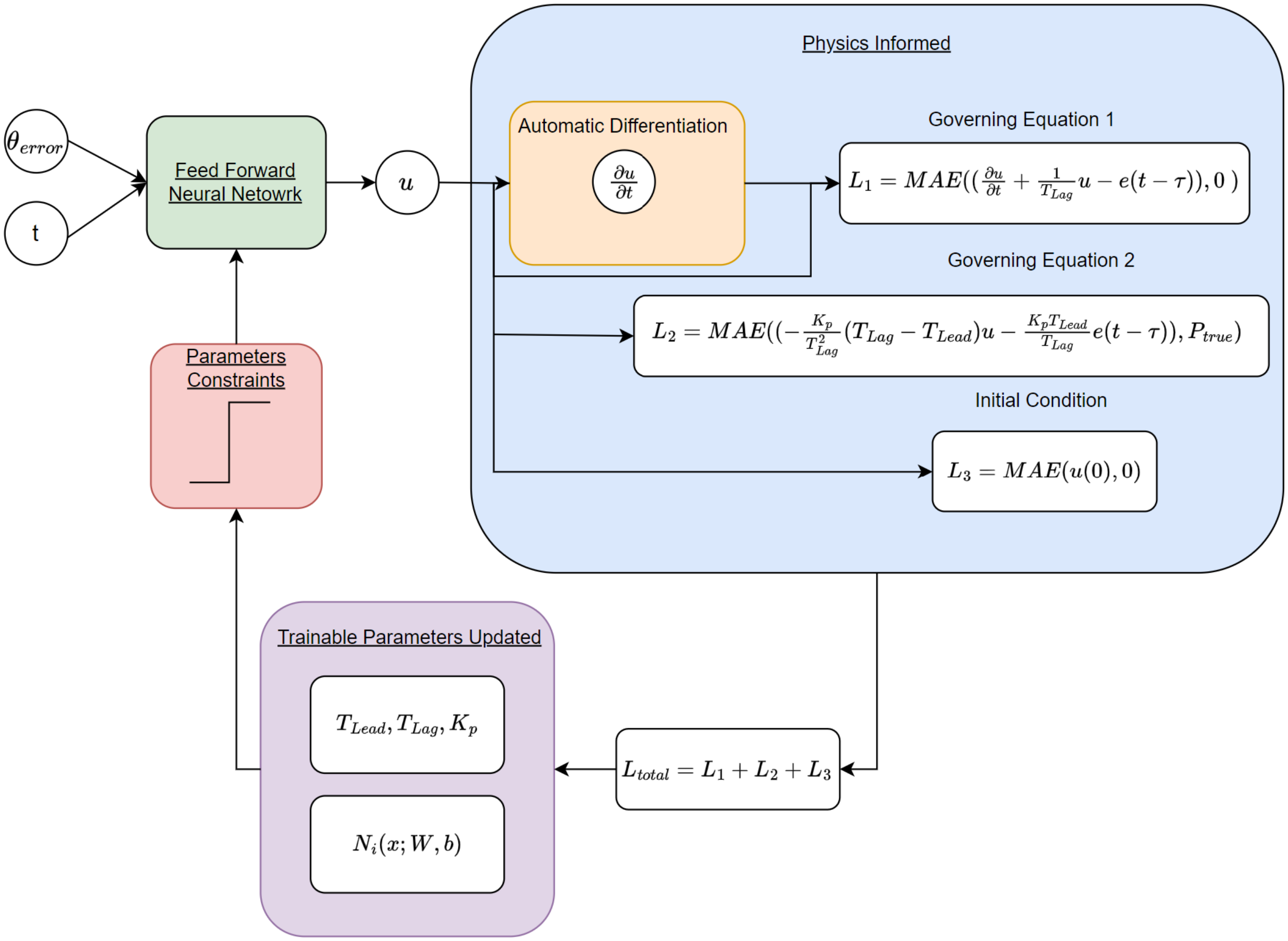
Google Colab Code 1 Notes

Intro: In this code the pilot model equations are taken from Thomas's thesis, make sure to take them from this because the ones from the Mandal WVY thesis are off by negative 1. These equations are for the pilot in which to keep in mind, the time delay τ is implicit and thus we are modeling the data as is without predicting the time delay -> aka we are only estimating T_{Lag}, T_{Lead}, K_p . These equations are also without the Pade approximation thus x is a nx1 state instead nx2. This is done so that we have less things essentially that we have to estimate. See below for the equations

$$\dot{x}(t) = -\frac{1}{T_{Lag}}x(t) + e(t-\tau)$$

$$p(t) = -\frac{K_p}{T_{Lag}^2}(T_{Lag} - T_{Lead})x(t) - \frac{K_p T_{Lead}}{T_{Lag}}e(t-\tau)$$

Architecture:


Physics Informed
θ_{error}
t
Feed Forward Neural Netowrk
u
Automatic Differentiation
$\frac{\partial u}{\partial t}$
Governing Equation 1
$L_1 = MAE((\frac{\partial u}{\partial t} + \frac{1}{T_{Lag}}u - e(t-\tau)), 0\,)$
Governing Equation 2
$L_2 = MAE((-\frac{K_p}{T_{Lag}^2}(T_{Lag} - T_{Lead})u - \frac{K_p T_{Lead}}{T_{Lag}}e(t-\tau)), P_{true})$
Initial Condition
$L_3 = MAE(u(0), 0)$
Parameters Constraints
Trainable Parameters Updated
T_{Lead}, T_{Lag}, K_p
$N_i(x; W, b)$
$L_{total} = L_1 + L_2 + L_3$

The feed forward Neural Neural network block is used to denote the Neural Network used in this work. The architecture of the network used is 2 hidden layers of 64 and 128 neurons respectively with Tanh() activation functions, Adam optimizer of learning rate of 0.0001, and 1001 epochs. Note for this formulation, the pilot parameters are defined as trainable parameters in the training loop, i.e. they are updated through each epoch just like the Neural Network weights and biases by the Adam optimizer. The automatic differentiation block allows the network to go from x to $\dot{x}$ and this is done by Pytorch's automatic differentiation function torch.autograd(). Lastly the parameters constraints block was implemented to be able to saturate the McRuer model parameters within certain reasonable bounds as the training occurred to reasonable min and max values as consistent with the literature.

Problem Formulation: If the standard relationship between the Neural Network input and output is,
$y = N_i(x; W, b)$ the relationship for the iPINN is $y = N_i(x; W, b, \theta)$ where θ can be denoted as any trainable parameters. For this setup the trainable parameters are the McRuer Pilot model parameters, T_{Lead}, T_{Lag}, K_p. The loss function for the iPINN will update the weights and biases of the Neural Network along with the pilot parameters.

