Predicting Forces for a Flapping Wing system using Linear Regression and Neural Networks Methods

Hugo Birch, Baudoin von Sury, Pierre Vuillecard
EPFL, Switzerland

***Abstract*—Nowadays, drones are a major field of innovation especially in micro aerial vehicles. Researchers are now trying to design new bio inspired types of flying robots using flapping wings. Then, understanding the flapping wing's dynamic in a fluid is crucial to design new types of flying robots. The necessary lift and power at any time of the wing's kinematic are key parameters to create a robust physic system. Until now, research have focused on finding those parameters empirically from a physical experiment system. Results from those experiments produced a large set of data which lead to the idea of predicting the lift and power from machine learning and deep learning techniques. The outcome was, the possibility to obtain the pitch evolution for a desired lift coefficient with the optimum efficiency.**

I. Introduction

The research of micro aerial vehicles (MAVs) is a new field of low-Reynolds-number flow, which attracts much attention in the advanced aeronautical area.[1]

Flapping wings is one of the solution to design MAVs. It is the most efficient way of flying for small objects with sizes less than 12 cm. Indeed, fruit fly scale robots can hover longer with flapping wings than with spinning wings and they benefit from a higher operation time and range compare to the most performant human-engineered flying vehicles.[2]

The aerodynamic performance of flapping wing flyers is directly tied to the unsteady vortex dominated flow field generated by their wings. EPFL lab UNFoLD conducted an experiment of a flapping wing model where they measured the average lift coefficient C_L and the power coefficient C_P given a kinematic. Based on the data produced, and its analysis in order to predict the forces of the flapping model, was done using Machine learning. The final aim of this experiment was to derive adapted kinematics for arbitrary lift and efficiency requirements. The analysis of this experiment led to predicting the lift and power coefficient given a kinematic, enabling a possibility to explore results beyond the system's physical constraints.[3]

II. Experiment and Data set

A. Experiment

The experiment was aimed at finding the optimum efficiency, in function of the lift coefficient of a hovering flapping wing model. With kinematics being defined by the sweeping motion of the wing, the stroke, defined by the angle ϕ and the pitch α which is the angle of attack. The lift L and power P, were determined from the torque and force measurements and then normalised using the stroke average velocity[3].

The lift and power coefficients are then derived as follow [4]:

$$C_L = \frac{L}{\frac{1}{2}\rho Rc\overline{U}^2}, \qquad C_P = \frac{P}{\frac{1}{2}\rho Rc\overline{U}^3}, \qquad \eta = \frac{\overline{C}_L}{\overline{C}_P} \tag{1}$$

To do so, the use of a Genetic algorithm was required, as they are capable to optimize multiple criterion at once. They test each individual in successive generations to find the fittest according to the objectives. The two objectives in the experiment are maximising the average lift coefficient $\overline{C}_L$ and the hovering efficiency η. The result of this optimization is the Pareto front obtained on Figure 1.a, from the bounded set of kinematics on Figure 1.b.

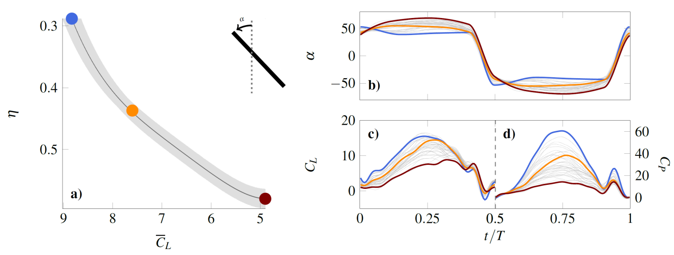

Fig. 1. a) Pareto front, plot of b) pitch α's kinematics, c) lift coefficient C_L and d) power coefficient from Adaptive pitching motion kinematics for tuning flapping wing aerodynamic performance by A. Gherke [3]

B. The data set

The study done by UNFoLD lab, lead to 4100 experiments being ran on the setup, the experiment lasted 4 seconds, for a stroke angle ϕ of 180. Each experiment, were run 8 times, then were filtered to reduce the noise, and the averaged over the 8 experiments. These 4100 experiments correspond to as many different kinematics of the pitch α, resulting in as many different lift (C_L) and power (C_P) plots. This means that the outputs don't follow a Normal Law which is an assumption which is done when using machine learning algorithms.

