# Hybrid Mechanistic + Neural Model of Laboratory Helicopter

Christopher Rackauckas[1] Roshan Sharma, Bernt Lie[2]

[1]Massachusetts Institute of Technology, USA, me@ChrisRackauckas.com
[2]University of South-Eastern Norway, Norway, {Roshan.Sharma,Bernt.Lie}@usn.no

## Abstract

There is a growing interest in data-driven models of dynamic systems developed from "Big Data". Data-driven models have some limitations, e.g., without data, there is no model. *Hybrid* models consisting of relatively simple mechanistic models in tandem with data-driven models offer a compromise: physics understanding and design/synthesis of control structure prior to building the system, and improved model as data becomes available.

Here, we studied a strategy to develop hybrid models based on a laboratory helicopter case study. We presented the system, with a model based on Lagrangian mechanics. Torques were assumed linear: actuator torque in voltages, and friction torque in angular velocities. Nominal model parameters were taken from a laboratory set-up. Experimental data from a laboratory process was presented; the model gave poor model fit with nominal parameters. Optimal model parameters were found based on a ballistic fit measure, and gave acceptable fit for pitch angle measurements, but poor fit for yaw angle measurements. Next, the assumption of linear controller torque was relaxed by adding a feed forward neural network block within the continuous time dynamic model. Upon training, this still gave imperfect, but considerably better prediction of the yaw angle. In a final, "model discovery" step, physically relevant alternatives to the neural network were explored.

The key idea of the paper was to illustrate a strategy for hybrid models. The current model still has some structural imperfections; these should be resolved before model validation becomes meaningful.

*Keywords: neural differential equations, mechanistic model, hybrid data-driven and mechanistic model, helicopter model, control relevant model*

## 1 Introduction

### 1.1 Background

Two main types of models for describing dynamic systems are (i) data-driven models (regression models, machine learning, empirical models) found by tuning parameters in some abstract mathematical structure, and (ii) mechanistic models (physics based models) which utilize balance laws based on the mechanisms that drive the change in the system coupled with semi-empiric phenomenological models (transport laws, reaction rates, etc.). These two model types have their own strengths and weaknesses. Today there is an increasing focus on the possibility of developing *data-driven* dynamic models from "Big data".

Data-driven models put little emphasis on understanding the system, can easily be automated in computers, but require large amounts of data and do not extrapolate well.

Mechanistic models require good understanding of the physics of the system, impose limitations in model generalization, and are more complex to automate in computers. On the other hand, mechanistic models require less data, and extrapolate better.

Ideally, one would combine the best of both model types. This would imply to use simplified mechanistic models prior to the availability of data, train regression models to describe uncertain phenomenological relations and missing dynamics in the mechanistic model, and this way gradually improve the model description.

### 1.2 Previous work

Feed forward neural networks constitute a modern example of nonlinear regression models, i.e., a static mapping from an input signal/feature vector to an output signal which can be fitted to experimental data by modifying weight/bias parameters. Traditional use of neural networks for dynamic models are based on feed forward neural networks with auto-regressive/delayed outputs and moving average/delayed inputs. Alternatively, recurrent neural networks include internal feedback paths in layers. Both approaches are based on discrete time data/models.

Since feed forward neural networks describe a nonlinear mapping, suppose that both inputs and states of a system are known over some time horizon, together with the derivative of the states. Then one can treat the inputs and states as inputs to the neural network, and the derivatives of the states as outputs from the neural network. By tuning neural network parameters, this enables fitting the neural network to the vector field of the differential equation, leading to Neural Differential Equations (Farrell and Polycarpou, 2006; Chen et al., 2018). If all states and state derivatives are known, fitting a neural network to the vector field is straightforward. This may be the case in simulation studies for model reduction. However, for real systems with experimental data from a limited number of sensors, two problems are faced: (i) derivatives are not available and can at best be found by some smoothing/spline fitting, and (ii) derivatives can be approximated for sensor outputs, but not directly for states. This makes the problem more challenging.

Based on the ideas in (Chen et al., 2018), a set of packages for computer language Julia (Bezanson et al., 2017) are combined to fit neural differential equation models (Rackauckas et al., 2019, 2020) to experimental data, leading to differential equations that can be solved by standard differential equation solvers (Rackauckas and Nie, 2017). However, the packages aim higher: they allow for a very general mixing of mechanistic models and neural differential equation models in the same framework, with possibilities for the user to choose whether only parameters in the neural network model are tuned, or parameters in both the mechanistic model and the neural differential equation. The advantage of this approach is significant: it allows for the desirable possibility of extending mechanistic models with data-driven elements.

In this paper, we consider a mechanistic model of a laboratory helicopter at University of South-Eastern Norway; the model is used in a course on Model Predictive Control. A mechanistic dynamic model of the system can be developed as in (Gäfvert, 2001) using Lagrangian mechanics. Using sets of experimental data, a hybrid model consisting of the mechanistic model extended with neural network blocks is fitted using Julia package DiffEqFlux.jl. This allows for developing an improved model compared to the purely mechanistic model, and serves as a starting point for challenges related to hybrid models.

### 1.3 Overview of paper

The paper is organized as follows. Section 2 presents the laboratory helicopter, and a mechanistic model developed from Lagrangian mechanics and nominal model parameters. In Section 3, experimental data are presented, and it is shown that poor knowledge of initial values for the model with the chosen nominal parameters gives low prediction accuracy. Next, a model fitting measure is proposed (loss function), and parameters and initial values are adjusted to minimize the loss function. The model is still imperfect. In Section 4, the torque model used in the mechanistic model is replaced by a neural network block from the controller inputs (voltages) to the controller torques. Finally, in Section 5, some conclusions are drawn. Nominal model parameters and operating conditions are provided in Appendix A.