The governing equations for the McRuer Pilot model are,
$\dot{x}(t) = -\frac{1}{T_{Lag}} x(t) + e(t - \tau)$

$p(t) = -\frac{K_p}{T_{Lag}^2}(T_{Lag} - T_{Lead})x(t) - \frac{K_p T_{Lead}}{T_{Lag}} e(t - \tau))$

$x(t = 0) = 0$

The iPINN will utilize these governing equations as part of its loss function. The loss function, L, can be defined as,
$L_{total} = L_{Data} + L_{ODE} + L_{I.C.}$
Where L_{Data} will be denoted as the residual of the pilot output, L_{ODE} will be denoted as the residual of the differential equation and $L_{I.C.}$ will be denoted as the residual of the initial condition. The residuals can be calculated as,

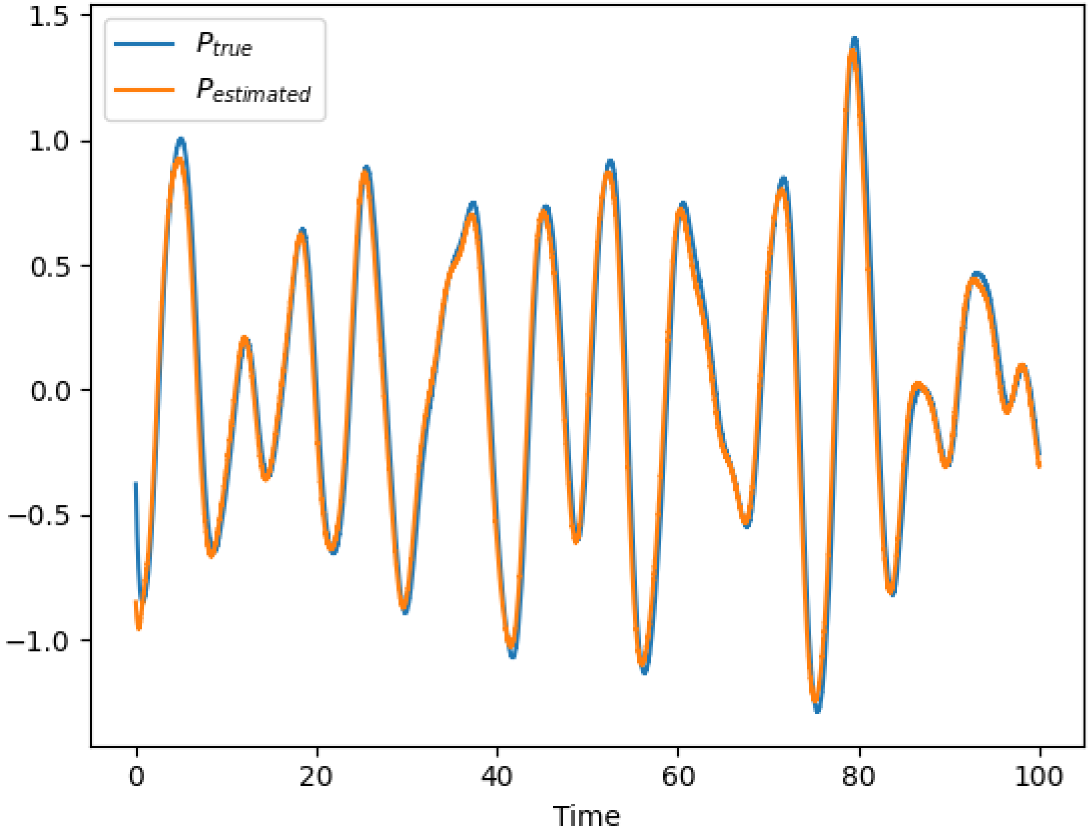
$L_{Data} = MAE((-\frac{K_p}{T_{Lag}^2}(T_{Lag} - T_{Lead})u - \frac{K_p T_{Lead}}{T_{Lag}} e(t - \tau)), P_{true})$

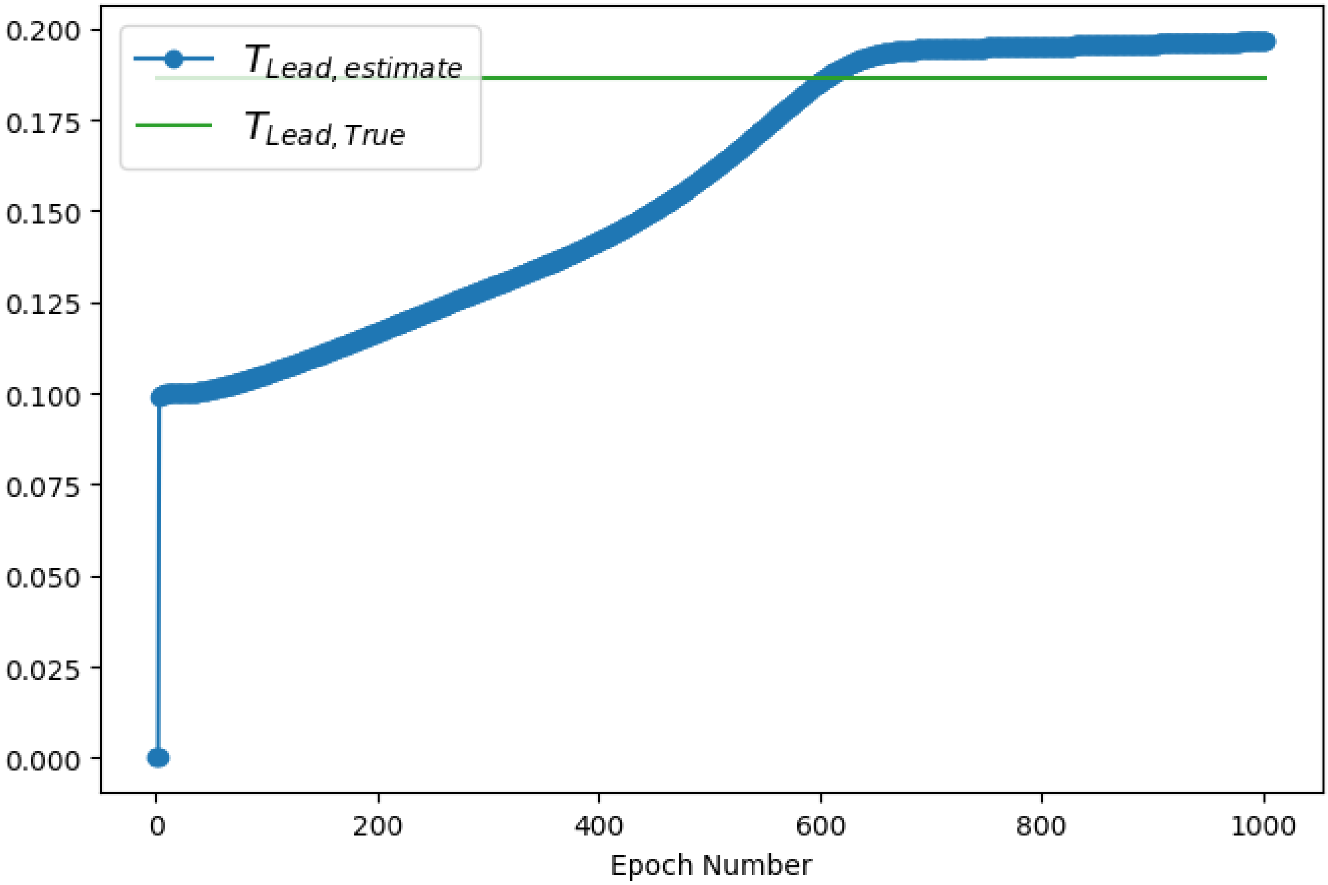
$L_{ODE} = MAE((\frac{du}{dt} + \frac{1}{T_{Lag}} u - e(t - \tau)), 0)$

