Engineering Notes

Machine Learning Models for Multirotor Performance Prediction

Jason Cornelius* and Sven Schmitz†
Pennsylvania State University, University Park, Pennsylvania 16802

https://doi.org/10.2514/1.C037460

Nomenclature

D	=	rotor diameter, m
Q	=	rotor torque, N · m
S	=	inter-rotor spacing as percent of diameter
SA	=	shaft angle, deg, angle between freestream velocity and rotor disk plane, negative nose-down
T	=	rotor thrust, N
V	=	flight speed, m/s
V_x	=	$V * \cos(SA)$, rotor edgewise (forward) speed, m/s
V_z	=	$-V * \sin(SA)$, rotor axial (vertical) speed, m/s
v_h	=	equivalent hover induced velocity, $\sqrt{T/(2 * \text{density} * A)}$, m/s

I. Introduction

THE field of multirotor flight vehicles is continually attracting increased attention from large aerospace companies all the way to venture-capital-backed startups in Silicon Valley. More than 700 conceptual designs have been proposed within the urban air mobility segment alone in the last 10 years [1]. The potential utility of these vehicles is still being explored and broadened. NASA has recently been getting involved with the design and use of multirotor configurations for planetary exploration after two decades of conceptual design [2–6]. Many of the conceptual designs exist only on paper or in sketches, but some have flown prototypes here on Earth in pursuit of FAA certification, while still others fly on different planets [7–9]. Multirotor aircraft technology also continues to rapidly expand in other sectors, such as the commercial hobby drone market and even military applications.

Although there are many different configurations for multirotor vehicles, the basic components are consistent across many of the designs. The large differentiator comes down to the vehicle control method to achieve a desired response, i.e., using blade collective and cyclic control or controlling the speed of a fixed-pitch rotor. Many of these multirotor aircraft are using the second approach and, more specifically, can be categorized as stiff, fixed-pitch, RPM-controlled rotors. This is very different from conventional rotorcraft operation and, as such, has opened the door for novel techniques in the design and analysis of these vehicles.

Several recent works have attempted to increase knowledge and understanding of these multirotor systems. Wind tunnel testing at the NASA Ames Research Center analyzed five multirotor unmanned aerial systems in the Army 7-ft by 10-ft Wind Tunnel [10,11]. Computational fluid dynamics (CFD) has also been used to analyze these systems [12–14]. A few examples of advanced computational methods analyzing these multirotor configurations include those of unstructured overset grids by Xu and Ye [15] and vortex particle methods by Singh and Friedmann [16]. Reviews of rotorcraft and multirotor analysis methods can be found in Refs. [17,18], which document recent methods used for both rotor performance and loads. Although much progress has been made in the capabilities to predict these quantities, the highest-fidelity approaches remain computationally prohibitive for more than a handful of cases in detailed design. Low-fidelity tools appropriate for rapid conceptual design can be used to generate much more output, however, at a much lower level of confidence compared to the higher-fidelity CFD. As such, there exists a need to cleverly combine the fidelity of various predictive methods to obtain efficient yet accurate models of multirotor performance.

This work documents the combination of various machine learning (ML) methods with a midfidelity CFD approach. The ML approaches are used to develop surrogate models for rotor performance based on a finite number of multirotor CFD simulations. The result is a highly computationally efficient predictive tool for rotor performance that has suitable accuracy for both conceptual and preliminary design while being fast enough for real-time simulation.

II. Background

Many concepts and approaches of ML have been around for some time, with the term itself dating back to 1959, but their adoption for aerospace applications has been slower than for computer science and other data-driven fields. The typical aerospace application involves training ML tools with data derived from either experimental testing or CFD, and then using this tool to further predict performance under other conditions. Critics of ML in aerospace applications argue that an ML black box is not an appropriate substitute for modeling the complicated physics as we currently understand it, such as the Navier–Stokes equations. There has been much progress in the last several years, however, to develop methodology for using ML in an appropriate manner to obtain substantial reductions in computational cost for a given calculation. In one example, Raissi, Perdikaris, and Karniadakis used ML to predict the three-dimensional flowfield for a cylinder in crossflow [19,20]. Raissi et al. used physics-informed neural networks (PI-NN), meaning that they used partial differential equations within the neural networks (NNs), to guide the ML tool to a solution of the flowfield characteristics. Notably, they also predicted the flowfield pressure without using any pressure inputs in the training data set, which shows the power of a PI-NN. Similarly, Wang et al. successfully used a PI-NN for turbulence modeling [21]. Although PI-NN can still be computationally expensive, which may make it less appealing than other ML approaches for some applications, it has been successfully used to solve problems with numerically unstable solutions such as shock discontinuities while avoiding the Gibbs phenomenon [22]. PI-NN has also successfully been used to upscale experimentally acquired particle image velocimetry (PIV) data to obtain the full three-dimensional flowfield ahead of and behind the PIV plane [23]. Karniadakis et al. recently provided a thorough review of PI-NN methods [24].

Another subset of ML is deep regression in deep neural networks (DNNs). Martinez et al. developed a DNN approach to reconstruct the vibration spectra of rotorcraft components with 95% accuracy for health usage and monitoring systems [25]. The work states that deep learning is a revolutionary aspect of ML and can be used on

Received 27 March 2023; revision received 19 January 2024; accepted for publication 26 March 2024; published online 3 May 2024.

*Aerospace Engineer. Member AIAA.
†Professor, Aerospace Engineering. Associate Fellow AIAA.

highly nonlinear multivariate data with additional layers (parameters) improving the capacity for high-dimensionality and nonlinear observations. The group also reports that while random search algorithms for tuning the DNN hyperparameters are more effective than a grid search method, evolutionary algorithms and Bayesian search methods may still provide a more accurate result. In general, the work found deep and wide networks to provide the most accurate surrogate models. Additional resources on deep learning support the claim that it is appropriate for mapping high-dimensional data [26].

Another study uses fully connected, recurrent, and convolutional neural networks (FCNN, RNN, CNN) to implement ML surrogate models in high-fidelity rotorcraft CFD to achieve speedup increases while retaining much accuracy [27]. The CFD program developed through these efforts, ROAM-ML, is being supported by the U.S. Army's Computational Aeromechanics group at NASA Ames Research Center [28,29]. The solver uses an actuator line representation for the rotor set in the Helios three-dimensional flow solver. ML surrogate models are used to create the source terms in the actuator line model using virtual sensors ahead of the blades instead of the conventional approach with C81 look-up tables. Similar to previous findings, the convolutional layers are found to be efficient at reducing complex data to a lower-dimensionality latent space, which is an approach similar to proper orthogonal decomposition (POD), and this approach has been pursued to further develop the work.

Chatterjee et al. also used a variety of ML approaches to model the effects of manufacturing variability on helicopter rotor dynamics, power required, and rotor stability [30]. The group used CNNs, random forests, support vector machines (SVMs), and adaptive Gaussian process regression (GPR) approaches to model the effects of stochastic manufacturing error and uncertainty via probability density functions. The work argues that surrogate models are more computationally efficient than sampling-based techniques such as Monte Carlo simulation. Both methods require high-fidelity simulations, but in principle, the ML surrogate models can be applied with a much smaller training data set, reducing the number of higher-fidelity simulations needed. The group found GPR and CNNs to create the most accurate surrogate models. Another group used a kriging approach, which is a form of GPR, to model and reduce adverse aerodynamic interactions between a representative UH-60 main rotor and fuselage [31]. Another work successfully applied GPR to linear parameter-varying models for the linearization of flight controls in real-time simulation [32]. The work gives a good overview of Gaussian process modeling and the associated mathematics. One key insight is that GPR scales in complexity with the number of observations cubed, which means that GPR quickly loses computational efficiency as the size of a dataset increases. One large advantage, however, of using GPR models is that they can provide an estimate of the model uncertainty at a similar computational cost to training the model. This means that GPRs can be used to iteratively add more simulations where the uncertainty is deemed to be the highest, thus increasing the accuracy of the models with the least computational cost. Additional resources on ML implementation for applications in fluid mechanics can be found in the reference section [33–36].

Specific to rotorcraft design and performance estimation, Allen et al. recently used ParFoil and simulation data from the two-dimensional airfoil CFD tool ARC2D to train ML surrogate models for airfoil shape optimization [37]. The team then coupled the surrogate airfoil performance model with a rotor design module using the Department of Defense's Dakota optimization tool to reduce the rotor power coefficient of the UH-60A main rotor in forward flight and hover. The group used supervised ML approaches such as NNs, regression trees, and ensembles of trees. Another recent work from Joby Aviation [38] used 1800 OVERFLOW CFD simulations of a single propeller with nacelle to create a surrogate model for their rotor blade loads. GPR was used to first develop a surrogate model using results from a lower-fidelity Euler CFD code with a much higher simulation point density, and then a second GPR was used to account for the discrepancy between these lower-order results and the OVERFLOW simulation data.