III. Models and methods

A. Linear Regression

Linear Regression fits a linear model with coefficients to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation [5]. It is an appropriate method to predict the coefficients $C_L(t)$, $C_P(t)$ and $\overline{C}_L$ and $\overline{C}_P$, as there is a linear correlation between the inputs, the library used to do it, is

Scikit learn[5]. The linear regression uses the mean squared error as loss function :

$$MSE = \frac{1}{N}\frac{1}{T}\sum_{n=1}^{N}\sum_{t=1}^{T}(\hat{y}(t) - y_{true}(t))^2 \quad (2)$$

In addition to the mse, the R^2 is an other score wich allows to analyse the performance of the models:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_{true}(t) - \hat{y}(t))^2}{\sum_{i=1}^{N}(y_{true}(t) - \overline{y}_{true}(t))^2} \quad (3)$$

1) Prediction of C_P and C_L

For each coefficient, 200 different values need to be predicted from the kinematic for each time t. That is why, the data set is then split for each of the 200 different values of t in order to perform the linear regression with 200 output corresponding to the coefficients. This configuration forms the baseline model define as LR baseline in Figure3.

Then, as C_P, C_L and α are time dependant, the next model defined as LR baseline + w in Figure3 also includes the previous and next inputs using a window of size $n = 2m + 1$ (equation 4).

$$C_P(t) \text{ or } C_L(t) = f(\alpha(t-m), ..., \alpha(t), ..., \alpha(t+m), w) \quad (4)$$

where $C_P(t)$ and $C_L(t)$ are the outputs at time t, $\alpha(t)$ is the kinematic at time t and w the weights of the linear regression. After computing the linear regression for different window size, the window $n = 5$ appears to be the optimal solution.

In addition, to prevent over-fitting, a regularized term has been added to the loss function (equation 5). This last configuration forms the model LR baseline + w + L_2 in Figure3.

$$loss = \frac{1}{N}\frac{1}{T}\sum_{n=1}^{N}\sum_{t=1}^{T}(\hat{y}(t) - y_{true}(t))^2 + \lambda \mathbf{w} \quad (5)$$

Finally, the input features matrix $X(t)$ has been augmented by adding a dummy variable and the squared of the same input features [6]. This configuration forms the model LR baseline + w + L_2 + squared in Figure3.

2) Prediction of $\overline{C}_L$ and $\overline{C}_P$

As the main interest of the study is to optimize the average lift coefficient and efficiency, it made sense to try to also predict the average lift and power, to see if this gave a better result. This meant having more features as the whole of the experiment could be used as a feature instead of only the ones within the selected window.

This means extracting more features from the same amount of data which, induces overfitting. A regularizer, is essential to prevent this, the Ridge regularization term best fitted the model.[7]

Finally, the feature matrix, was augmented by adding a dummy variable and adding the square of itself, as the the features are highly correlated. This, also increases the overfitting making the regularizing term even more essential.

B. Neural Networks

Neural Networks techniques have been really popular in the past few years. Neural Networks can learn any function from a data set [8]. Here the goal is to learn a function, that given a kinematic α predicts the coefficient C_P ,C_L or $\overline{C}_L$, $\overline{C}_P$. The score that is chosen to compare the goodness of fit is the same as (2). In every neural networks in this study, the loss function is the mean squared error and we use the Adam optimization algorithm[9]. The performance of neural networks technique depends a lot on the size of the data[10]. As there was only 4100 observations, the expectation was that the neural networks would not outperform the linear regression. Moreover, this low number of observation may cause overfitting. To prevent this effect, it was essential to use a regularisation term [7], such as a dropout layer or early stopping in the number of epoch [11].

The base line model was a simple neural network that takes 200 inputs and 200 outputs for C_P ,C_L or 1 output for $\overline{C}_L$, $\overline{C}_P$.

The goal was now to find a good architecture and to tune the hyper parameters, in order to obtain the minimum mean square error. The following techniques were used :

- Adding hidden layers.
- Playing with the hidden layers' sizes.
- Trying different activation functions.
- Increasing the number of epoch.
- Choosing a wise early stopping criteria.

In every neural network two approaches will be defined. The first one takes the kinematic of α as input whereas the second one takes the rescaled kinematic of α. In fact, it has been shown that scaling the input data between [0,1] improves the Deep Learning model's stability and performance.[11]

1) Prediction of C_P and C_L

The first neural network that predicts C_P from non scaled kinematic of α has 200 inputs and 200 ouptuts, with 2 hidden layers (structure CP NN tuned in Table 2). With two hidden layers its seems that the model is sufficiently complex to fit C_P. Many activation function were tried, but it seems that the elu function (6) has the best score. Here, alpha is equal to 2.5 as it gives the best score and also because the output space of elu matches with the space of C_P. Note that there is no activation function between the last hidden layers and the output.