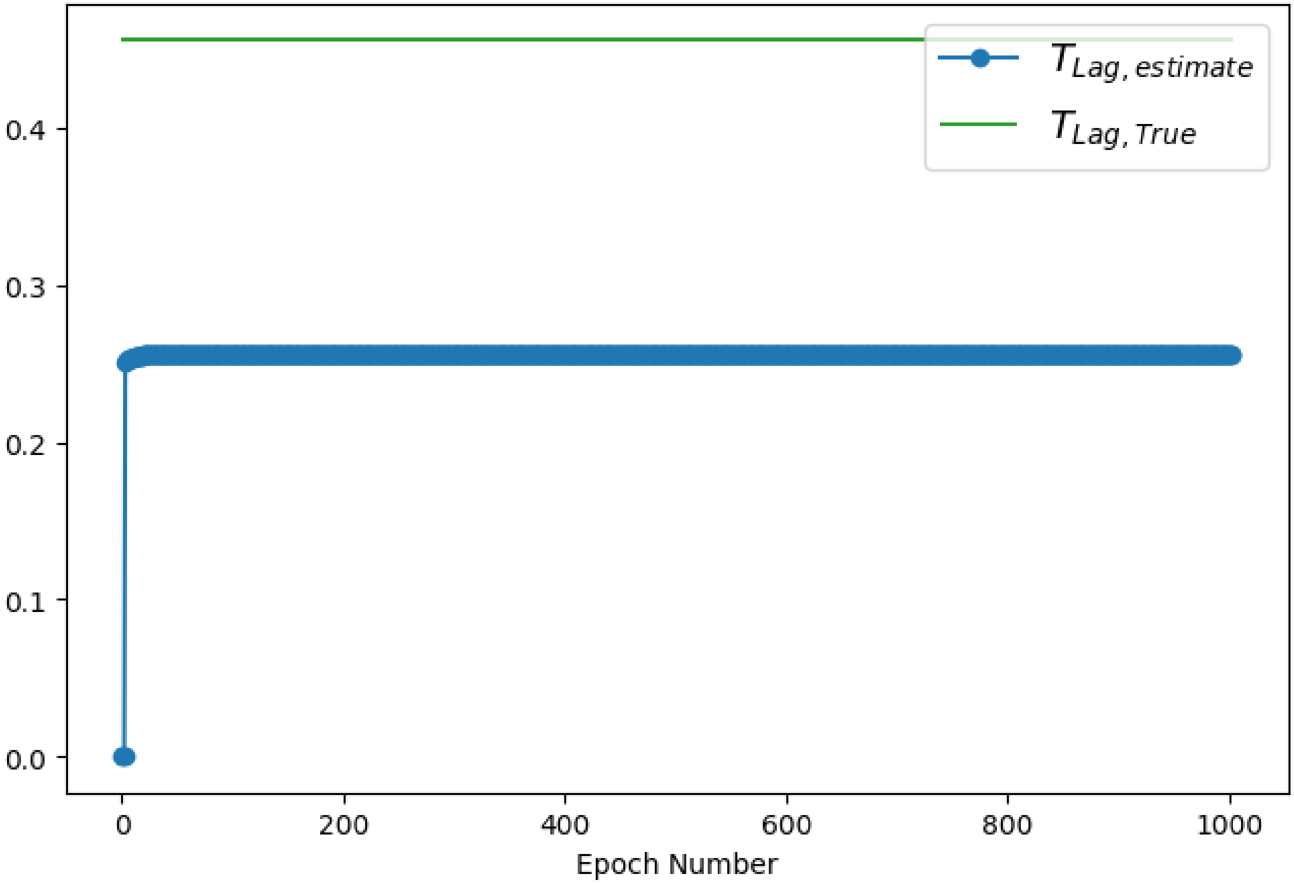
$L_{I.C.} = |u(t = 0) - 0|$
Where MAE stands for the mean absolute error. To note we can add scale factors to each of the loss terms i.e.
$Loss = \sum_{i=1}^{n} \lambda_i L_i$
where lambda is the weight factor and L is each of the loss terms. There has been research done on other methods to balance these multiple loss terms such that the terms are evenly

weighted/not one term dominates the losses, however those approaches in this research have not been explored. The lambda values for this were kept to 1 as finding these lambda values would at the moment require manual tuning and retraining which is time consuming.

