F., Kutz, J. N., & Brunton, B. W. (2020). Numerical differentiation of noisy data: A unifying multi-objective optimization framework. *IEEE Access*, *8*, 196865–196877. http://dx.doi.org/10.1109/ACCESS.2020.3034077.

Vangheluwe, H., De Lara, J., & Mosterman, P. (2002). An introduction to multi-paradigm modelling and simulation.

Volponi, A. J. (1998). Gas turbine parameter corrections.

Volponi, A. J. (2014). Gas turbine engine health management: past, present, and future trends. *Journal of Engineering for Gas Turbines and Power*, *136*(051201), http://dx.doi.org/10.1115/1.4026126.

Wei, Z., Jafari, S., Zhang, S., & Nikolaidis, T. (2021). Hybrid Wiener model: An on-board approach using post-flight data for gas turbine aero-engines modelling. *Applied Thermal Engineering*, *184*, Article 116350. http://dx.doi.org/10.1016/j.applthermaleng.2020.116350.

Xia, M., Richard Hahn, P., & Gustafson, P. (2020). A Bayesian mixture of experts approach to covariate misclassification. *The Canadian Journal of Statistics*, *48*(4), 731–750. http://dx.doi.org/10.1002/cjs.11560.

Yazar, I. (2018). Simulation of a high fidelity turboshaft engine-alternator model for turboelectric propulsion system design and applications. *International Journal of Turbo & Jet-Engines*, http://dx.doi.org/10.1515/tjj-2018-0036.

Yazar, I., Anagun, Y., & Isik, S. (2024). Predicting compressor mass flow rate using various machine learning approaches. *International Journal of Turbo & Jet-Engines*, http://dx.doi.org/10.1515/tjj-2023-0105.

Yu, M., Yang, X., & Liu, X. (2021). LPV system identification with multiple-model approach based on shifted asymmetric Laplace distribution. *International Journal of Systems Science*, *52*(7), 1452–1465. http://dx.doi.org/10.1080/00207721.2020.1859158.

Yuksel, S. E., Wilson, J. N., & Gader, P. D. (2012). Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(8), 1177–1193. http://dx.doi.org/10.1109/TNNLS.2012.2200299.

Zhang, T., Bokrantz, R., & Olsson, J. (2022). A similarity-based Bayesian mixture-of-experts model. (arXiv:2012.02130), http://dx.doi.org/10.48550/arXiv.2012.02130, arXiv.

Zhang, L., & Schaeffer, H. (2018). On the convergence of the SINDy algorithm. (arXiv:1805.06445), arXiv.

Zhao, J., Zhao, Z., Shi, L., Kuang, Z., & Liu, Y. (2023). Collaborative mixture-of-experts model for multi-domain fake news detection. *Electronics*, *12*(16), 3440. http://dx.doi.org/10.3390/electronics12163440.

Zheng, Q., Xu, Z., Zhang, H., & Zhu, Z. (2018). A turboshaft engine NMPC scheme for helicopter autorotation recovery maneuver. *Aerospace Science and Technology*, *76*, 421–432. http://dx.doi.org/10.1016/j.ast.2018.01.034.