Although these studies are making much progress toward leveraging ML in rotorcraft performance prediction applications, they still require access to supercomputers and millions of CPU hours, making them cost prohibitive for the average user [29]. Additionally, the authors do not desire to replace higher-fidelity computational methods such as CFD. On the contrary, this paper proposes the combination of midfidelity CFD training data with various ML approaches to derive a very efficient tool for predicting multirotor performance throughout the entire flight envelope from climb, through edgewise flight and descent, and even in vortex ring state and windmill brake state. The methodology developed in this work allows for a high-accuracy real-time prediction of multirotor performance throughout the full range of possible operating conditions. To the authors' knowledge, this is the first application of various ML techniques to identify the best approach for developing ML surrogate models for multirotor performance prediction.

In this work, a methodology is proposed for developing accurate ML surrogate models for the performance prediction of stiff multirotor aircraft. The major advantage of this approach is the realization of high-accuracy multirotor performance prediction in millionths of a second. The results suggest that the methodology and resulting surrogate models can be used for conceptual and preliminary design of multirotor vehicles, closed-loop simulation, Monte Carlo simulation, or in real-time flight dynamics models, controllers, and simulators.

III. Machine Learning Methods

Machine learning (ML) is a subset within the very large field of artificial intelligence. ML approaches and methods aim to determine relationships between data and enable a user to make predictions or determine outcomes based on new unseen data. The typical aerospace application involves training ML tools with data derived from either experimental testing or CFD, and then using this tool to further predict performance under conditions similar to but different from the training data set. The major branches of ML and a few examples of each are summarized in Fig. 1. This work focuses on the first branch of ML: supervised learning. Even within supervised learning, there are still many different approaches and algorithms. Table 1 summarizes the approaches used in this work. Several existing studies typically focus on one of the approaches from Table 1, based on assumptions about the data or which ML approach may be best suited to represent it. This can lead to inadequately representing the data, so this study considers all the approaches and builds on the ones showing the most promise for multirotor performance prediction.

IV. Rotor CFD Modeling for ML Surrogate Model Training Data

As hinted at earlier, the process of creating a training data set can at times outweigh the computational cost benefit of using ML. As such, one must balance the computational cost of the training set with the eventual use of the ML surrogate model. Figure 2 summarizes several approaches that could be used to create a multirotor performance training data set and is reproduced from Cornelius et al. [39].

The state-of-the-art approaches are on the right side of the chart, with the highest fidelity and the highest computational cost. On the left side are the blade-modeled approaches, which are much faster but of lower fidelity. The hybrid BEMT-URANS methodology, highlighted orange in the middle of the chart, mixes the speed of a blade-modeled approach with the improved accuracy of a URANS CFD-resolved inflow and wake. This provides a computational cost somewhere in between the low- and high-fidelity methods due to its blade-modeled description of the rotor. For time-averaged rotor performance metrics such as thrust, torque, roll moment, and pitching moment, this method provides accuracy comparable with the higher-fidelity methods. Although the blade-resolved models can provide more accurate information in the form of a time-accurate solution, a very carefully constructed model of the rotor blade and its cross-sectional (two-dimensional) airfoil performance can

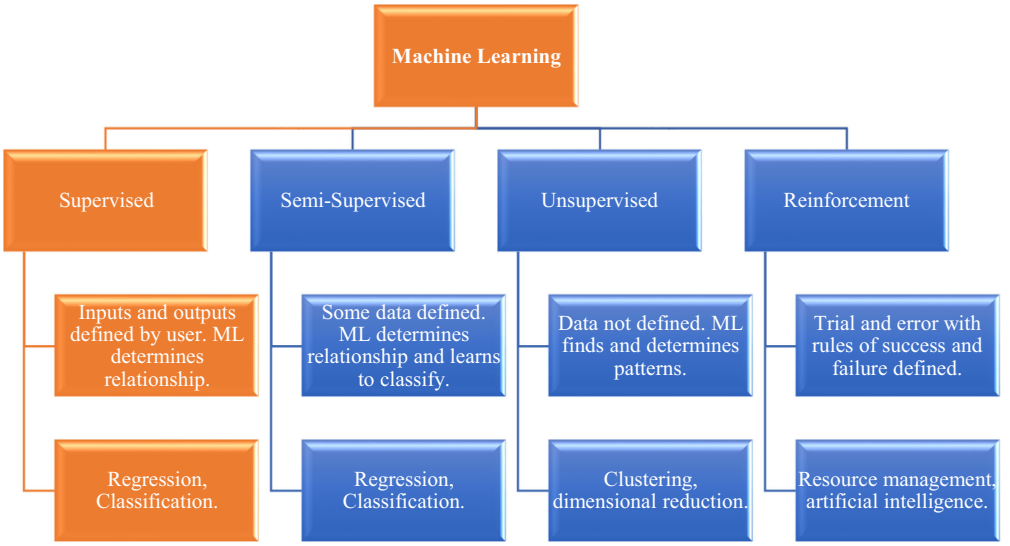

Fig. 1 Summary of machine learning approaches and some common applications.

Table 1 Supervised ML approaches used in this work

ML approach	Time to train	Pros	Cons
Linear regression	Low	Simple, fast	Linear only
Regression trees	Low	Fast	Difficult to use with complex data
Ensemble of trees	Low	Fast	
Support vector machines (SVM)	Medium	Fast	
Gaussian process regression (GPR)	Medium, scales poorly	Efficient with small–medium-sized data	Inefficient with large data sets
Neural network (NN)	High, scales well	Efficient with medium–large data sets	Hard to interpret, can be abstract

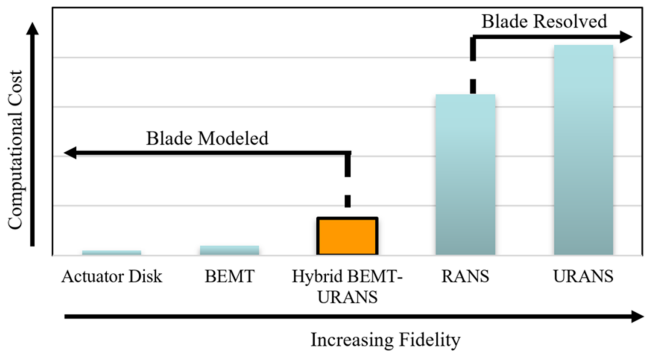

Fig. 2 Computational cost vs model fidelity for various rotor analysis approaches [39]. Reprinted with permission from Cornelius et al. [39].

provide very accurate results for the time-averaged values relevant to the flight control system and vehicle conceptual and preliminary design. As such, a hybrid BEMT-URANS multirotor approach, coaxial in this work, will be used to efficiently create a training data set to develop the subsequent ML surrogate models.

The commercial CFD toolset Rotorcraft CFD (RotCFD) is used in this work to implement the hybrid BEMT-URANS methodology. RotCFD is a self-contained program with provisions for the entire workflow, from geometry creation through CFD solution and visualization of the results [40–43]. RotCFD uses discretized momentum sources to interface the BEMT rotor module with a finite-volume, unstructured Cartesian grid system. Implicit time integration is used to solve the incompressible URANS equations with a two-equation k-ϵ turbulence closure and the SIMPLE-based solution method [44]. One of the main benefits of this hybrid BEMT-URANS methodology is that it enables multirotor CFD simulations to be massively parallelized on GPUs while still retaining a high level of accuracy for steady rotor performance predictions. This minimizes the time needed to create a training dataset of adequate size and accuracy for the objectives of this work on order thousands of flight conditions and accuracy of 5–10% for rotor thrust and power [45].

V. Creating the Hybrid BEMT-URANS CFD Training Dataset

The development and best practices for creating a high-fidelity rotor model using this hybrid BEMT-URANS methodology have been previously documented [46]. As such, only a very brief overview will be provided here. Cornelius and Schmitz [46] also detail the mass parallelization of the solver on graphical processing units, which will only be described here at a high level for its relevance to quickly creating a large training dataset of CFD-resolved multirotor performance data.

One of the most critical steps to assemble an accurate hybrid BEMT-URANS model is the development of high-fidelity airfoil performance tables. The CFD solver ARC2D, which was the precursor to OVERFLOW, has been used to create the C81 input decks following best practices for airfoil CFD simulation. More recently, the OVERFLOW2d CFD solver has been used to update the airfoil performance input decks being run in RotCFD. Roughly 3000 two-dimensional airfoil CFD simulations are run to create a single-rotor input deck, which consists of 13 radial stations to account for varying airfoil and thickness distributions, Reynolds number, and Mach number along the blade as a function of rotor RPM. The tables contain airfoil performance values for various RPMs of interest relevant to experimental wind tunnel testing at the NASA Langley Research Center [47].