## 2 Helicopter mechanistic model

### 2.1 Laboratory helicopter

The laboratory helicopter used as a case study, is shown in Fig. 1.

A video file is available detailing the operation of the helicopter, and some elements which makes it complicated to achieve a perfect mechanistic model of the helicopter.[1]

### 2.2 Geometry of helicopter

Consider the helicopter in Fig. 2, with upward-pointing $z$ axis. *Origo* of the body-fixed coordinate system is in the longitudinal inertial axis of the helicopter. The laboratory helicopter is hinged to the ground; in Fig. 2, the pivot point is located in the body-fixed origo.

If the helicopter body is elevated a distance $h$ above the pivot point, differential mass $\mathrm{d}m(r)$ at position $r$ along the longitudinal axis of the helicopter is given by coordinates $z_r$ and $\xi_r$,

$$\begin{aligned} z_r &= r\sin\theta + h\cos\theta \\ \xi_r &= r\cos\theta - h\sin\theta. \end{aligned}$$

The angle of nose raise $\theta$ is termed the *pitch* angle. Seen from a birds-eye view, the helicopter can also rotate in the plane by the *yaw* angle $\psi$, Fig. 3.

The positions in the plane are

$$\begin{aligned} x_r &= \xi_r \cos\psi \\ y_r &= \xi_r \sin\psi. \end{aligned}$$

In summary, the position of mass $\mathrm{d}m(r)$ is given by Cartesian coordinates $(x_r, y_r, z_r)$

$$\begin{aligned} x_r &= (r\cos\theta - h\sin\theta)\cos\psi \\ y_r &= (r\cos\theta - h\sin\theta)\sin\psi \\ z_r &= r\sin\theta + h\cos\theta, \end{aligned}$$

where the generalized coordinates are $\mathsf{x} = (\theta, \psi)$.

### 2.3 Kinetic energy of helicopter

With linear velocities $v_x \triangleq \frac{\mathrm{d}x_r}{\mathrm{d}t}$, $v_y \triangleq \frac{\mathrm{d}y_r}{\mathrm{d}t}$, and $v_z = \frac{\mathrm{d}v_z}{\mathrm{d}t}$, and introducing angular velocities $\omega_\theta \triangleq \frac{\mathrm{d}\theta}{\mathrm{d}t}$ and $\omega_\psi \triangleq \frac{\mathrm{d}\psi}{\mathrm{d}t}$, the squared velocity $v_r^2 = v_x^2 + v_y^2 + v_z^2$ can be expressed by the generalized velocities $\mathsf{v} = (\omega_\theta, \omega_\psi)$ as

$$\begin{aligned} v_r^2 = {} & r^2\left(\omega_\theta^2 + \cos^2\theta\cdot\omega_\psi^2\right) + h^2\left(\omega_\theta^2 + \sin^2\theta\cdot\omega_\psi^2\right) \\ & - 2rh\sin\theta\cos\theta\cdot\omega_\psi^2. \end{aligned}$$

Combining the rotation of the helicopter body around the pivot point with body center moment of inertia $J_{\mathrm{bc}}$ and mass center distance $r_{\mathrm{c}}$, and rotation around the shaft with moment of inertia $J_{\mathrm{s}}$ gives a total kinetic energy expressed as

$$K(\mathsf{x},\mathsf{v}) = \frac{1}{2}\mathsf{v}^T \mathsf{M}(\mathsf{x})\,\mathsf{v}$$

where the mass matrix $\mathsf{M}(\mathsf{x})$ can be shown to be diagonal with

$$\begin{aligned} \mathsf{M}_{11}(\mathsf{x}) &= J_{\mathrm{bc}} + m r_{\mathrm{c}}^2 + m h^2 \\ \mathsf{M}_{22}(\mathsf{x}) &= J_{\mathrm{bc}}\cos^2\theta + m\left(r_{\mathrm{c}}\cos\theta - h\sin\theta\right)^2 + J_{\mathrm{s}}. \end{aligned}$$

### 2.4 Potential energy of helicopter

Setting the potential energy to zero in the pivot point, the potential energy of the helicopter becomes

$$\begin{aligned} & P = \int_m g z_r \mathrm{d}m(r) \\ & \Downarrow \\ & P = g\int_m \left(r\sin\theta + h\cos\theta\right)\mathrm{d}m(r), \end{aligned}$$