The second neural network takes the rescaled kinematic of α in [0,1] as input. Here, the structure is defined in Table 2 as CP NN tuned + rescale. There is only one hidden layer but its size is bigger than the other two layers, and respects the rule of 2/3 of the size of input plus the size of the output[12]. The Sigmoid activation function gives the best result. It may be because the input that is between 0 and 1.

Finally, to predict C_L the same procedure was used. Again, the structure of the two neural networks for input kinemactic of α and rescaled kinematic of α can be found in Table 2. The architecture of these two structure are better than the previous ones. In fact, 4 hidden layers have been used and their size has been decreased by a factor two. This configuration can tackle the problem of high correlation in the input. In fact, it is possible only half of the data are really interesting to fit the coefficient lift. The activation function are similar except for the alpha of elu that is equal to 1.

$$elu = \begin{cases} y = \alpha(e^x - 1) \text{ if } x < 0 \\ y = x \text{ if } x > 0 \end{cases} \tag{6}$$

	Layer	Number of nodes	Activation Function
CP NN tuned	Input	200	
	2 Hidden Layers	(200 , 200)	elu (alpha = 2.5)
	Output	200	
CP NN tuned + rescale	Input	200	
	1 Hidden Layer	(350)	sigmoid
	Output	200	
CL NN tuned	Input	200	
	4 Hidden Layers	(128 , 64 , 64, 64)	elu (alpha = 1)
	Output	200	linear
CL NN tuned + rescale	Input	200	
	2 Hidden Layers	(128 , 64)	sigmoid
	Output	200	

Fig. 2. Table of neural networks structures to predict C_P and C_L

2) *Prediction of $\overline{C}_L$ and $\overline{C}_P$*

To design the neural networks in order to predict $\overline{C}_L$ and $\overline{C}_P$, the techniques described in section III-B1 were also used. The structures of the different neural networks are not specified in this report because the performance of those ones are not relevant enough. However, the MSE score are listed in table 3.

C. *Results*

All the results in Figure 3 are computed with a 4-fold cross validation. Thanks to this cross validation, it was possible to find the best value for the regularizing term as seen in Figure 3 and got more relevant results.

The base line model for the linear regression is relatively poor for the C_P prediction, and slightly better for the C_L prediction. Base line for $\overline{C}_P$, mean $\overline{C}_L$ is already much better. Then, by adding a window, this decreases by half the mean square error. Concerning the overfitting, for C_P and C_L adding a penalized term doesn't seems to decrease a lot the mean square error but it has a slightly better R^2 (3). Unlike $\overline{C}_P$ and $\overline{C}_L$ that have overfitting, adding a penalizing term improves the model (better mse and R^2). Finally, augmenting the input features with the square of itself gives a much better prediction as long as its combined with a regularizing term. However the result for $\overline{C}_P$ and $\overline{C}_L$ could lead to some error because the design matrix is ill conditioned. Especially for mean CL where the difference between adding the square or not is very high.

Concerning the neural network the base line model has poor results to predict C_P and better result to predict the other coefficient but not as good as the linear regression base line. After tuning all parameters, the score improves drastically, in fact it's nearly 20 times better for C_P. However the result are not better than the best linear regression model, especially for mean C_L. But, if the inputs are rescaled, the model becomes as

Models-MSE	CP	CL	mean CL	mean CP
Linear Regression				
LR base line	3.499	0.617	0.0864	0.0796
LR window	1.134	0.377	x	x
LR window + L_2 (λ)	1.133(0.016)	0.376(0.016)	0.0728(0.6)	0.0682(0.6)
LR window + square + $L_2(\lambda)$	0.784(4e-4)	0.155(4e-4)	0.004(78)	0.0303(263)
Neural Network				
NN base line	15.153	1.499	2.023	13.864
NN tuned	0.826	0.155	0.795	11.092
NN base line rescaled	1.178	0.361	0.107	0.190
NN rescaled + tuned	0.756	0.149	0.020	0.122

Fig. 3. Results of the models, using mean squared error as a score

good as the linear regression with a window. After tuning the parameter of a rescaled neural network we get slightly better result. In fact for C_L and C_P the neural networks are slightly better than the best linear regression. But again it doesn't beat the mean C_L and mean C_P from the best linear regression model.