A. RotCFD Model Creation

The next step was to assemble the RotCFD model. This work uses rotor performance data relevant to the NASA Dragonfly Mission [48]. A visualization of the CFD model is depicted in Fig. 3, with the rotor blades only included as a representation of the rotor model inputs. The rotor twist and chord distributions were discretized every 2.5% of the blade radius from the root cutout to the blade tip. As previously mentioned, the blade's airfoil performance tables consist of 13 radial stations from root to tip to accurately capture changes in airfoil performance with a local Reynolds number, Mach number, and airfoil profile. A grid-resolution study was carried out to ensure converged upper and lower rotor performance predictions. An example of the rotor grids used to interface the BEMT rotor module with the URANS structured grid is also included in Fig. 3. The CFD model

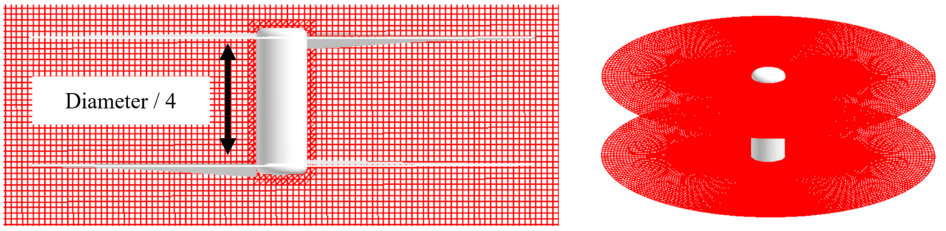

Fig. 3 Hybrid BEMT-URANS model of coaxial-rotor system, structured Cartesian grid (left) and rotor BEMT grids (right), $D = 1.35$ m, $S = 25\%$.

has approximately one million grid points and requires about 2.5 h to run each flight condition on a single GPU.

B. Parallelization of the Model on a Multi-GPU Desktop Workstation

Custom Linux bash scripting was developed for mass GPU parallelization of the RotCFD solver on multi-GPU machines. Information on two custom-built workstations, along with additional details on the bash scripting, is documented in Ref. [46]. The largest time savings come from implementing GPU concurrency for several simultaneous simulations on each card, with the total available GPU memory and CPU power determining the maximum number of simulations running concurrently. An example of the most likely application for this type of script is a matrix of flight performance predictions for rotor thrust and power with the parameterization of RPM, rotor shaft angle, and flight speed. In such a scenario, a table of several thousand flight conditions is required for closed-loop simulations to develop an aircraft's flight control system and for Monte Carlo simulations used to quantify the robustness of the flight controller and vehicle performance under modeling uncertainties. The mass parallelization across GPUs enabled by this hybrid BEMT-URANS methodology allows a matrix of approximately 1000 three-dimensional multirotor URANS CFD simulations to be completed in a few weeks on the previously mentioned GPU workstations.

VI. Machine Learning Approach

A. Creation of the Multirotor CFD Training Datasets

This work aims to reduce the total number of CFD cases required to build a multirotor performance matrix that can be used for controller design and performance analysis. This section will show various approaches in ML being applied to two training datasets consisting of 150 single-rotor and 840 coaxial-rotor flight conditions spanning the entire rotorcraft flight envelope. Flight conditions relevant to planned wind tunnel testing in support of the Dragonfly program were used.

Table 2 Parameterization of the CFD datasets

Parameter	Coaxial rotor	Single rotor
Shaft angle, SA	[−90, −75, −60, −45, −30, −15, −5, 0, 5, 15, 30, 45, 60, 75, 90] deg	[−90, −60, −30, −15, −5, 0, 15, 30, 60, 90] deg
Flight speed, V	[2.25, 6, 9] m/s	[2.25, 6, 9] m/s
RPM	[600, 750, 900, 1050, 1250]	[600, 750, 900, 1050, 1250]

The single-rotor CFD model is the same as the coaxial one previously described, but with the lower rotor removed, and is used first to quickly assess the best ML techniques for creating surrogate models of rotor performance. The RPM range simulated for the coaxial-rotor system went from 200 up to 1250, which corresponds to blade-tip Mach numbers of 0.07 and 0.46, respectively. Rotor shaft angles, SA, ranged from −90 to +90 deg, covering the full range of flight conditions from axial climb through axial descent. The shaft angle is measured between the freestream flow vector and the rotor disk plane. Simulated flight speeds ranged from 2.25 to 13.5 m/s. For the single-rotor dataset, the lowest RPM simulated was 600, and the highest flight speed was 9 m/s. For each of these parameters, the discretization was chosen to balance covering the full potential flight envelope while limiting the number of CFD-resolved simulations. The full parameterization for each dataset is reported in Table 2.

The flight conditions are a parameterization of the rotor shaft angle, flight speed, and RPM, and are mapped onto rotor aerodynamic state charts in Fig. 4. The y axis is a nondimensional vertical speed parameter, and the x axis is a nondimensional forward speed parameter. This allows mapping from the three-dimensional flight condition matrix to a lower two-dimensional latent space.

The areas of the chart are broken down into shallow and steep climbs, along with the same for descent. Additionally, tags are placed on the descending flight side to identify the vortex ring state (VRS) and windmill brake state (WBS) regions. The training datasets well surpass the typical range of rotorcraft flight conditions. A gap is observed between the hover condition at the origin of the graphs and the first semicircle of points plotted out from the origin. The innermost data points are a function of the minimum flight speed and maximum RPM. Since the rotors being analyzed will operate above 2.25 m/s outside of hover, the datasets cover the full anticipated flight envelope.

B. MATLAB Regression Learner App

Figure 1 and Table 1 provide some background information on the various supervised ML models explored in this work for the

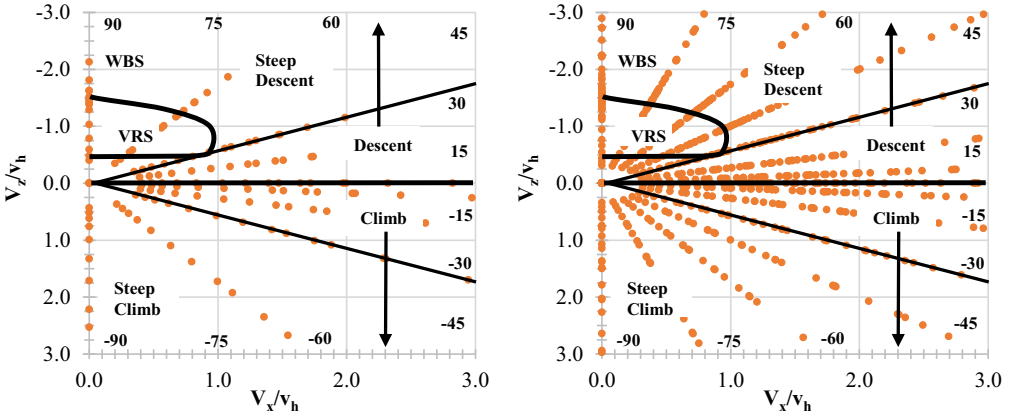

Fig. 4 Flight conditions in the single-rotor (left) and coaxial-rotor (right) CFD training datasets.

prediction of rotor performance. The MATLAB Regression Learner App was used to train, validate, and test these models. This tool allows the user to import datasets, train models, and analyze their performance all within a user-friendly graphical user interface. The MATLAB Regression Learner App help center on the MathWorks website provides ample detailed information on the specific implementation of the various models that will be discussed in this work [49].

VII. Results

A. Single-Rotor ML Surrogate Model Testing and Development

The various ML techniques from Table 1 were applied to the single-rotor CFD training dataset to develop surrogate models for predicting rotor thrust and torque. The resulting training time, R^2 values, and root-mean-squared error (RMSE) for the various ML models tested are reported in Table 3 for single-rotor thrust prediction. Inputs to the model were the parameterized flight conditions from Table 2, and the test statistics reported were calculated using both a fivefold cross-validation and a 10% hold-out test dataset for the prediction of rotor thrust. The k-fold cross-validation is one of the most common cross-validation approaches and prevents overfitting while still using the entire available dataset to train the model. This validation approach is particularly well suited for the small datasets used in this work. The approach segments the training data into five partitions and does five training iterations, where each iteration trains on a distinct set of four out of the five partitions. The last remaining partition in each iteration is the test set used to estimate the model's accuracy. The results for the five iterations are averaged to create the fivefold cross-validation metrics such as R^2 and RMSE. The final model is trained on all five partitions, but the averaged fivefold cross-validation test statistics are still used. True test statistics are then separately calculated using the 10% hold-out, which is all unseen data. The R^2 value is a nondimensional metric for how well the model represents the test dataset. The RMSE calculation, Eq. (1), provides a dimensional quantification of the mean error while also capturing the effect of outliers. A smaller value for the RMSE means a given model more precisely estimates the real data. The RMSE values are reported for both the k-fold validation and using the 10% hold-out data. The R^2 values are only reported for the 10% hold-out calculation. The surrogate model results for single-rotor thrust are reported in Table 3. The training times for the models are also reported. All models were trained in MATLAB using a single central processing unit (CPU) core.