[1] https://web01.usn.no/~roshans/mpc/videos/Heli_deadband_effect.mp4

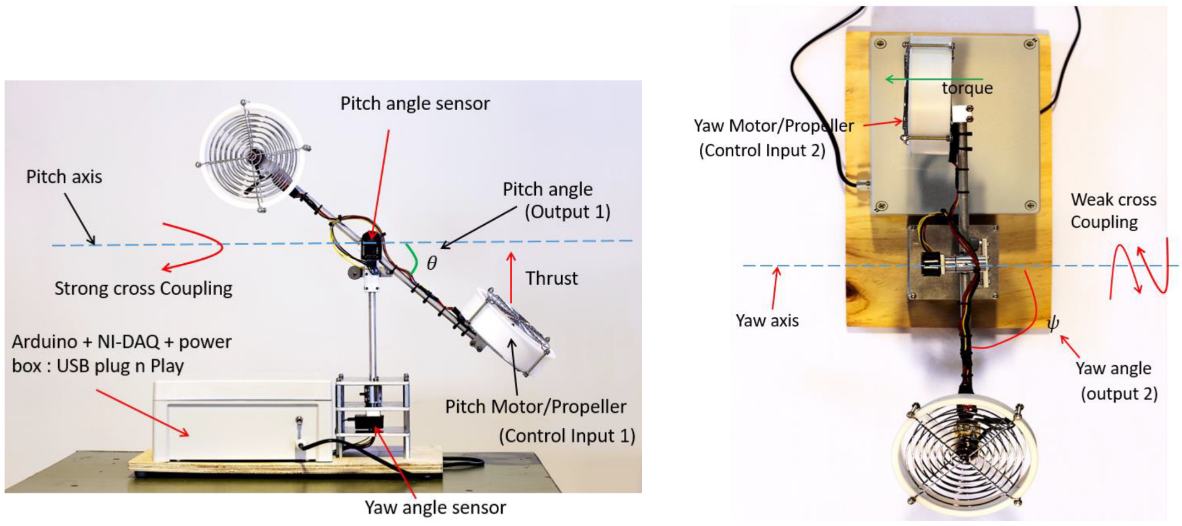

**Figure 1.** Laboratory helicopter, (Sharma, 2020).

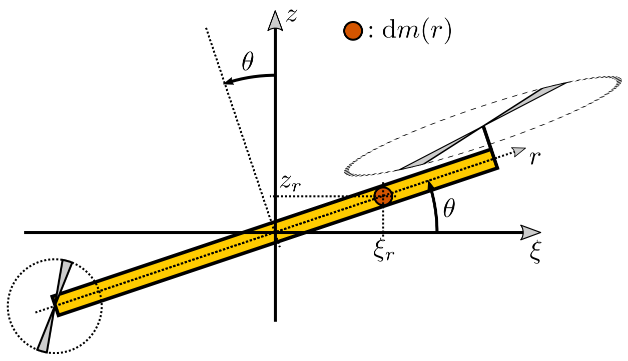

**Figure 2.** Helicopter in side profile, pitch angle $\theta$.

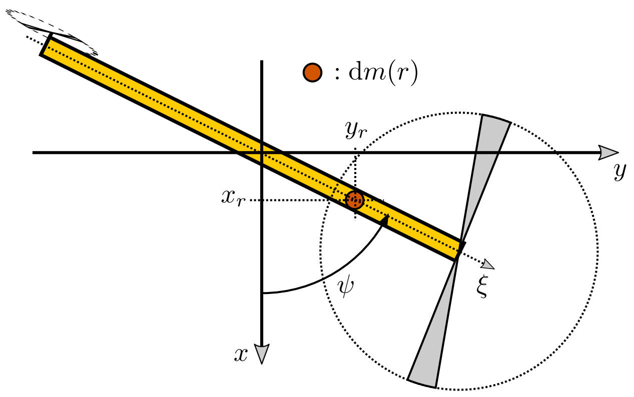

**Figure 3.** Helicopter in birds-eye view, yaw angle $\psi$.

which can be expressed as

$$P(\mathsf{x}) = mg\left(r_{\mathrm{c}}\sin\theta + h\cos\theta\right).$$

## 2.5 Helicopter torques

We need the generalized force $\mathsf{F} = \left(T_\theta, T_\psi\right)$, which is given by some phenomenological relation

$$\mathsf{F} = \mathsf{F}\left(\mathsf{x}, \mathsf{v}, u\right).$$

Here, $u = \left(u_\theta, u_\psi\right)$ contains the voltages $u_\theta$ and $u_\psi$ applied to the rotors. In mechanistic models, a proposal for $\mathsf{F}$ could be

$$\mathsf{F} = Ku - D\mathsf{v}$$

with a full motor gain matrix $K$ (Sharma, 2020)

$$K = \begin{pmatrix} K_{\theta,\theta} & -K_{\theta,\psi} \\ K_{\psi,\theta} & -K_{\psi,\psi} \end{pmatrix},$$

and a diagonal damping friction matrix $D$

$$D = \begin{pmatrix} D_\theta & 0 \\ 0 & D_\psi \end{pmatrix}.$$

In practice, this linear description may be too simple: the system may exhibit dead-band thus invalidating the term $Ku$. Presence of nonlinear friction such as stiction or quadratic terms in $\mathsf{v}$ invalidate the term $D\mathsf{v}$. Torsional effects would imply missing potential energy in $\mathsf{x}$. The system may even exhibit backlash, and thus make it necessary to introduce extra states $z$ with generalized force $\mathsf{F}(\mathsf{x}, \mathsf{v}, z, u)$ (Lichtsinder and Gutman, 2016, 2019).

Particular problems related to how the laboratory helicopter is constructed, are discussed in link `https://web01.usn.no/~roshans/mpc/videos/Heli_deadband_effect.mp4`: power from $u$ is transmitted to the helicopter via wires in the shaft, with twisting of these wires. This design may give rise to some dead-band, but it may perhaps also give rise to some resisting torsion spring torque.