Results:




$T_{Lead, estimate}$
$T_{Lead, True}$
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
200
400
600
800
1000
Epoch Number


$T_{Lag, estimate}$
$T_{Lag, True}$
0.4
0.3
0.2
0.1
0.0
0
200
400
600
800
1000
Epoch Number

$K_{p, estimate}$
0.40
0.35

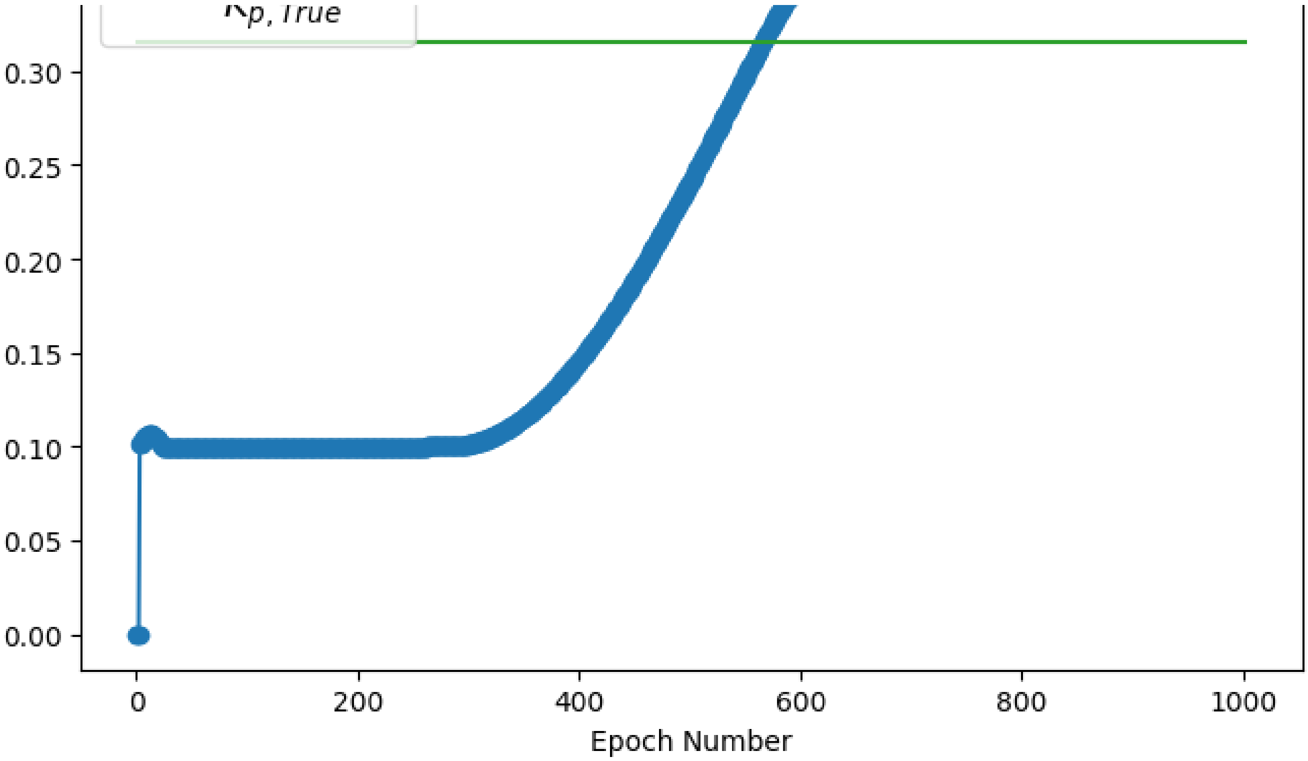
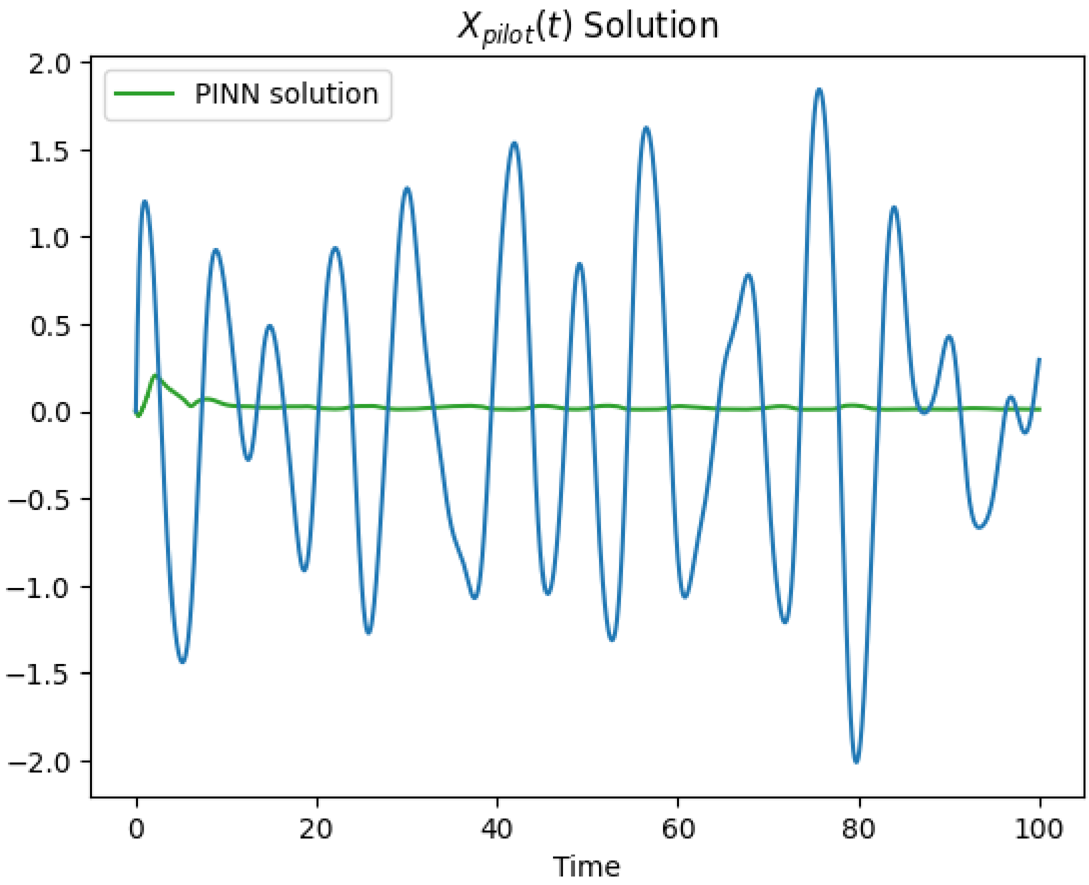


Table 1: Analysis Results from iPINN training

	$p(t)$	T_{Lead}	T_{Lag}	K_p
MAE	0.0916	0.0103	0.1993	0.0629
% difference	6.69	5.52	4.377	19.97



Analysis: With a lack of a forcing function in the loss function for the parameters to converge to their true values the Network simply just learns a combination of the parameters that results in a minimized loss function. The result of the parameters constraints can be seen in the parameters training figures where the parameters are initialized at $1 * 10^{-4}$ but after the first epoch they jump to the lower bound of the constraint and train from there. Even though the parameters may have trouble converging to the true values the overall response of the system, the pilot output does converge very well. Looking at weighting each part of the loss function can also be explored as well.