Table 3 Single-rotor thrust, ML surrogate model testing

ML approach	Training time, s	RMSE, N validation	RMSE, N test	R^2 test
Support vector machine (SVM) kernel	18.2	169.5	108.2	−0.42
Linear regression	2.6	90.1	56.4	0.61
Stepwise linear regression	3.4	62.3	28.0	0.90
Fine regression tree	2.8	0.77	82.3	80.8
NN with Bayesian optimization, 30 iterations (1 L: 263 N)	169.5	58.5	20.4	0.95
Ensemble of trees Bayesian optimization, 30 iterations	95.2	34.6	12.9	0.98
Cubic support vector machine (SVM)	0.8	44.0	11.9	0.98
Trilayered NN (3 L: 10 N)	17.7	22.4	9.2	0.99
Wide NN (1 L: 100 N)	12.7	31.1	8.0	0.99
Trilayered wide NN (3L: 100 N)	24.2	20.3	4.8	1.00
GPR, rational quadratic	2.8	19.7	2.6	1.00
GPR, exponential	3.3	26.3	2.5	1.00
GPR, Matern 5/2	4.0	19.1	2.2	1.00
GPR with Bayesian optimization[a] 30 iterations, nonisotropic Matern 3/2	26.4	17.6	1.5	1.00

[a]Best-performing model.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}, \quad \text{where } x_i = \text{real value};$$
$$\hat{x}_i = \text{predicted value}; \; N = \text{number of values} \tag{1}$$

The model requiring the longest training time for a single training iteration was the Bayesian-optimized NN, which ran 30 iterations in 169.5 s on a single CPU core. The optimized NN had one layer with 263 neurons. Other, more basic models, however, performed much better. A trilayered wide NN, e.g., achieved an R^2 value of 1 and a test RMSE of 4.8 N. The model has three fully connected layers, each with 100 neurons, and uses ReLU activation. This trilayered NN was outperformed by all GPR models tested. The best GPR model was developed using Bayesian optimization on the model hyperparameters. This optimized GPR, which was the best performing of all surrogate models in Table 3, has an R^2 value of 1.0, has the lowest test RMSE of 1.5 N, and required 26.4 s for 30 training iterations. The resulting model hyperparameters used a nonisotropic Matern 3/2 kernel. Figure 5 summarizes the predicted versus true response for the optimized GPR model. The plot of residuals shows that the 10% hold-out was pulled from axial climb, edgewise, and descent conditions. The y axis of the residuals chart ranges from negative to positive 3 newtons, indicating the high accuracy of the model.

Since several models performed quite well, including an optimized ensemble of trees and a SVM, all models were again trained and tested for the prediction of single-rotor torque. The resulting training

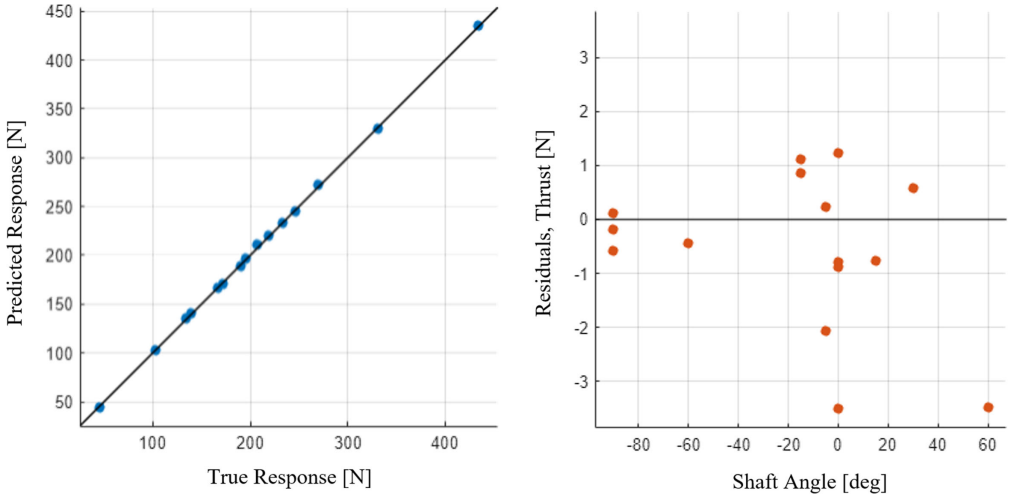

Fig. 5 Single-rotor thrust—best model results: GPR, Matern 3/2 predicted vs true response (left) and thrust residuals vs shaft angle (right).

times and test statistics for the single-rotor torque are reported in Table 4. The GPR model with Bayesian optimization again performed quite well, with 30 iterations yielding an RMSE of 0.8 N·m and an R^2 value of 0.99. The optimized GPR used the exponential kernel. Various optimization methods, including grid and random search were, used, but the Bayesian optimizer consistently produced superior results. Optimization on the NN was performed, but the trilayered wide NN with 100 neurons per layer was again the best performing of all the NNs and was actually the best of all models tested. The default GPR settings for various kernels also produced results close to the optimized GPR, although in a fraction of the training time. For larger datasets where the cost to train GPR models increases rapidly, the default GPR models may be a good-enough approach rather than training with Bayesian optimization.

The results for the best-performing model from Table 4, the trilayered wide NN, are reported in Fig. 6. The response plots again show very high accuracy of the surrogate model at predicting the unseen test data from the 10% hold-out. The residual plot shows test data covering the full range of shaft angle, from positive to negative 90 deg. The largest error is observed for the higher shaft angles, which correspond to challenging descent conditions where the rotor performance is more nonlinear and harder to model. Still, the model appears to predict the unseen data quite well, with a test R^2 value of 1.

Table 4 Single-rotor torque, ML surrogate model testing

ML approach	Training time, s	RMSE, N · m validation	RMSE, N · m test	R^2 test
SVM, Gaussian kernel	14.9	7.2	6.2	0.26
Linear regression	2.8	3.5	3.6	0.75
Stepwise linear regression	1.3	3.6	3.6	0.74
Fine regression tree	6.2	3.6	3.5	0.77
NN with Bayesian optimization, 30 iterations (2 layers: 117, 3 neurons)	169.1	1.9	0.9	0.98
Ensemble of trees Bayesian optimization, 30 iterations	197.3	2.2	1.2	0.97
Cubic support vector machine (SVM)	1.4	2.9	1.7	0.94
Trilayered NN (3 L: 10 N)	14.4	2.4	1.1	0.98
Wide NN (1 L: 100 N)	11.8	2.0	0.8	0.99
Trilayered wide NN (3L: 100 N)[a]	24.3	1.6	0.4	1.00
GPR, rational quadratic	4.1	1.9	0.8	0.99
GPR, exponential	3.4	1.7	0.7	0.99
GPR, Matern 5/2	4.2	2.1	0.9	0.99
GPR with Bayesian optimization, 30 iterations, exponential	28.9	1.7	0.8	0.99

[a]Best-performing model.

B. Coaxial-Rotor ML Surrogate Model Testing and Development

The best ML models from the single-rotor training study were used to create and test surrogate models for the coaxial-rotor system with the much larger 840 flight condition training dataset. The training data are again a parameterization of the flight speed, rotor shaft angle, and rotor speed, as reported in Table 2. The point density and range for this training dataset have been increased and include conditions in deeper descents and at additional rotor shaft angles of attack. The models were again checked by first using fivefold cross-validation and then testing against the 10% hold-out data. The ML surrogate model results for predicting upper rotor thrust in the coaxial system are reported in Table 5.

A trilayered NN with 100 neurons per layer resulted in the best-performing surrogate model for upper rotor thrust. The second-best model was a GPR developed with Bayesian optimization. The optimizer determined the squared exponential kernel to be the best GPR model for the dataset, which may suggest the exponential kernel to be a generally good approach for building ML surrogate models of rotor performance data. The best GPR kernel for the single-rotor thrust, however, was the Matern 3/2. For this reason, it is advisable to explore multiple available kernels when creating and training GPR models for this application. The best-performing model for upper rotor thrust prediction, the trilayered NN, has an R^2 value of 1 and a test RMSE of 6.5 Newtons. This model's response plot and residuals versus shaft angle for the test data are reported in Fig. 7.

With the much larger coaxial training dataset, which has 5.6 times the number of flight conditions as the single-rotor matrix, the GPR requires a much longer amount of time to train compared to the single-rotor dataset. The GPR was significantly faster than the NN approach for the single-rotor model training, but the larger coaxial model dataset required a similar order of magnitude between the best

Table 5 Upper rotor thrust, ML surrogate model testing

ML approach	Training time, s	RMSE, N validation	RMSE, N test	R^2 test
NN with Bayesian optimization, 30 iterations (3 layers: 4, 81, 16 neurons)	267.6	98.8	90.2	0.74
Trilayered NN (3 L: 10 N)	8.4	20.5	15.9	0.99
Wide NN (1 L: 100 N)	14.2	15.3	9.6	1.00
Trilayered wide NN (3L: 100 N)[a]	63.9	14.9	6.5	1.00
GPR, rational quadratic	80.2	12.5	8.2	1.00
GPR, exponential	91.3	13.7	8.2	1.00
GPR, Matern 5/2	98.4	12.6	8.0	1.00
GPR with Bayesian optimization 30 iterations, squared exponential	316.4	12.4	7.7	1.00

[a]Best-performing model.