### 2.6 DAE formulation of model

We now introduce the *Lagrangian* $\mathsf{L}$ defined as

$$\mathsf{L}(\mathsf{x},\mathsf{v}) \triangleq K(\mathsf{x},\mathsf{v}) - P(\mathsf{x}).$$

Defining the generalized momentum $\mathsf{p} \triangleq \partial \mathsf{L}/\partial \mathsf{v}$, a DAE formulation of the Euler-Lagrange equation can be posed as

$$\begin{aligned}\frac{d\mathsf{x}}{dt} &= \mathsf{v}\\ \frac{d\mathsf{p}}{dt} &= \frac{\partial \mathsf{L}}{\partial \mathsf{x}} + \mathsf{F}\\ \mathsf{p} &= \mathsf{M}(\mathsf{x})\,\mathsf{v}.\end{aligned}$$

Here, the momentum $\mathsf{p}$ contains angular momenta, $\mathsf{p} = (\mathsf{p}_\theta, \mathsf{p}_\psi)$, while term $\frac{\partial \mathsf{L}}{\partial \mathsf{x}} = \left(\frac{\partial \mathsf{L}}{\partial \theta}, \frac{\partial \mathsf{L}}{\partial \psi}\right)$.

### 2.7 ODE formulation of model

If we eliminate the momentum, we can rewrite the model as implicit ODEs,

$$\begin{aligned}\frac{d\mathsf{x}}{dt} &= \mathsf{v}\\ \mathsf{M}(\mathsf{x})\frac{d\mathsf{v}}{dt} &= -\frac{\partial \mathsf{p}}{\partial \mathsf{x}}\mathsf{v} + \frac{\partial \mathsf{L}}{\partial \mathsf{x}} + \mathsf{F}.\end{aligned}$$

The ODE form of the model can then be expressed as

$$\begin{aligned}\mathsf{M}_{11}\frac{d\omega_\theta}{dt} = &-\omega_\psi^2\left(\left(J_{\mathrm{bc}} + m\left(r_{\mathrm{c}}^2 - h^2\right)\right)\cos\theta\sin\theta\right.\\ &\left.-r_{\mathrm{c}}h\left(\cos^2\theta - \sin^2\theta\right)\right)\\ &-mg\left(r_{\mathrm{c}}\cos\theta - h\sin\theta\right) + T_\theta\end{aligned}$$

$$\begin{aligned}\mathsf{M}_{22}\frac{d\omega_\psi}{dt} = &\,2\omega_\psi\omega_\theta\left(\left(J_{\mathrm{bc}} + m\left(r_{\mathrm{c}}^2 - h^2\right)\right)\cos\theta\sin\theta\right.\\ &\left.+mr_{\mathrm{c}}h\left(\cos^2\theta - \sin^2\theta\right)\right) + T_\psi\end{aligned}$$

with the trivial additional equations

$$\begin{aligned}\frac{d\theta}{dt} &= \omega_\theta\\ \frac{d\psi}{dt} &= \omega_\psi.\end{aligned}$$

In (Sharma, 2020), a simplified model is used with $h \equiv 0$, neglecting the shaft moment of inertia $J_{\mathrm{s}}$, etc.

## 3 Preliminary model fitting

### 3.1 Experimental data

Voltages for pitch motor $u_\theta$ and yaw motor $u_\psi$ used in one particular experiment[2] with $N_{\mathrm{d}} = 13\,100$ datapoints sampled at $\Delta t = 0.01$ are displayed in Fig. 4; notice the different ordinate axes on each side of the plot.

Observed pitch angle $\theta$ and yaw angle $\psi$ from the particular experiment are shown in Fig. 5. Again, notice the use of different ordinate axes on each side of the plot.

Pitch angle $\theta$ and yaw angle $\psi$ from experiments with nominal model parameters are displayed in Fig. 6.

[2]https://web01.usn.no/~roshans/mpc/videos/Heli_deadband_effect.mp4

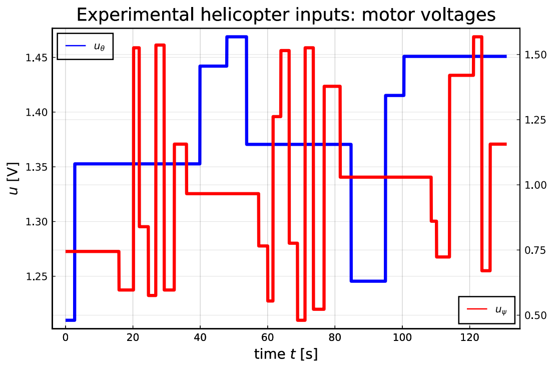

**Figure 4.** Pitch motor voltage $u_\theta$ and yaw motor $u_\psi$ experiments.

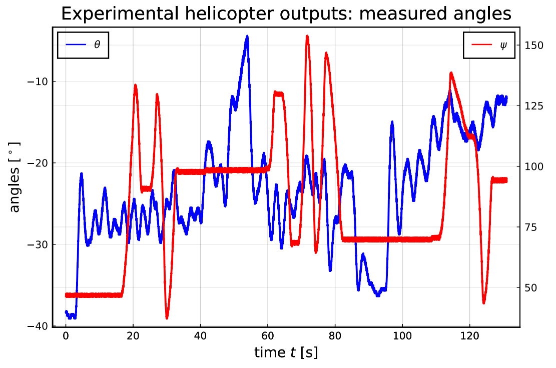

**Figure 5.** Pitch angle $\theta$ and yaw angle $\psi$ from experiments.

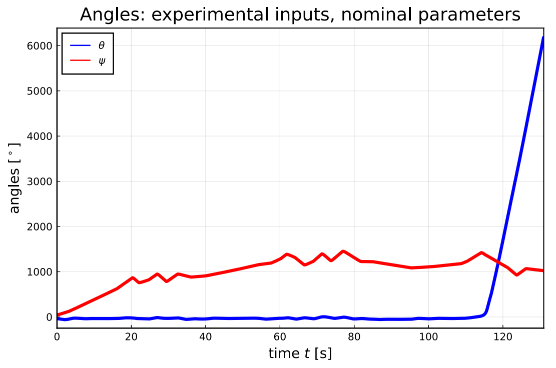

**Figure 6.** Model pitch angle $\theta$ and yaw angle $\psi$ based on nominal model parameters, using experimental inputs.