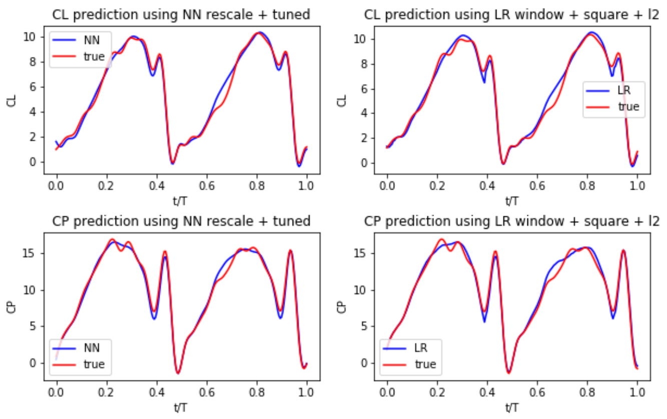

Fig. 4. Plot of the predicted C_l and C_P over one period compared to the true value for different models

Finally, as expected, the neural network doesn't outperform the linear regression overall. Despite that the universal approximation theorem [8] states that a simple neural network can fit any continuous function. Here, if we look at Figure 4 we can clearly see that the prediction is almost perfect. So is the linear regression's prediction, which also fits the data pretty well. It may be, that the data is too simple to see a large difference between linear regression and neural network.

IV. Prediction of kinematics from $\overline{C}_L$ and $\overline{C}_P$

First, in this study, efficient model have been designed to predict C_P, C_L, $\overline{C}_L$ and $\overline{C}_P$. But now to go further, what if it could be possible to predict kinematics of a flapping wing given average coefficients $\overline{C}_L$ and $\overline{C}_P$. What about if from specific requirements (efficiency and $\overline{C}_L$) for a Micro Aerial Vehicles, it could be possible to predict the optimal kinematic to apply to the flapping wings of the robot ? This part of the study tries to answer this problem.

A. *Models and methods*

From two features, predicting a kinematic α with 200 discrete time values is not an easy task. Again, we compare linear regression and Neural Network to solve this problem.

Concerning the linear regression for each time t of the kinematics $\alpha(t)$ a model was built with two feature average $\overline{C}_P$ and $\overline{C}_L$. This model is a ridge regression with feature expansion with a specific degree n. Following the same process as described in section III-B1 , but with a model with input equal to 2, rescaled between 0 and 1 and output equal to 200, an efficient model with the structure described in table 5 have been designed.

	Layer	Number of nodes	Activation Function
Alpha NN	Input	2	
	4 Hidden Layers	(32 , 64, 128, 200)	sigmoid
	Output	200	

Fig. 5. Neural Network structure to predict kinematic alpha

B. Results

The linear regression model described above has a $MSE = 14.267$ with a penalize term $\lambda = 0.483$ and adding the feature to a degree 2. A degree 3 improve slightly the model but nothing compare to the variance of the output. Indeed, the output takes value between 80 and -80.
Note that all the hyper parameter have been chosen with a 4 fold cross validation. The neural networks have almost the same results, in fact it gets a $MSE = 13.6$, again using a 4 fold cross validation.

Let's take an approximation of the Pareto front using a simple linear regression. From this approximation, 15 equally spaced points are selected. Some of those points are deliberately selected outside the bounds of the data set (cf 6) in order to analyse the behaviour of the models to predict kinematic α on unseen data $\overline{C}_P$ and $\overline{C}_L$.

The two models designed (linear regression and neural network) were then tested. It appeared that the linear regression predicted poorly on unseen data. Indeed, the predicted kinematics α was outside of the physical bounds (pitch angle α reach 400). On the other hand, neural networks predictions seems more physically acceptable.
Therefore, neural network is used to predict kinematic α of the 15 selected before, seen in Figure 6.

Finally, to validate the predicted kinematics. They have been reset as input of the two models (LR window + square + L_2 cf 3) to predict again the value of $(\overline{C}_L, \overline{C}_P)$ and then to plot it again on the Pareto front plot. It appears that after all this loop process, the predicted points are really close to original data, as long as they have been chosen close to the points from the original data set.
Thus, allowing to predict the Pareto front and its corresponding α, C_L and C_P. Nevertheless, the points selected outside do not follow the same dynamic.