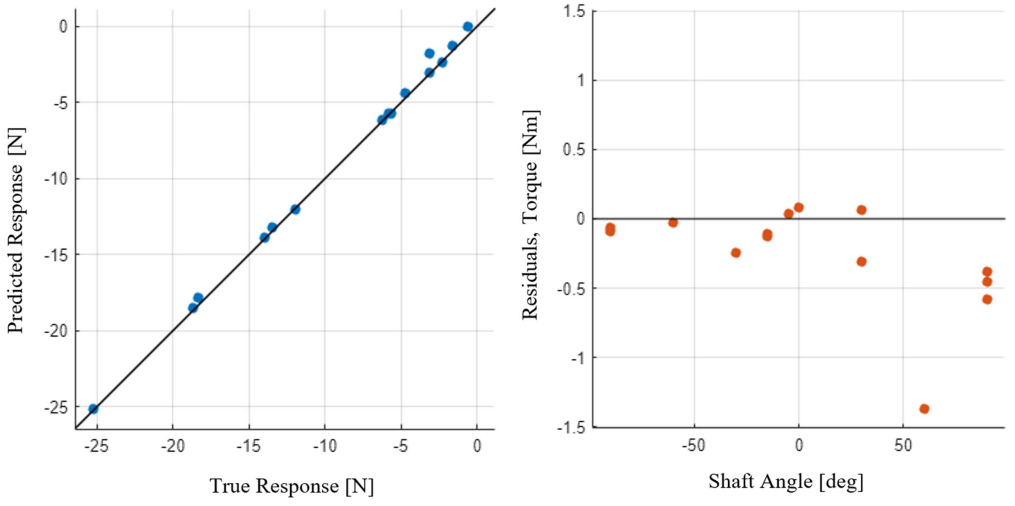

Fig. 6 Single-rotor torque (best model results): trilayered wide NN predicted vs true response (left) and torque residuals vs shaft angle (right).

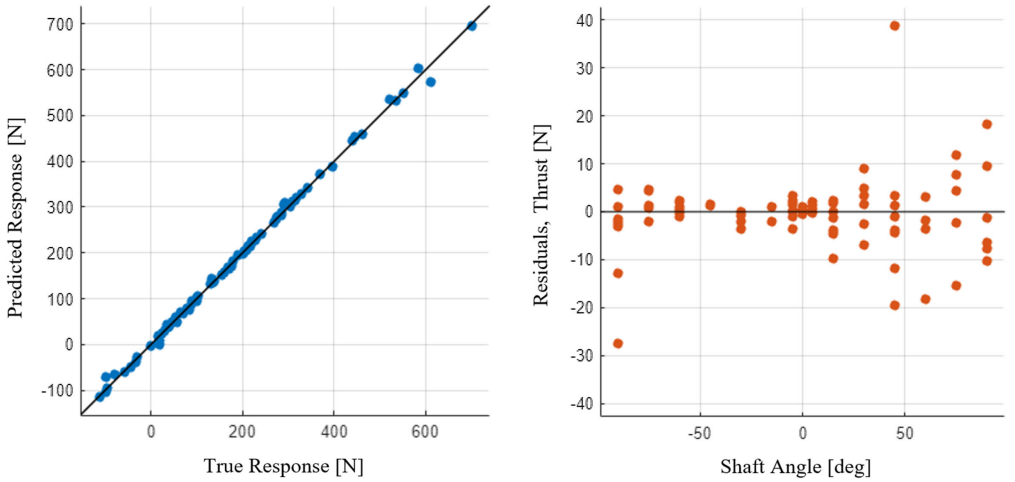

Fig. 7 Upper rotor thrust (best model results): GPR, squared exponential predicted vs true response (left) and thrust residuals vs shaft angle (right).

models from each method, which highlights the poor scalability of GPR. The relatively low number of inputs and outputs for this application of time-averaged rotor performance prediction, however, results in GPR still being a very suitable method, requiring only a few minutes for the 840-flight condition dataset.

The same ML methods were used to create surrogate models for the lower rotor thrust, and the results are reported in Table 6. Similar trends in the model performance are observed, although with slightly larger RMSE values than were reported for the upper rotor models in Table 5. For the lower rotor thrust prediction, GPR using the exponential kernel was observed to perform the best. The best-performing lower rotor thrust surrogate model results are plotted in Fig. 8. More scatter is observed in the prediction of lower rotor thrust, which, along with the higher RMSE values, suggests that the lower rotor is a bit harder to accurately model. This makes sense when considering the interactional aerodynamics of the coaxial-rotor system. The lower rotor ingests the wake of the upper rotor differently as a function of the coaxial-rotor system's shaft angle. For example, axial climb conditions have the full upper rotor wake impinging on the lower rotor, while edgewise conditions can have almost no rotor–rotor interactional effects. This means that, for some shaft angles, the lower rotor thrust is degraded by the additional inflow from the upper rotor, while in others it operates as if it is an isolated rotor.

Table 6 Lower rotor thrust, ML surrogate model testing

ML approach	Training time, s	RMSE, N validation	RMSE, N test	R^2 test
Trilayered NN (3 L: 10 N)	8.0	20.8	12.8	1.00
Wide NN (1 L: 100 N)	15.3	16.9	10.3	1.00
Trilayered wide NN (3L: 100 N)	62.6	18.1	14.1	1.00
GPR, rational quadratic	79.8	13.9	10.1	1.00
GPR, exponential[a]	89.7	15.6	9.2	1.00
GPR, Matern 5/2	97.6	13.9	10.1	1.00
GPR with Bayesian optimization, 30 iterations squared exponential	437.5	14.5	10.6	1.00

[a]Best-performing model.

For the coaxial-rotor torque, the same NNs and GPR models were trained and tested. The model training results are reported in Table 7 for the upper rotor and Table 8 for the lower rotor. The upper rotor torque appears to be better represented by the surrogate models, which is in agreement with the previous observation of the upper rotor thrust being better predicted since the rotor torque is in large part a function of rotor thrust. Both the upper and lower rotor torques were best predicted by GPR models. The upper rotor model was found using Bayesian optimization with a Matern 3/2 kernel. The best model for the lower rotor used a rational quadratic kernel.

When considering the results of Tables 5–8, GPR appears to be the best ML model for coaxial-rotor performance prediction. The best kernel, however, varies and indicates that multiple GPR kernels should be trained and tested for future applications. The results for the best GPR surrogate model for upper and lower rotor torque are reported in Figs. 9 and 10, respectively. Although the response plots for both upper and lower rotor models look quite good, the lower rotor does appear to have more scatter. When considering the residuals, the lower rotor again has slightly larger values. The largest residuals are

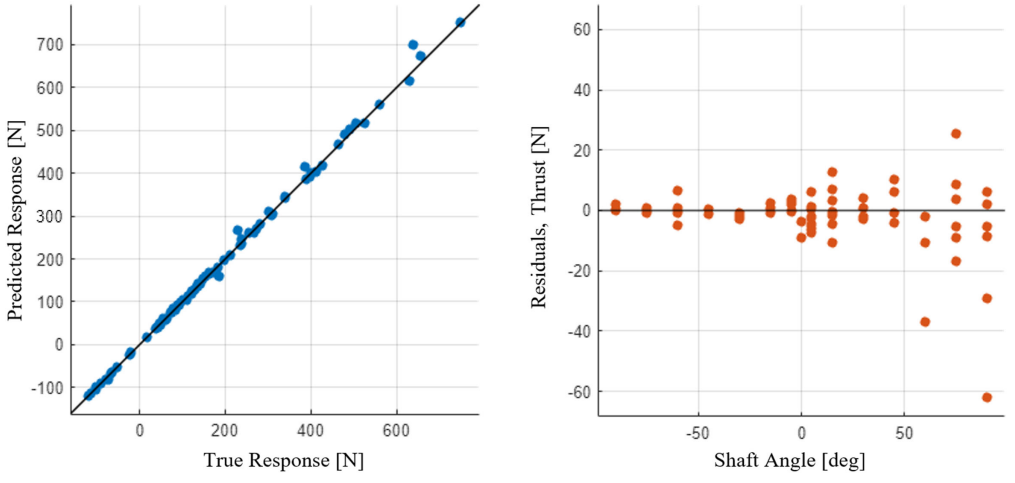

Fig. 8 Lower rotor thrust (best model results): GPR, squared exponential predicted vs true response (left) and thrust residuals vs shaft angle (right).