The drawbacks for this method is that now the pilot parameters are a part of the NN thus if we were to use what looks like a well trained NN on a "different pilot" aka different parameters used in simulation we would find that the pilot output will not estimate well bc it is using the previously learned parameters. So we can only say the NN learned the behavior of this 1 pilot. Lastly its interesting that the pilot output can match but the $x(t)$ does not match at all.

The true parameters for this run are:
$T_{Lead} = 0.1862$
$T_{Lag} = 0.4533$
τ= 0.0.3152
The parameters were clamped for this way such that,
$0.1 < T_{Lead} < 3$

$0.25 < T_{Lag} < 0.65$
$0.1 < K_P < 0.5$

We constrain the parameters consistent with the literature's approximate values and to allow the neural network to learn values that are realistic (see below).

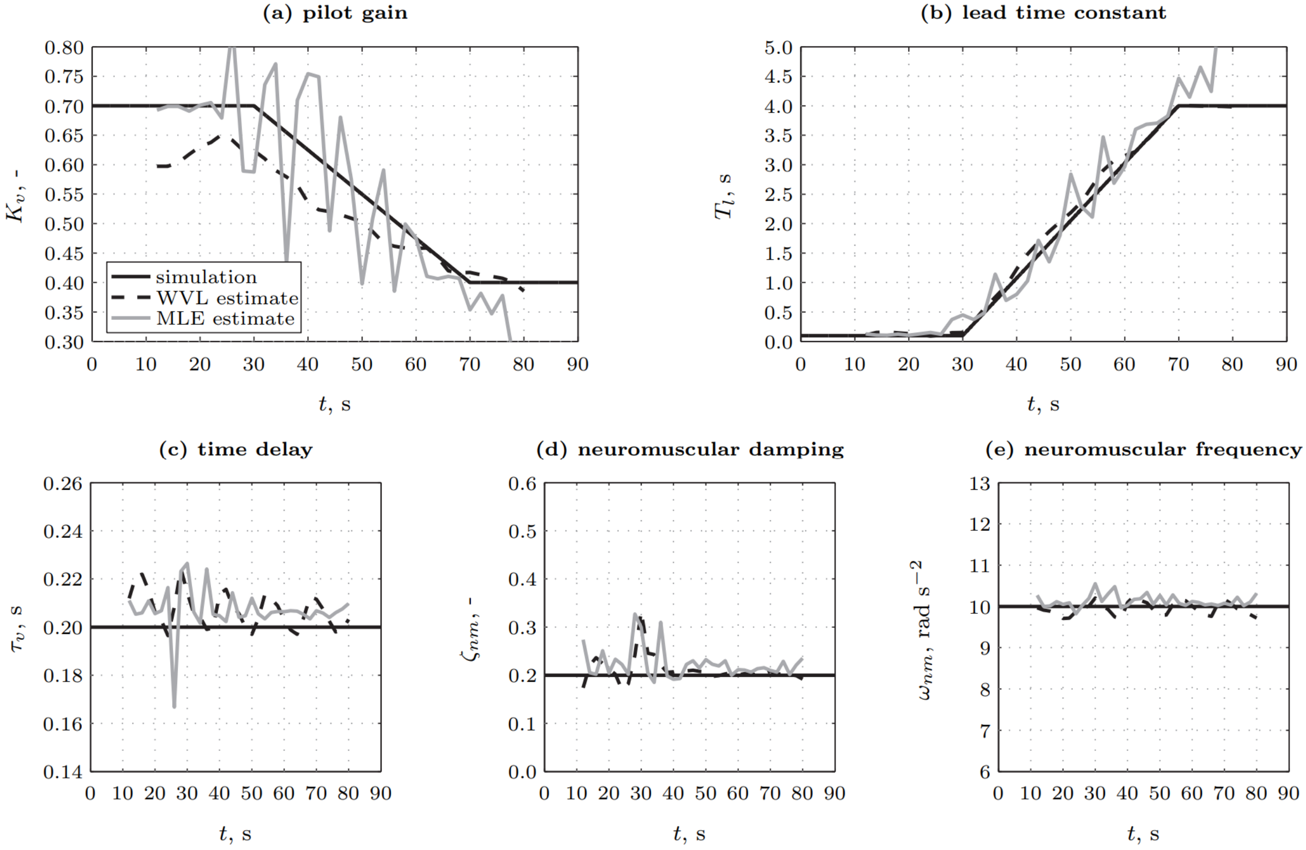


Figure 14: Estimated time-varying pilot model parameters from wavelet and MLE methods (P_n=0).

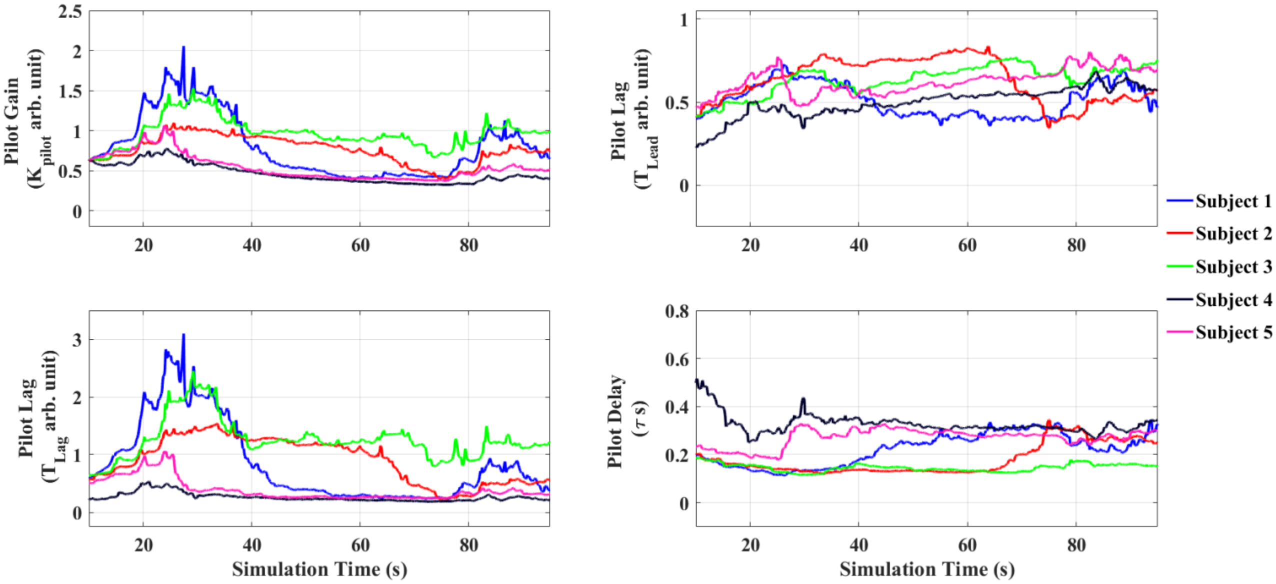


FIGURE 3.18: Comparison of estimated parameters for 5 different subjects for a single trial under nominal condition.