**Table 1.** Fitted parameters for laboratory helicopter.

| Parameter | Nominal value $p_0$ | Fitted value $p^*$ |
|---|---|---|
| $r_c$ | $1.5\,\mathrm{cm}$ | $2.2\,\mathrm{cm}$ |
| $h$ | $0\,\mathrm{cm}$ | $0.1\,\mathrm{cm}$ |
| $m$ | $0.5\,\mathrm{kg}$ | $0.72\,\mathrm{kg}$ |
| $J_{bc}$ | $0.015\,\mathrm{kg\,m^2}$ | $0.01\,\mathrm{kg\,m^2}$ |
| $J_s$ | $0.005\,\mathrm{kg\,m^2}$ | $0.0017\,\mathrm{kg\,m^2}$ |
| $K_{\theta,\theta}$ | $0.055\,\mathrm{Nm/V}$ | $0.105\,\mathrm{Nm/V}$ |
| $K_{\theta,\psi}$ | $0.005\,\mathrm{Nm/V}$ | $0.0017\,\mathrm{Nm/V}$ |
| $K_{\psi,\theta}$ | $0.15\,\mathrm{Nm/V}$ | $0.03\,\mathrm{Nm/V}$ |
| $K_{\psi,\psi}$ | $0.20\,\mathrm{Nm/V}$ | $0.043\,\mathrm{Nm/V}$ |
| $D_\theta$ | $0.01\,\mathrm{Nm/(rad/s)}$ | $0.014\,\mathrm{Nm/(rad/s)}$ |
| $D_\psi$ | $0.08\,\mathrm{Nm/(rad/s)}$ | $0.11\,\mathrm{Nm/(rad/s)}$ |

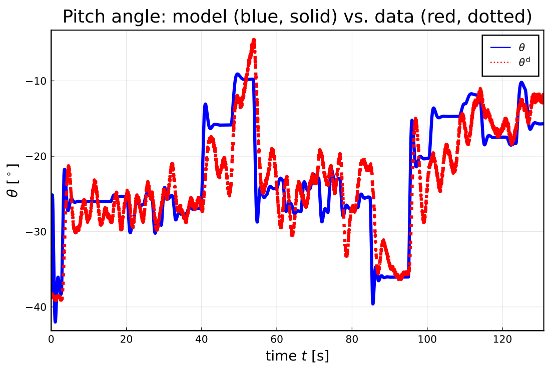

**Figure 7.** Pitch angle $\theta$: comparing model angle with optimized parameters (blue, solid) and measured experimental angle.

## 3.2 Preliminary model fitting

We now tune model parameters $p$ to minimize the *loss* function $\ell$

$$\begin{aligned}\ell(p) &= \sqrt{\frac{1}{n_y \cdot N_d}\sum_{k=1}^{N_d}\left[\left(\theta_k-\theta_k^d\right)^2+\left(\psi_k-\psi_k^d\right)^2\right]}\\ &= \frac{\left\|y-y^d\right\|_2}{\sqrt{n_y\cdot N_d}} = \sigma_\varepsilon \end{aligned} \qquad (1)$$

where $\theta_k$ is the model pitch angle and $\psi_k$ is the model yaw angle, while $\theta_k^d$ and $\psi_k^d$ are logged data of the same angles, and $n_y = \dim y_k = 2$. It follows that $\ell(p)$ is the average standard deviation $\sigma_\varepsilon$ in each measured angle. The model angles vary with the model parameters $p$; $\theta_k = \theta_k(p)$ and $\psi_k = \psi_k(p)$. In Eq. 1, $\theta_k$ and $\psi_k$ are found from *ballistic* simulation/single shooting of the model from the initial conditions, and not from multiple shooting as in prediction error methods.

In the model fitting, the 4 initial states and 11 model parameters are tuned to minimize $\ell(p)$ using method `bboptim` with default settings in package `BlackBoxOptim.jl` for Julia. Table 1 shows the fitted values of model parameters.

Pitch angle $\theta$ from the model and from experimental data with optimized model parameters are given in Fig. 7.

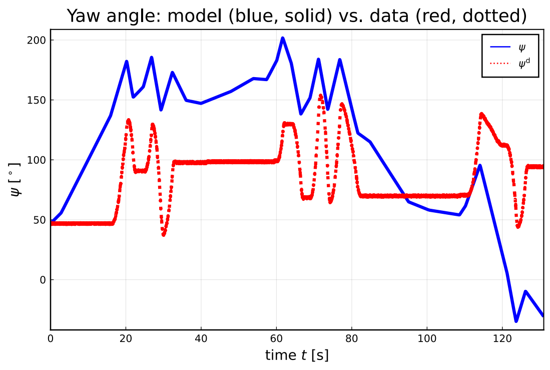

**Figure 8.** Yaw angle $\psi$: comparing model angle with optimized parameters (blue, solid) and measured experimental angle.

Yaw angle $\psi$ from model and from experimental data with optimized model parameters are displayed in Fig. 8.

Because a mechanistic model with given parameter set is used, and because poor model fit is achieved for the yaw angle, it is not relevant to consider model validation: the model structure is too limited to allow for good fit. The optimal loss function is $\ell(p^*) \approx 92$, with the error mainly in the yaw angle.