Table 7 Upper rotor torque, ML surrogate model testing

ML approach	Training time, s	RMSE, N · m validation	RMSE, N · m test	R^2 test
Trilayered NN (3 L: 10 N)	8.4	1.4	1.17	0.98
Wide NN (1 L: 100 N)	14.4	1.4	0.78	0.99
Trilayered wide NN (3L: 100 N)	61.7	1.2	0.62	1.00
GPR, rational quadratic	83.3	1.5	0.63	1.00
GPR, exponential	93.7	1.3	0.83	1.00
GPR, Matern 5/2	103.8	1.5	0.63	1.00
GPR with Bayesian optimization,[a] 30 iterations, Matern 3/2	440.4	1.1	0.35	1.00

[a]Best-performing model.

Table 8 Lower rotor torque, ML surrogate model testing

ML approach	Training time, s	RMSE, N · m validation	RMSE, N · m test	R^2 test
Trilayered NN (3 L: 10 N)	7.9	1.2	1.06	0.98
Wide NN (1 L: 100 N)	15.0	1.2	1.16	0.98
Trilayered wide NN (3L: 100 N)	60.0	0.9	0.70	0.99
GPR, rational quadratic[a]	78.0	1.0	0.58	1.00
GPR, exponential	88.4	1.0	0.70	0.99
GPR, Matern 5/2	99.8	1.1	0.62	0.99
GPR with Bayesian optimization, 30 iterations exponential	432.89	0.9	0.69	0.99

[a]Best-performing model.

again found at the highest positive shaft angles, corresponding to descent-type conditions where the rotors are in challenging aerodynamic flow states.

C. Testing the Computational Time of the Final ML Surrogate Models

The previous section documented the ability of the best single-rotor and coaxial-rotor surrogate models to predict rotor performance for unseen data by using the 10% hold-out data. This characterized the various models' ability to predict rotor performance for new, unseen flight conditions. Two practical uses of the resulting surrogate models are data upscaling and real-time simulation. To demonstrate the predictive speed, which is one of the major benefits of deriving these ML surrogate models, the training datasets were upscaled. Using the best single-rotor GPR surrogate model for predicting thrust, a 1.6-million-flight-condition performance table was generated. This table, which is roughly eleven thousand times larger than the training dataset, was created in about 5.9 s on a single core on the CPU. This equates to about 3.7 millionths of a second per thrust prediction. The best NN for single-rotor thrust prediction generated the same size table in 2.3 s, or 1.4 millionths of a second per flight condition.

The coaxial-rotor surrogate models were then used to generate the same 1.6-million-flight-condition table. This coaxial-rotor performance table was generated in 77.7 s using the best GPR models, which is 48 millionths of a second per flight condition, or 12 millionths of a second per calculation. The best-performing NN models for each calculation of the coaxial-rotor system generated the same table in 7.5 s, or 4.6 millionths of a second per flight condition and 1.1 millionths of a second per prediction. This shows the immense power of using ML surrogate models for look-up table performance generation, data upscaling, or even applications requiring real-time performance estimation. These flight conditions are plotted in Fig. 11 using the same rotor state chart as before. The much higher point density is observed covering the full possible flight envelope of all vehicle conditions. Such a table could be used in Monte Carlo simulation or as a look-up in flight dynamics software to increase the accuracy of linear interpolation on the look-up tables. Alternatively, the ML surrogate models can be directly programmed in for real-time continuous performance estimation, precluding the need for any interpolation between points within the data range.

To put the previous numbers into context for computational cost savings, an average flight condition from the 840 point coaxial-rotor training dataset required 2.5 h on a Tesla V100 high-performance GPU. Using the best-performing ML surrogate models, with R^2 values of 0.99–1.0 calculated using a 10% hold-out, the rotor performance was estimated in 48 millionths of a second per flight condition. The best-performing NNs, which were close behind the GPR models and had R^2 values of 0.99–1.0, predicted the coaxial-rotor performance in 4.6 millionths of a second per flight condition. ML critics often argue that creating the training dataset requires a lot of time and thus erodes the benefits of the ML approach. This application of creating a relevant high-point-density flight condition look-up table from a much smaller table, however, saved roughly 4,000,000 wall-clock compute hours, which is 456 years on the previously described GPU workstation or 4.6 years on 100 Tesla V100 GPUs using a supercomputer. Running that amount of CFD is clearly impractical, while the resulting ML flight performance table, with its associated accuracy, is highly applicable to many disciplines within aircraft design and analysis. The resulting ML flight performance table is discretized every 1 deg from axial climb through axial descent, every 10 RPM from 600 through 1250, and every 0.1 m/s flight speed from 0 through 13.5, which is a very reasonable, not superfluous, discretization for the purposes of controller design and Monte Carlo simulation of vehicle performance.

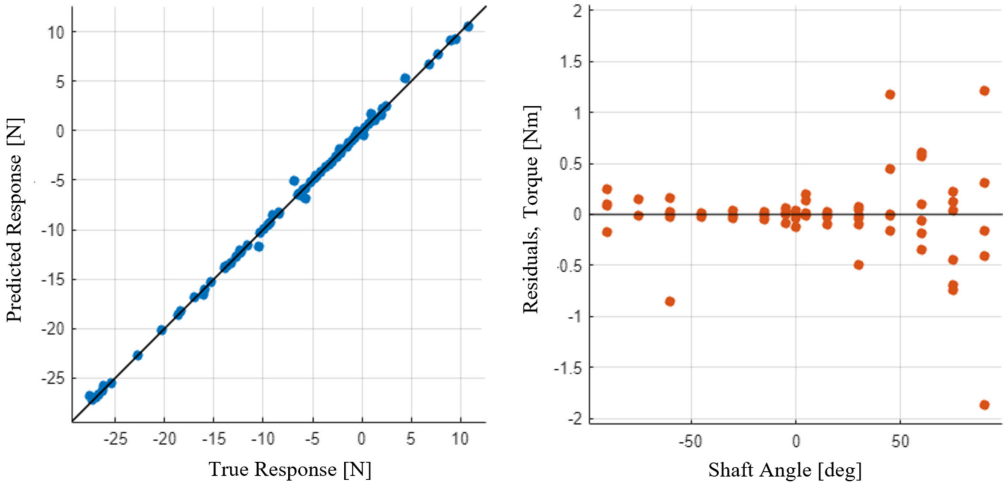

Fig. 9 Upper rotor torque (best model results): GPR, Matern 3/2 predicted vs true response (left) and torque residuals vs shaft angle (right).

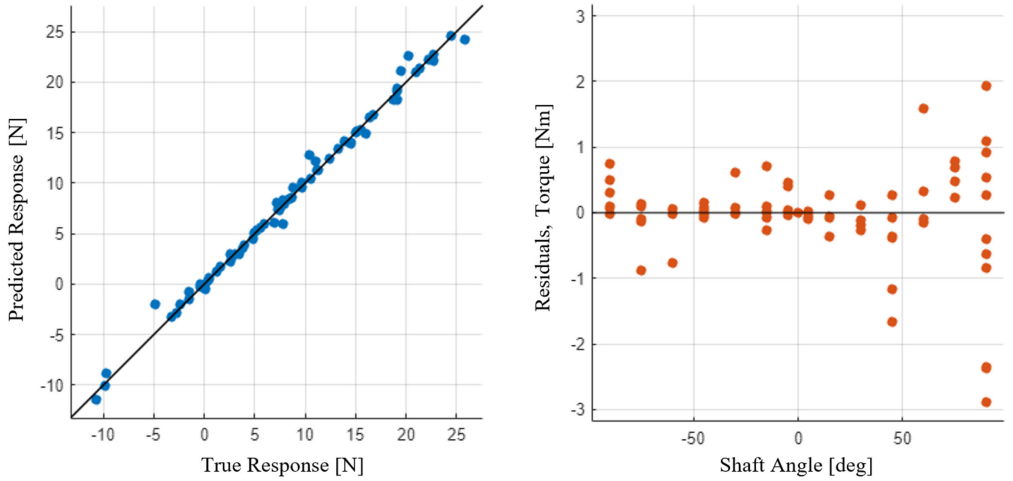

Fig. 10 Lower rotor torque (best model results): GPR, exponential predicted vs true response (left) and torque residuals vs shaft angle (right).

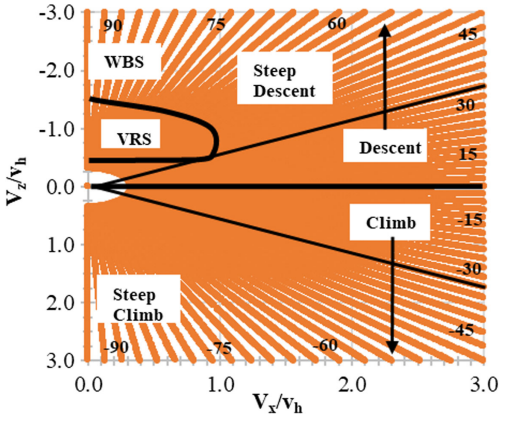

Fig. 11 ML generated a 1.6-million-flight-condition multirotor performance table generated in 7.5 s using NNs, or 77.7 s using GPR models.