V. Conclusion and improvements

The first aim of the study was to predict the lift C_L and power C_P coefficients from its corresponding pitch evolution α. The different models created gives a reliable prediction as long as the kinematic of α is within the bounds given by the

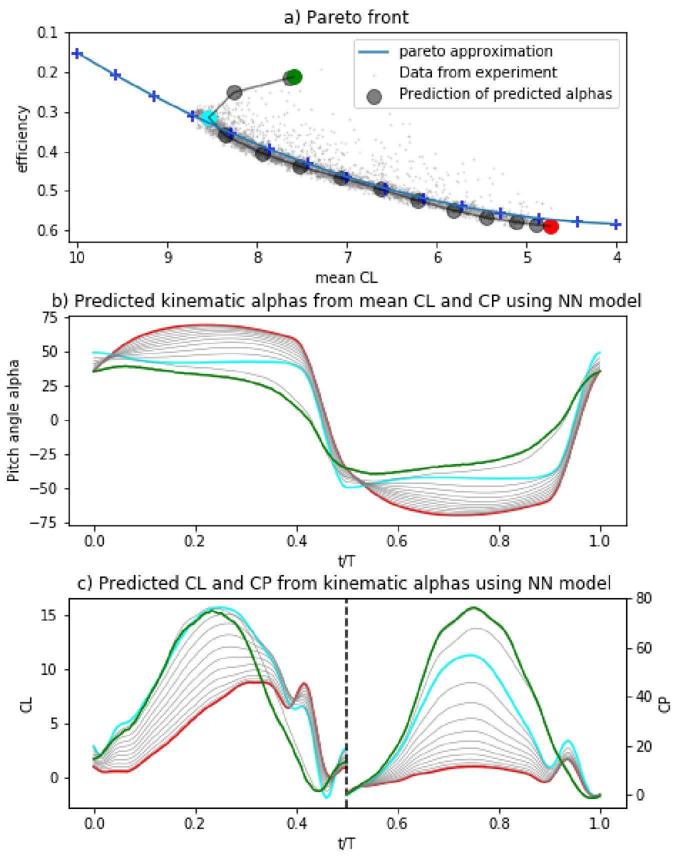

Fig. 6. Prediction of Pareto front, its corresponding pitch kinematic, C_L and C_P

limit of the experimental setup. No evidence has been found enabling to say which algorithm is superior between Linear Regression and Neural Network.
It was then decided to go further and apply Machine Learning techniques, which has allowed to reproduce the Pareto Front. Moreover, this allows to extract the optimum efficiency, and to get the corresponding kinematics of α for any desired lift. Which isn't possible with the optimized experiment as it gives discrete values.

A possible way to improve the predictions, is to apply the Machine Learning techniques directly on the 7 parameters, from which the function which characterizes the pitch evolution α is derived. Indeed, as it's these parameters which need to be optimized.
An idea for the next steps, would be to test some of our predictions on the experimental setup.

VI. Acknowledgement

This project was only possible thanks to the Laboratory UNFoLD and especially Alexander Gehrke and Karen Mulleners.

References

[1] L.-J. Yang, C. Hsu, J.-Y. Ho, H.-H. Wang, and G.-H. Feng, “The micro aerial vehicle (MAV) with flapping wings,” Aug. 2005, pp. 811–815.

[2] E. W. Hawkes and D. Lentink, “Fruit fly scale robots can hover longer with flapping wings than with spinning wings,” *Journal of The Royal Society Interface*, vol. 13, no. 123, p. 20160730, Oct. 2016. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsif.2016.0730

[3] A. Gehrke and K. Mulleners, “Adaptive pitching motion kinematics for tuning flapping wing aerodynamic performance,” 2019. [Online]. Available: https://infoscience.epfl.ch/record/272185

[4] A. Gehrke, G. De Guyon-Crozier, and K. Mulleners, “Genetic Algorithm Based Optimization of Wing Rotation in Hover,” *Fluids*, vol. 3, no. 3, p. 59, Sep. 2018. [Online]. Available: https://www.mdpi.com/2311-5521/3/3/59

[5] “sklearn.linear_model.LinearRegression — scikit-learn 0.22 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[6] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[7] B. Ghojogh and M. Crowley, “The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial,” *arXiv:1905.12787 [cs, stat]*, May 2019, arXiv: 1905.12787. [Online]. Available: http://arxiv.org/abs/1905.12787

[8] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, “The Expressive Power of Neural Networks: A View from the Width,” in *NIPS*, 2017.

[9] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[10] M. Johnson and D. Q. Nguyen, *How much data is enough? Predicting how accuracy varies with training data size*. September, 2017.

[11] J. Brownlee, *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, Dec. 2018.

[12] F. S. Panchal and M. Panchal, “Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network,” 2014.