## 4 Hybrid model

### 4.1 Neural torque extensions

In Section 2.5, we considered a helicopter torque given as $\mathsf{F} = Ku - D\mathsf{v}$. We now modify this torque expression to

$$\mathsf{F} = Ku - D\mathsf{v} + \mathrm{FNN}(u;p) \qquad (2)$$

where we keep the optimal parameter values in Table 1, but re-fit the model by tuning parameters $p$ in $\mathrm{FNN}(\cdot)$ consisting of weights and biases in a two layer Feed forward Neural Net with 2 inputs, 16 nodes in the hidden layer and a $\tanh^{-1}(\cdot)$ activation function, and 2 outputs in the linear output layer. The tuning of the neural net is carried out in two phases: in the first phase, gradient search is done using an ADAM method of Julia package DiffEqFlux.jl, using the global loss function in Eq. 1 of Section 3.2. In a second, polishing step, we use a quasi-Newton method BFGS method to fine tune the parameters of the neural net.

After fitting the neural network, the pitch angles are as in Fig. 9 — which should be compared to Fig. 7.

The model fitting of the yaw angle is as in Fig. 10 — which should be compared to Fig. 8.

The resulting loss function after fitting the 16 node FNN is $\ell(p^*) \approx 9.2$ after phase 2 with the quasi-Newton BFGS method ($\ell(p^*) \approx 55.4$ after the ADAMS gradient search of phase 1). Using a 32 node FNN leads to $\ell(p^*) \approx 9.3$ after phase 2 ($\ell(p^*) \approx 35$ after phase 1); thus going to 32 nodes does not improve the fit.

We may also modify the torque expression to include a neural network which both depends on input voltage $u$ and

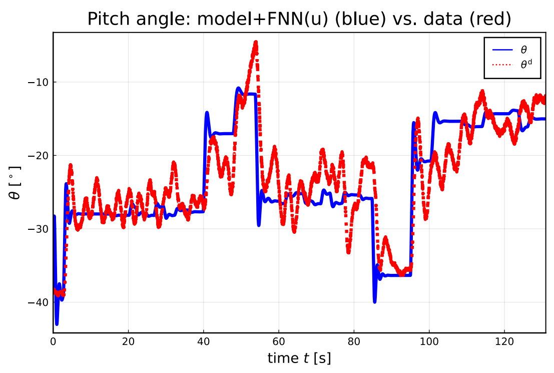

**Figure 9.** Pitch angle with torque modeled as in Eq. 2 with additive term $\mathrm{FNN}(u;p)$.

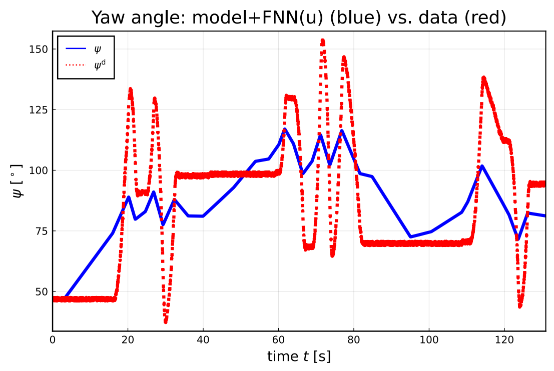

**Figure 10.** Yaw angle with torque modeled as in Eq. 2 with additive term $\mathrm{FNN}(u;p)$..

states x, v:

$$\mathsf{F} = Ku - D\mathsf{v} + \mathrm{FNN}(u,\mathsf{x},\mathsf{v};p)\,. \tag{3}$$

Now we have a neural network with 6 inputs ($u$, x, v) and two outputs x, with a hidden layer with 16 nodes. The resulting loss function is $\ell(p^*) \approx 11.1$ after phase 2 ($\ell(p^*) \approx 14.1$ after phase 1). With the limited available data, this extension doesn't improve the model fit: the pitch angle prediction is somewhat poorer, while the yaw angle prediction is slightly better.

## 4.2 Equation discovery

When finding the model augmented with a neural network addition to the torque F, we know the experimental input $u$ and compute the resulting solution $x$. We can then post process the solution and compute $\mathrm{FNN}(u;p)$. This gives us the time series of $u$ and $\mathrm{FNN}(u;p)$, which we will use for "equation discovery". The idea is simply to propose a regression model between $u$ and $\mathrm{FNN}(u;p)$, and then fit this regression model. In such a regression model, we will postulate a large number of basis function which may have a physical origin. As an *example*, we could postulate a regression model of form

$$\mathrm{FNN}(u;p) = c_1 + c_2 u_\theta + c_3 u_\psi + c_4 u_\theta^2 + c_5 u_\psi^2 + c_6 u_\theta u_\psi \cdots \tag{4}$$

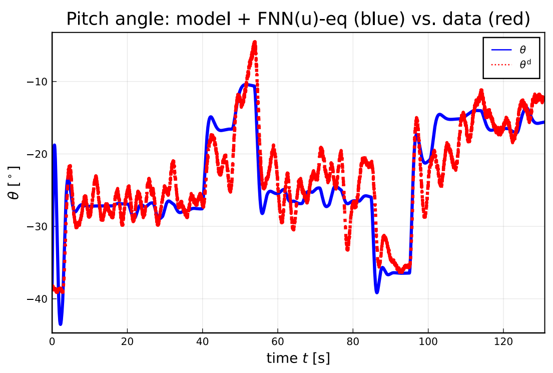

**Figure 11.** Pitch angle with regression approximation of $\mathrm{FNN}(u;p)$.

In this case, we have a linear regression model that can easily be fitted to the data set $(u, \mathrm{FNN}(u;p))$. Some of the data set can be used for training the model, and some of it can be used for validation. This way, it is possible to "discover" which of the basis functions/equations that are relevant to describe the fitted neural network part of the torque.