VIII. Conclusions

The novelty of this work lies in the exploration of applying various supervised ML techniques to multirotor performance training datasets for the creation of multirotor performance surrogate models. The training datasets were created using a previously documented methodology for midfidelity CFD using a hybrid BEMT-URANS solver. The CFD solver has been validated in previous studies to an accuracy on the order of 5–10% for predicting multirotor steady thrust and torque. A single-rotor training dataset was created with 150 flight conditions, and a coaxial-rotor dataset was created with 840 flight conditions. Each of the three-dimensional CFD simulations required about 2.5 h on a GPU.

Various ML techniques were applied to the training datasets using the MATLAB Regression Learner App to create ML surrogate models. Several of the models showed high accuracy with R^2 values of 0.99–1.0 using a 10% hold-out from the data for testing. The models were trained to predict individual rotor thrust and torque. The two best approaches were consistently NNs and GPR. For most cases, the GPR model slightly outperformed the NNs, although both had R^2 values of 0.99–1.0 calculated using the hold-out data. Analyzing plots of the residual, or prediction error, for the various models typically showed the highest error for positive shaft angles. This corresponds to descent-type conditions, where the rotors are operating in a more complicated aerodynamic state. Additionally, higher scatter was observed in general for the lower rotor as compared to the upper rotor, which indicates the lower rotor performance is harder to capture. This makes sense when considering the flow physics, with the lower rotor ingesting the upper rotor wake as a function of shaft angle.

For the small single-rotor training dataset, the GPR models trained faster than the NN. The much larger coaxial-rotor training dataset, however, required comparable time from each approach. This suggests the NN may outperform the GPR from the perspective of training time if much larger training datasets were to be used. The best-performing surrogate models for both the single and coaxial-rotor systems were used to create a 1.6-million-flight-condition multirotor performance table. The single-rotor predictions required about 3.7 millionths of a second per flight condition using the best GPR models and about 1.4 millionths of a second using the best NN. For the coaxial-rotor performance, the same 1.6-million-flight-condition look-up table was generated in 77.7 s using the GPR models and only 7.5 s using the NNs on a single CPU core. This equates to about 48 and 4.6 millionths of a second per flight condition, respectively. This demonstrates the poor scalability of GPR models for this application and suggests that NNs are likely the best choice for larger rotor performance tables and fast prediction times.

These surrogate model prediction speeds can be put into context by comparing them to the 2.5 h required for each CFD simulation on a high-performance GPU-accelerated workstation. The nearly five-order-of-magnitude computational speedup with such high accuracy of performance prediction shows the power of applying ML techniques to create surrogate models for high-dimensionality nonlinear multirotor performance estimation. The resulting flight performance table saved roughly 4,000,000 computer wall-clock hours, or 4.6 years, on 100 Tesla V100 GPUs. The multirotor ML surrogate model methodology presented here can be applied to vehicle performance analysis, flight control design, Monte Carlo simulation, closed-loop simulation, and even real-time simulation.

Future studies should assess the applicability of this approach when complex interactional aerodynamics are involved, such as rotor–fuselage interactions. ML can also likely be leveraged in future studies to decrease the computational cost associated with developing surrogate models, including transient aerodynamics. Additionally, adding additional parameters such as rotor design and coaxial-rotor spacing could be incorporated to create surrogate models for conceptual and preliminary design optimization. As the number of parameters increases, the computational efficiency of some approaches used in this work will rapidly degrade, which may push the engineer toward a specific set of supervised ML methods, such as neural networks.

Acknowledgments

This research effort was funded through the NASA New Frontiers Dragonfly Program. The authors would also like to extend their gratitude to Nappinnai Rajagopalan and the late Ganesh Rajagopalan of Sukra Helitek, Inc., for their much support in using the RotCFD tool. Additionally, the authors would like to thank Nicholas Peters, Tove Aagren, Kristen Kallstrom, and Witold Koning of the NASA Ames Research Center for discussions around machine learning and applications to rotorcraft. Many thanks to Ethan Romander and

Carlos Pereyra of NASA Ames, along with Kirk Heller of Penn State, for the many long conversations around high-performance computing and Linux scripting that enabled this work.

References

[1] "eVTOL Aircraft Directory," Vertical Flight Soc., https://evtol.news/aircraft [retrieved 12 Jan. 2022].

[2] Balaram, J., Canham, T., Duncan, C., Golombek, M., Grip, H., Johnson, W., Maki, J., Quon, A., Stern, R., and Zhu, D., "Mars Helicopter Technology Demonstrator," *AIAA SciTech Forum*, AIAA Paper 2018-0023, Jan. 2018. https://doi.org/10.2514/6.2018-0023

[3] Young, L., Briggs, G., Derby, M., and Aiken, E., "Use of Vertical Lift Planetary Aerial Vehicles for the Exploration of Mars," *NASA Headquarters and Lunar and Planetary Institute Workshop on Mars Exploration Concepts, LPI Contribution # 1062*, July 2000, https://rotorcraft.arc.nasa.gov/Publications/files/Young2000Mars_Workshop.pdf.

[4] Lorenz, R. D., Turtle, E. P., Barnes, J. W., Trainer, M. G., Adams, D. S., Hibbard, K. E., Sheldon, C. Z., Zacny, K., Peplowski, P. N., Lawrence, D. J., et al., "Dragonfly: A Rotorcraft Lander Concept for Scientific Exploration at Titan," *Johns Hopkins APL Technical Digest*, Vol. 34, No. 3, Oct. 2018, p. 14.

[5] "Sample Recovery Helicopters," NASA Mars Sample Return Mission, https://mars.nasa.gov/msr/spacecraft/sample-recovery-helicopters/ [retrieved 25 April 2024].

[6] Johnson, W., "Mars Science Helicopter Conceptual Design," NASA TM-2020-220485, https://ntrs.nasa.gov/citations/20200002139 [retrieved 25 April 2024].

[7] Stoll, A., and Mikic, G., "Transition Performance of Tilt Propeller Aircraft," *78th VFS Annual Forum*, May 2022, https://www.jobyaviation.com/publications/.

[8] Koning, W., Johnson, W., and Grip, H., "Improved Mars Helicopter Aerodynamic Rotor Model for Comprehensive Analyses," *AIAA Journal*, Vol. 57, No. 9, Sept. 2019, pp. 3969–3979. https://doi.org/10.2514/1.J058045

[9] "Mars Helicopter," NASA Ingenuity Mars Helicopter Tech Demonstrator, https://mars.nasa.gov/technology/helicopter/# [retrieved 25 April 2024].

[10] Russell, C., Jung, J., Willink, G., and Glasner, B., "Wind Tunnel and Hover Performance Test Results for Multicopter UAS Vehicles," *72nd AHS Annual Forum*, May 2016, https://rotorcraft.arc.nasa.gov/Publications/files/72-2016-374.pdf.

[11] Gregory, D., Cornelius, J., Waltermire, S., Loob, C., and Schatzman, N., "Acoustic Testing of Five Multicopter UAS in the U.S. Army 7- by 10-Foot Wind Tunnel," NASA TM-2018-219894, May 2018, https://rotorcraft.arc.nasa.gov/Publications/files/Schatzman_TM_2018_219894_Final.pdf.

[12] Yoon, S., Lee, H., and Pulliam, T., "Computational Analysis of Multi-Rotor Flows," AIAA Paper 2016-0812, Jan. 2016. https://doi.org/10.2514/6.2016-0812

[13] Yoon, S., Chan, W., and Pulliam, T., "Computations of Torque-Balanced Coaxial Rotor Flows," AIAA Paper 2017-0052, Jan. 2017. https://doi.org/10.2514/6.2017-0052

[14] Yoon, S., Diaz, P., Boyd, D., Chan, W., and Theodore, C., "Computational Aerodynamic Modeling of Small Quadcopter Vehicles," *AHS 73rd Annual Forum*, May 2017, https://rotorcraft.arc.nasa.gov/Publications/files/73_2017_0015.pdf.

[15] Xu, H., and Ye, Z., "Coaxial Rotor Helicopter in Hover Based on Unstructured Dynamic Overset Grids," *Journal of Aircraft*, Vol. 47, No. 5, 2010, pp. 1820–1824. https://doi.org/10.2514/1.C031079

[16] Singh, P., and Friedmann, P., "Application of Vortex Methods to Coaxial Rotor Wake and Load Calculations in Hover," *Journal of Aircraft*, Vol. 55, No. 1, 2018, pp. 373–381. https://doi.org/10.2514/1.C034520

[17] Ahaus, L., Makinen, S., Meadowcroft, T., Tadghighi, H., Sankar, L., and Baeder, J., "Assessment of CFD/CSD Analytical Tools for Improved Rotor Loads," *AHS 71st Annual Forum*, May 2015, https://vtol.org/store/product/assessment-of-cfdcsd-analytical-tools-for-improved-rotor-loads-10136.cfm.