To automate this process, package `DataDrivenDiffEq.jl` is used. A set of basis functions are proposed in line with the regression equation in Eq. 4 using the `Basis()` constructor of package `DataDrivenDiffEq.jl`. Next, the optimizer `SINDy` (Sparse Identification of Nonlinear Dynamics) of this package is used to "discover" the relevant equations that explain the neural network. If we denote by $\mathrm{FNN}(\cdot)$ the output data from the neural network, by $\Phi(u_\theta, u_\psi)$ denote the regressor data (i.e., the RHS terms in Eq. 4), and by $p = (c_1, c_2, \ldots)$ denote the vector of unknown parameters, we can pose a multi-objective problem combined by weight $\lambda$ as

$$p^* = \arg\min_p \left( \left\| \mathrm{FNN}(\cdot) - \Phi(u_\theta, u_\psi)\, p \right\|_2 + \lambda \left\| p \right\|_1 \right). \tag{5}$$

Here, the use of 1-norm in the regularization term, $\|p\|_1$, tends to favor sparse solutions, and the weight $\lambda$ is varied to give a number of alternative models. The model choice is then based on the Akaike Information Criterion. Finally, this set of discovered equations can replace the neural network in the expression for F.

We find that

$$\mathrm{FNN}(u;p)_1 \approx -4.37 \cdot 10^{-4} \cos(u_\psi) + 4.02 \cdot 10^{-4} \sin(u_\psi)$$
$$\mathrm{FNN}(u;p)_2 \approx -1.35 \cdot 10^{-2} \cos(u_\psi) + 7.74 \cdot 10^{-3} u_\psi^2.$$

By replacing the neural network with the approximate, "discovered" equations, the model fit is as in Figs. 11 and 12.

When comparing Figs. 9 and 10 to Figs. 11 and 12, observe that the pitch approximation is almost identical, while the yaw approximation is good — but a little bit off that from $\mathrm{FNN}(u;p)$.

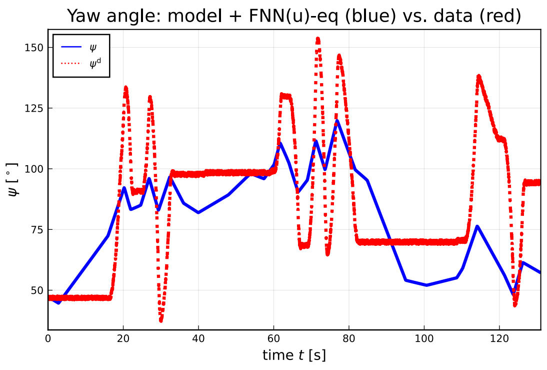

**Figure 12.** Yaw angle with regression approximation of $\mathrm{FNN}(u;p)$.

## 5 Conclusions and Future work

The key problem of fitting mechanistic models to experimental data has been considered, based on a simple model of a laboratory helicopter and experimental data from a lab rig. As in all engineering practice, the model structure of the original mechanistic model limits how well it is possible to fit the model to real data. Still, a mechanistic model gives important information about the system and may often be used for control design with some success, even when a perfect model fit is not possible. The reason why control design works for imperfect models is that feedback to some degree corrects for the effect of model error.

It is of interest to *improve* on a mechanistic model as more data becomes available. Thus, we do not want to replace the mechanistic model by a completely data driven model, but instead augment the mechanistic model with data driven elements as more information becomes available. An obvious way to do this is to add regression type blocks in the mechanistic model, both in phenomenological laws and possibly also in adding more states to the system with regression type vector fields. One possible data driven model structure is a neural network. The challenge with including regression blocks in dynamic models is how to integrate these in an efficient way so that differential equation solvers can handle them, so that it is possible to take advantage of GPUs, and how to make accessible model gradients for training the network within the dynamic model.

In this paper, we have used packages within the DifferentialEquations.jl package and the Flux.jl package ecosystems of Julia to achieve this. The original, fitted mechanistic model is good for pitch angle predictions, but poor for yaw angle predictions. By adding a torque $\mathrm{FNN}(u;p)$ computed via a trained neural network, the yaw angle prediction improves considerably. Specifically, the loss function used for model fit is reduced from $\ell(p^*) = 92$ with the original model to $\ell(p^*) = 9.2$ when the neural network torque term with 16 internal nodes has been added. In our case of model and available data, it appears that there is no gain in increasing the number of internal nodes, or to

**Table 2.** Summary of loss function values. For Mechanistic model of F, a black box optimizer is used. For hybrid models of F, a gradient based phase 1 is used ($\ell(p^*)_1$) followed by a quasi-Newton phase ($\ell(p^*)_2$). The index on the neural networks indicate the number of nodes, 16 or 32.

| Case | $\ell(p^*)_1$ | $\ell(p^*)_2$ |
|---|---|---|
| Mechanistic | | 92 |
| Hybrid, $\mathrm{FNN}(u;p)_{16}$ | 55.4 | 9.2 |
| Hybrid, $\mathrm{FNN}(u;p)_{32}$ | 35 | 9.3 |
| Hybrid, $\mathrm{FNN}(u,\mathrm{x},\mathrm{v};p)_{16}$ | 14.1 | 11.1 |

include mapping $\mathrm{FNN}(u,\mathrm{x},\mathrm{v};p)$ from the states; this does not reduce the loss function value. The loss function values are summarized in Table 2.

With multiple candidate models, it is common to use model validation on independent data to choose model. The key reason for doing this is to avoid model overfitting/fitting noise. With the data we have available and the tested neural network mappings, the model fit is still relatively poor, so there is little sense in doing model validation at the current stage.

For practical implementation, a neural network is not the most convenient model, as it is rather complex with many parameters. Instead, we would like to replace the neural network by a regression approximation using more physically relevant basis functions. This is what the *Equation discovery* part of the Julia eco-system allows for. As seen, such equation discovery can lead to dramatically reduced model complexity compared to a neural network.

Overall, the tools offered in Julia give a first glimpse into what future modeling practice will look like: one starts by developing an efficient, and relatively simple mechanistic model for design/control synthesis/... As more data become available, the model can be improved by adding unstructured data driven terms (e.g., neural networks) which subsequently are given a physical interpretation by equation discovery.

Future work with the helicopter model will include experiments with more data, neural networks with added states, equation discovery with more extensive sets of basis functions, and testing how the improved model influences the controller performance.

## A Nominal parameters and operating conditions

Nominal parameters for the mechanistic model are given in Table 3.

Nominal operating conditions for simulations are chosen as in Table 4.

## References

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Sha. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 49(1):65–98, 2017. doi:10.1137/141000671.

**Table 3.** Parameters for laboratory helicopter.

| Parameter | Value | Description |
|---|---|---|
| $r_\mathrm{c}$ | $1.5\,\mathrm{cm}$ | Distance between pivot point and center of mass. |
| $h$ | $0\,\mathrm{cm}$ | Elevation of helicopter from pivot point. |
| $m$ | $0.5\,\mathrm{kg}$ | Mass of helicopter |
| $J_\mathrm{bc}$ | $0.015\,\mathrm{kg\,m^2}$ | Moment of Inertia of body about center of mass. |
| $J_\mathrm{s}$ | $0.005\,\mathrm{kg\,m^2}$ | Moment of Inertia of shaft. |
| $g$ | $9.81\,\mathrm{m/s^2}$ | Acceleration of gravity. |
| $K_{\theta,\theta}$ | $0.055\,\mathrm{Nm/V}$ | Torque constant on pitch coordinate from pitch motor. |
| $K_{\theta,\psi}$ | $0.005\,\mathrm{Nm/V}$ | Torque constant on pitch coordinate from yaw motor. |
| $K_{\psi,\theta}$ | $0.15\,\mathrm{Nm/V}$ | Torque constant on yaw coordinate from pitch motor. |
| $K_{\psi,\psi}$ | $0.20\,\mathrm{Nm/V}$ | Torque constant on yaw coordinate from yaw motor. |
| $D_\theta$ | $0.01\,\mathrm{Nm/(rad/s)}$ | Friction torque damping coefficient in pitch coordinate. |
| $D_\psi$ | $0.08\,\mathrm{Nm/(rad/s)}$ | Friction torque damping coefficient in yaw coordinate. |

**Table 4.** Initial operating conditions for laboratory helicopter.

| Quantity | Value | Description |
|---|---|---|
| $\theta\,(0)$ | $\frac{\pi}{4}\,\mathrm{rad}$ | Initial pitch angle. |
| $\psi\,(0)$ | $0\,\mathrm{rad}$ | Initial yaw angle. |
| $\omega_\theta\,(0)$ | $-0.1\,\mathrm{rad/s}$ | Initial pitch angular velocity. |
| $\omega_\psi\,(0)$ | $0.1\,\mathrm{rad/s}$ | Initial yaw angular velocity. |
| $u_\theta\,(t)$ | $0\,\mathrm{V}$ | Pitch motor voltage. |
| $u_\psi\,(t)$ | $0\,\mathrm{V}$ | Yaw motor voltage. |

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv CoRR*, abs/1806.07366, 2018. http://arxiv.org/abs/1806.07366.

Jay A. Farrell and Marios M. Polycarpou. *Adaptive Approximation Based Control. Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*. Wiley-Interscience, Hoboken, New Jersey, 2006.

G.M. Gäfvert. Modelling of the eth helicopter laboratory process. Technical Report TFRT-7596, Department of Automatic Control, Lund Institute of Technology (LTH), 2001.

Arkady Lichtsinder and Per-Olof Gutman. Closed-form sinusoidal-input describing functionfor the exact backlash model. *IFAC-PapersOnLine*, 49(18):422–427, 2016. doi:https://doi.org/10.1016/j.ifacol.2016.10.202.

Arkady Lichtsinder and Per-Olof Gutman. On the dual properties of friction andbacklash in servo control systems. *IFAC-PapersOnLine*, 52(16):340–345, 2019. doi:https://doi.org/10.1016/j.ifacol.2019.11.803.

Christopher Rackauckas and Qing Nie. DifferentialEquations.jl — A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *Journal of Open Research Software*, 5(15), 2017. doi:10.5334/jors.151.

Christopher Rackauckas, Mike Innes, Yingbo Ma, Jesse Bettencourt, Lyndon White, and Vaibhav Dixit. Diffeqflux.jl - A julia library for neural differential equations. *arXiv CoRR*, abs/1902.02376, 2019. http://arxiv.org/abs/1902.02376.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv CoRR*, abs/2001.04385, 2020. https://arxiv.org/abs/2001.04385.

Roshan Sharma. Two degrees of freedom (2-dof) helicopter. Laboratory problem in course iia 4117 model predictive control, University of South-Eastern Norway, 2020.