[18] Ho, J., Yeo, H., and Bhagwat, M., "Validation of Rotorcraft Comprehensive Analysis Performance Predictions for Coaxial Rotors in Hover," *Journal of the American Helicopter Society*, Vol. 62, No. 2, 2017, pp. 1–3. https://doi.org/10.4050/JAHS.62.022005

[19] Raissi, M., Perdikaris, P., and Karniadakis, G., "Machine Learning of Linear Differential Equations Using Gaussian Process," *Journal of Computational Physics*, Vol. 348, Nov. 2017, pp. 683–693. https://doi.org/10.1016/j.jcp.2017.07.050

[20] Raissi, M., Perdikaris, P., and Karniadakis, G., "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational Physics*, Vol. 378, Feb. 2019, pp. 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

[21] Wu, J., Xiao, H., and Paterson, E., "Physics-Informed Machine Learning Approach for Augmenting Turbulence Models: A Comprehensive Framework," *Physical Review Fluids*, Vol. 3, July 2018, Paper 074602. https://doi.org/10.48550/arXiv.1701.07102

[22] Liu, L., Liu, S., Xie, H., Xiong, F., Yu, T., Xiao, M., Liu, L., and Yong, H., "Discontinuity Computing Using Physics-Informed Neural Network," arXiv:2206.03864v2, Aug. 2022. https://doi.org/10.2139/ssrn.4224074

[23] Wang, H., Liu, Y., and Wang, S., "Dense Velocity Reconstruction from Particle Image Velocimetry/Particle Tracking Velocimetry Using a Physics-Informed Neural Network," *Physics of Fluids*, Vol. 34, No. 1, Jan. 2022, Paper 017116. https://doi.org/10.1063/5.0078143

[24] Karniadakis, G., Kevrekidis, I., Perdikaris, P., Wang, S., and Yang, L., "Physics-Informed Machine Learning," *Nature Review Physics*, Vol. 3, No. 6, 2021, pp. 422–440. https://doi.org/10.1038/s42254-021-00314-5

[25] Martinez, D., Brewer, W., Strelzoff, A., Wilson, A., and Wade, D., "Rotorcraft Virtual Sensors via Deep Regression," *Journal of Parallel and Distributed Computing*, Vol. 135, Jan. 2020, pp. 114–126. https://doi.org/10.1016/j.jpdc.2019.08.008

[26] LeCun, Y., Bengio, Y., and Hinton, G., "Deep Learning," *Nature*, Vol. 521, No. 7553, 2015, pp. 436–444.

[27] Brewer, W., Martinez, D., Boyer, M., Jude, D., Wissink, A., Parsons, B., Yin, J., and Anantharaj, V., "Production Deployment of Machine-Learned Rotorcraft Surrogate Models on HPC," *IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments*, 2021.

[28] Martinez, D., Siaraman, J., Brewer, W., Rivera, P., and Jude, D., "Machine Learning Based Aerodynamic Models for Rotor Blades," *VFS Transformative Vertical Flight Meeting*, 2020.

[29] Martinez, D., Jude, D., Sitaraman, J., Brewer, W., and Wissink, A., "ROAM-ML: A Reduced Order Aerodynamic Module Augmented with Neural Network Digital Surrogates," *AIAA SciTech Forum*, AIAA Paper 2022-1248, Jan. 2022.

[30] Chatterjee, T., Essien, A., Ganguli, R., and Friswell, M., "The Stochastic Aeroelastic Response Analysis of Helicopter Rotors Using Deep and Shallow Machine Learning," *Neural Computing and Applications*, Vol. 33, Dec. 2021, pp. 16,809–16,828. https://doi.org/10.1007/s00521-021-06288-w

[31] Lu, Y., Su, T., Chen, R., Li, P., and Wang, Y., "A Method for Optimizing the Aerodynamic Layout of a Helicopter that Reduces the Effects of Aerodynamic Interaction," *Aerospace Science and Technology*, Vol. 88, May 2019, pp. 73–83. https://doi.org/10.1016/j.ast.2019.03.005

[32] Schuet, S., Malpica, C., and Aires, J., "A Gaussian Process Enhancement to Linear Parameter Varying Models," *AIAA Aviation Forum*, AIAA Paper 2021-3006, Aug. 2021. https://doi.org/10.2514/6.2021-3006

[33] Brunton, S., Noack, B., and Koumoutsakos, P., "Machine Learning for Fluid Mechanics," *Annual Review of Fluid Mechanics*, Vol. 52, Jan. 2020, pp. 477–508.

[34] Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V., "Integrating Physics-Based Modeling with Machine Learning: A Survey," arXiv preprint arXiv:2003.049191.1, 2020, pp. 1–34.

[35] Guo, X., Li, W., and Iorio, F., "Convolutional Neural Networks for Steady Flow Approximation," *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 481–490. https://doi.org/10.1145/2939672.2939738

[36] Ghatnagar, S., Afshar, Y., Shaowu, P., Duraisamy, K., and Kaushik, S., "Prediction of Aerodynamic Flow Fields Using Convolutional Neural Networks," *Computational Mechanics*, Vol. 64, Jan. 2019, pp. 525–545.

[37] Allen, L., Lim, J., Haehnel, R., and Dettwiller, I., "Rotor Blade Design Framework for Airfoil Shape Optimization with Performance Considerations," *AIAA SciTech Forum*, AIAA Paper 2021-0068, Jan. 2021. https://doi.org/10.2514/6.2021-0068

[38] Stoll, A., and Mikic, G., "Transition Performance of Tilt Propeller Aircraft," *VFS Forum 78*, May 2022.

[39] Cornelius, J., Kinzel, M., and Schmitz, S., "Efficient Computational Fluid Dynamics Approach for Coaxial Rotor Simulations in Hover,"

Journal of Aircraft, Vol. 58, No. 1, 2021, pp. 197–202. https://doi.org/10.2514/1.C036037

[40] *RotCFD: Rotor Computational Fluid Dynamics Integrated Design Environment, Software Package*, Ver. 0.9.15 Build 402, Sukra Helitek, Ames, IA, 2020, http://sukra-helitek.com/.

[41] Rajagopalan, G., Baskaran, V., Hollingsworth, A., Lestari, A., Garrick, D., Solis, E., and Hagerty, B., "RotCFD—A Tool for Aerodynamic Interference of Rotors: Validation and Capabilities," *AHS Future Vertical Lift Aircraft Design Conference*, Jan. 2012, https://rotorcraft.arc.nasa.gov/Publications/files/A-5-D_rajagopalan.pdf.

[42] Conley, S., Russell, C., Kallstrom, K., Koning, W., and Romander, E., "Comparing RotCFD Predictions of the Multirotor Test Bed with Experimental Results," *VFS 76th Annual Forum*, Oct. 2020, https://rotorcraft.arc.nasa.gov/Publications/files/1429_Conley_070720.pdf.

[43] Rajagopalan, G., Thistle, J., and Polzin, W., "The Potential of GPU Computing for Design in RotCFD," *AHS Technical Meeting on Aeromechanics Design for Transformative Vertical Lift*, Jan. 2018, https://rotorcraft.arc.nasa.gov/Publications/files/Rajagopalan_2018_TechMx.pdf.

[44] Mathur, S. R., and Murthy, J. Y., "A Pressure-Based Method for Unstructured Meshes," *Journal of Numerical Heat Transfer*, Vol. 31, No. 2, 1997, pp. 195–215. https://doi.org/10.1080/10407799708915105

[45] Cornelius, J., and Schmitz, S., "Rotor Performance Predictions for UAM—Single vs Coaxial Rigid Rotors," *VFS Aeromechanics for Advanced Vertical Flight Technical Meeting*, Jan. 2022, https://rotorcraft.arc.nasa.gov/Publications/files/Jason_Cornelius_Schmitz_12-Jan-22.pdf

[46] Cornelius, J., and Schmitz, S., "Massive Graphical Processing Unit Parallelization for Multirotor Computational Fluid Dynamics," *Journal of Aircraft*, Vol. 60, No. 6, Nov. 2023, pp. 2010–2016. https://doi.org/10.2514/1.C037356

[47] "Rotors for Mission to Titan Tested at Langley's Transonic Dynamics Tunnel," NASA Langley Research Center, Dec. 2022, https://www.nasa.gov/feature/langley/rotors-for-mission-to-titan-tested-at-langley-s-transonic-dynamics-tunnel.

[48] Lorenz, R. D., Turtle, E. P., Barnes, J. W., Trainer, M. G., Adams, D. S., Hibbard, K. E., Sheldon, C. Z., Zacny, K., Peplowski, P. N., Lawrence, D. J., et al., "Dragonfly: A Rotorcraft Lander Concept for Scientific Exploration at Titan," *Johns Hopkins APL Technical Digest* Vol. 35, No. 3, 2018, pp. 374–387, https://dragonfly.jhuapl.edu/News-and-Resources/docs/34_03-Lorenz.pdf.

[49] "Regression Learner App," R2023b, MathWorks Help Center, 2023, https://www.mathworks.com/help/stats/regression-learner-app.html.