Table 5.2 Time-invariant pilot model parameters used to produce the results shown in cases 1 and 3. These values were derived from Monte Carlo simulations and retrieved from similar studies of the McRuer model [5] [6].

Parameter	Value
K_p	0.54
T_{Lead}	0.32
T_{Lag}	0.4
τ	0.25s

To see the estimation with a larger range of saturation I adjusted the bounds in the code to:

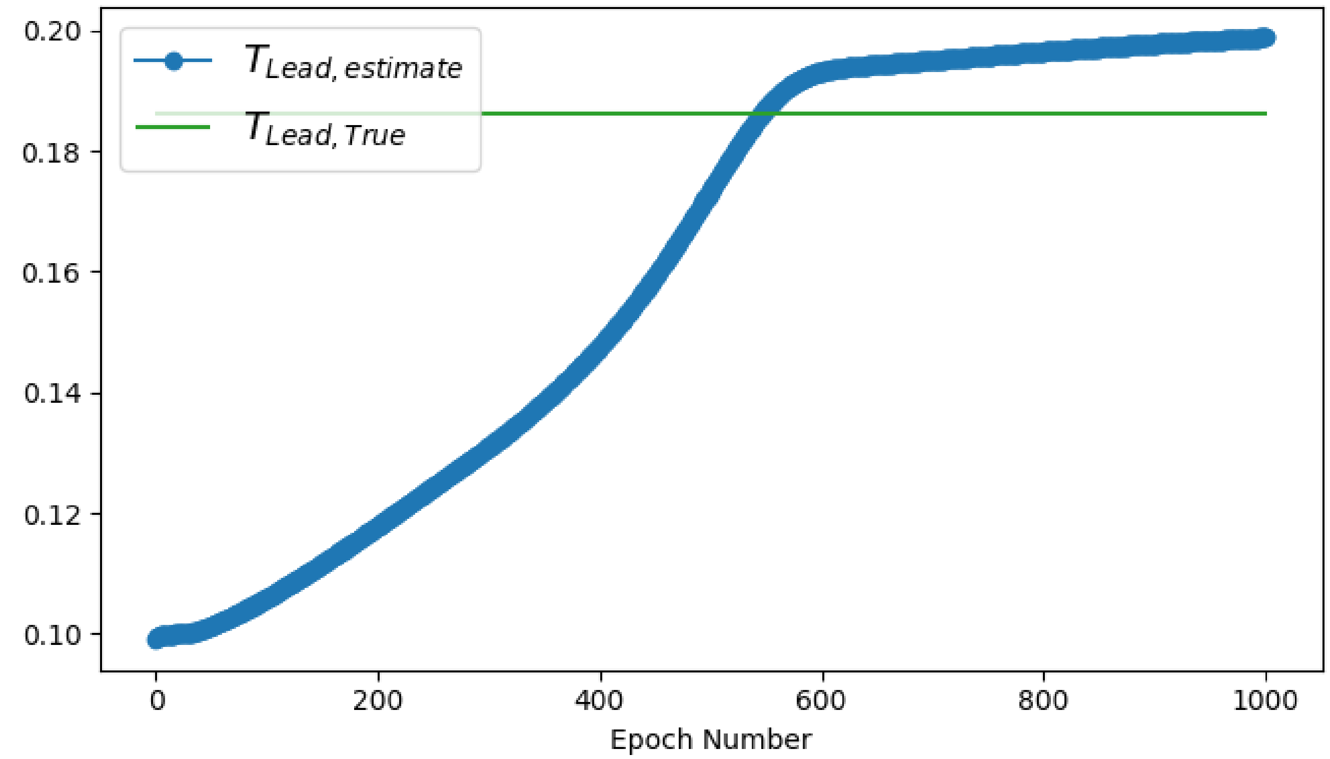
$0.1 < T_{Lead} < 3$

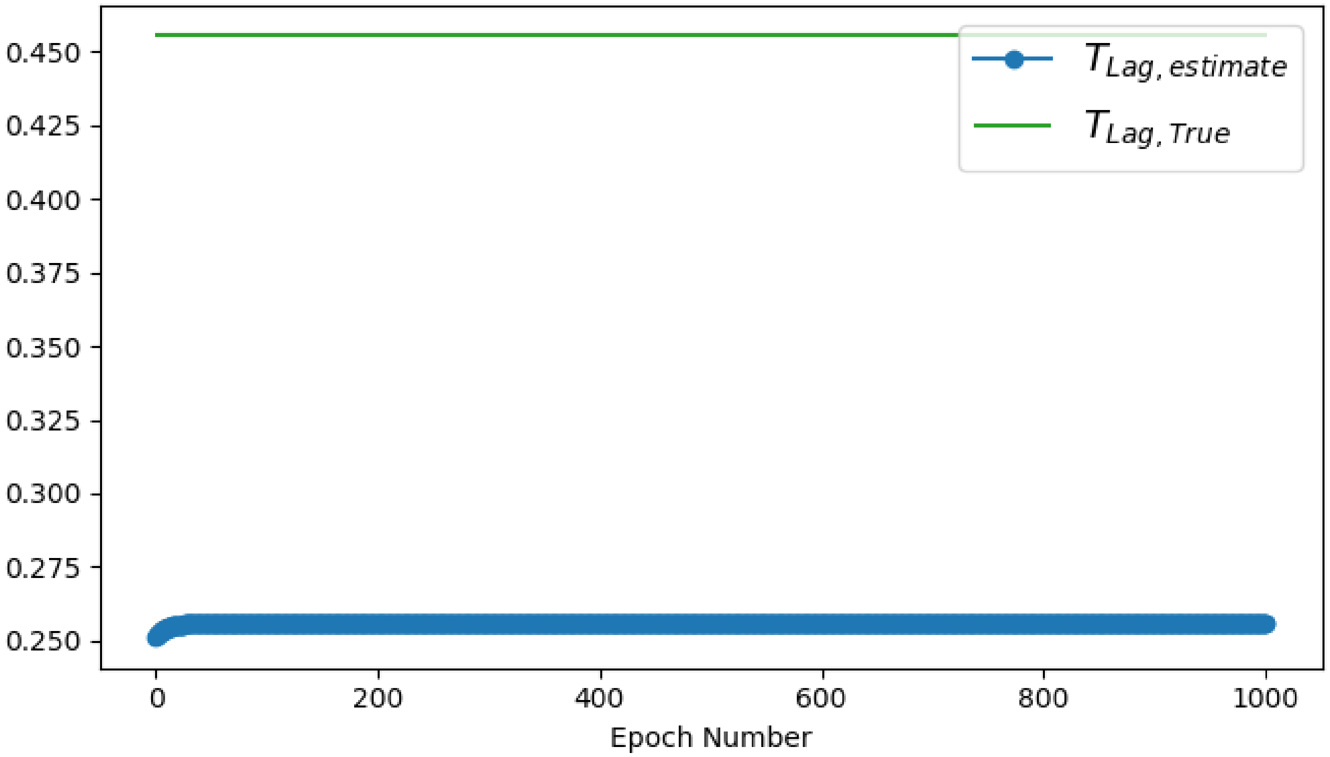
$0.25 < T_{Lag} < 3$

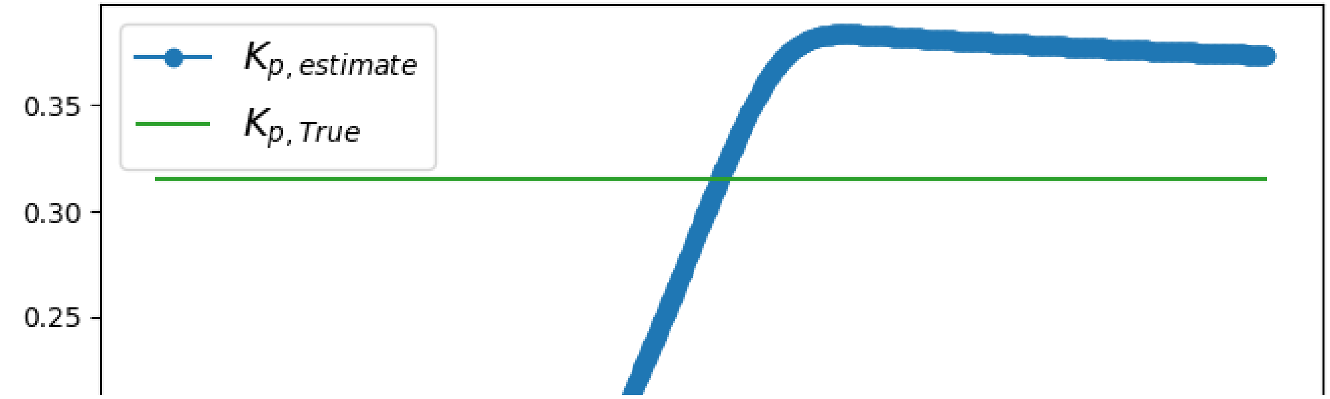
$0.1 < K_P < 2$

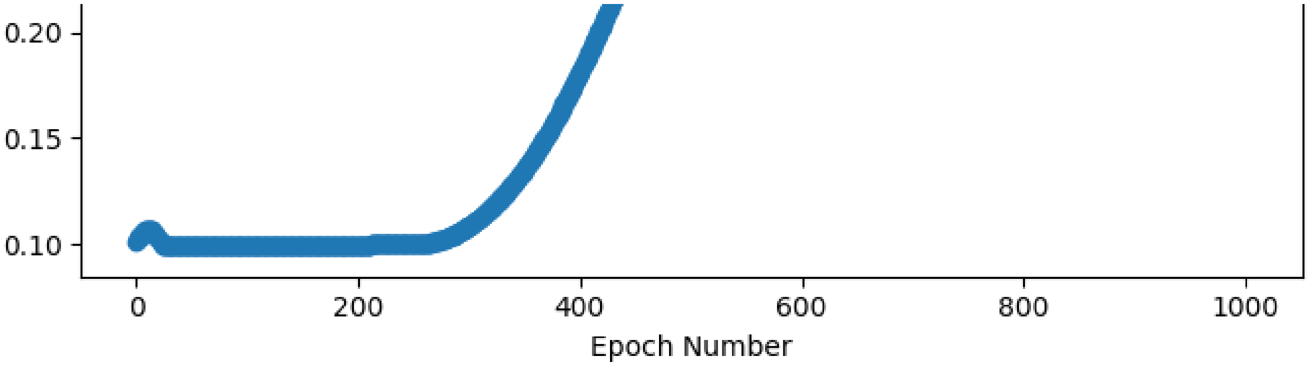
the Results can be seen below:

Trial 1:

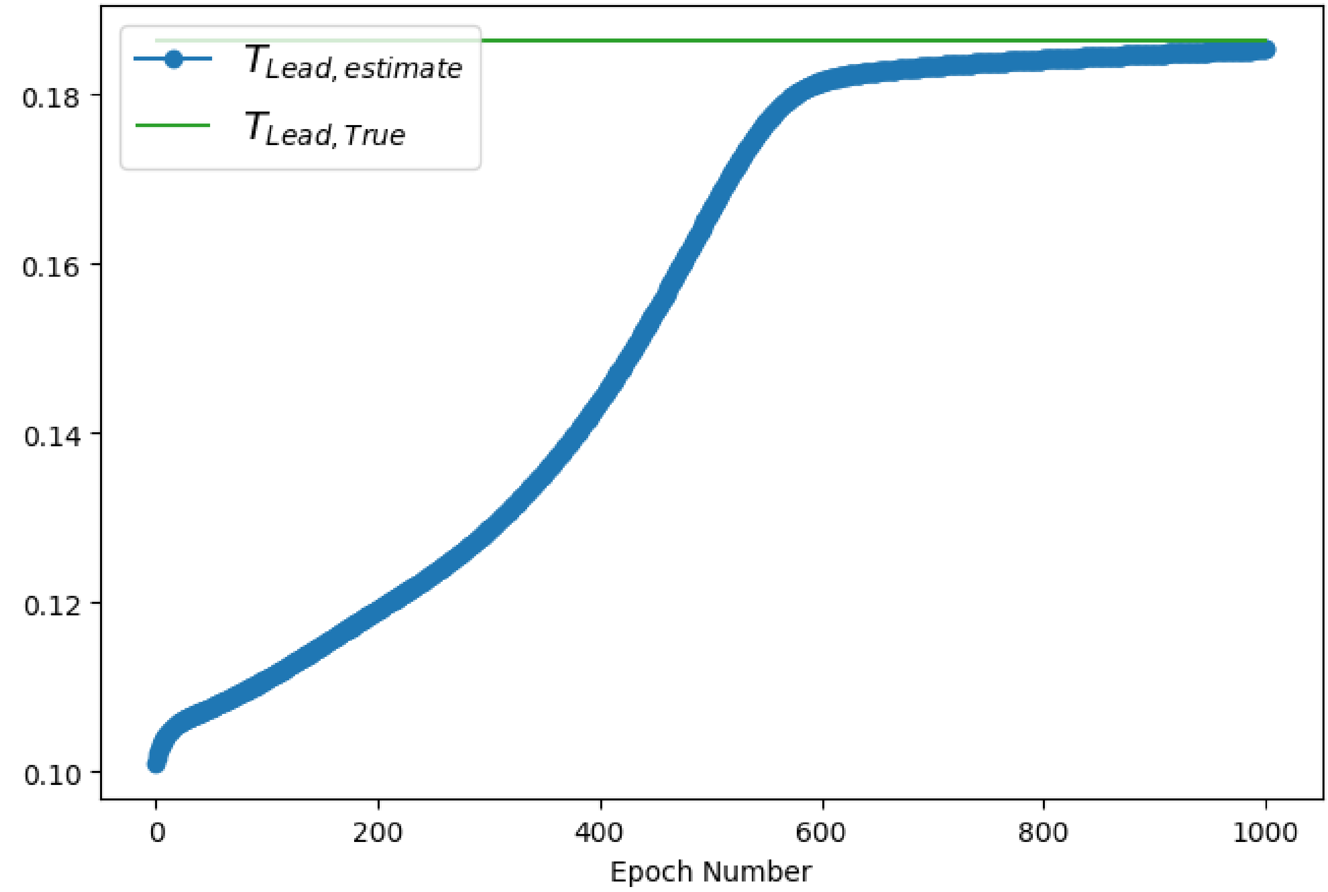

$T_{Lead, estimate}$
$T_{Lead, True}$
0.20
0.18
0.16
0.14
0.12
0.10
0
200
400
600
800
1000
Epoch Number

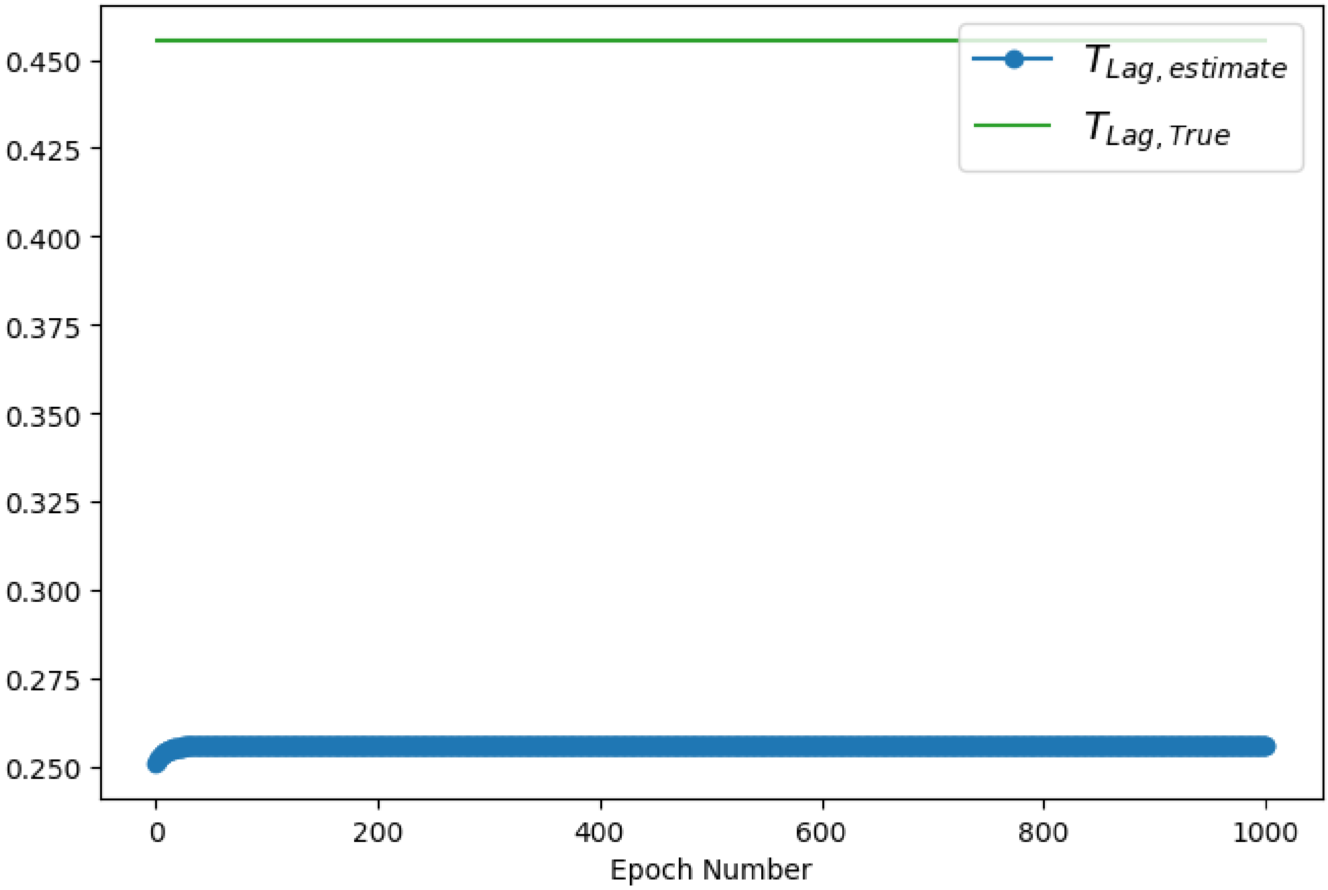

$T_{Lag, estimate}$
$T_{Lag, True}$
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0
200
400
600
800
1000
Epoch Number

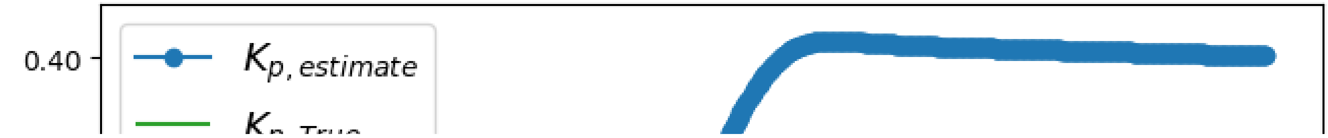

$K_{p, estimate}$
$K_{p, True}$
0.35
0.30
0.25

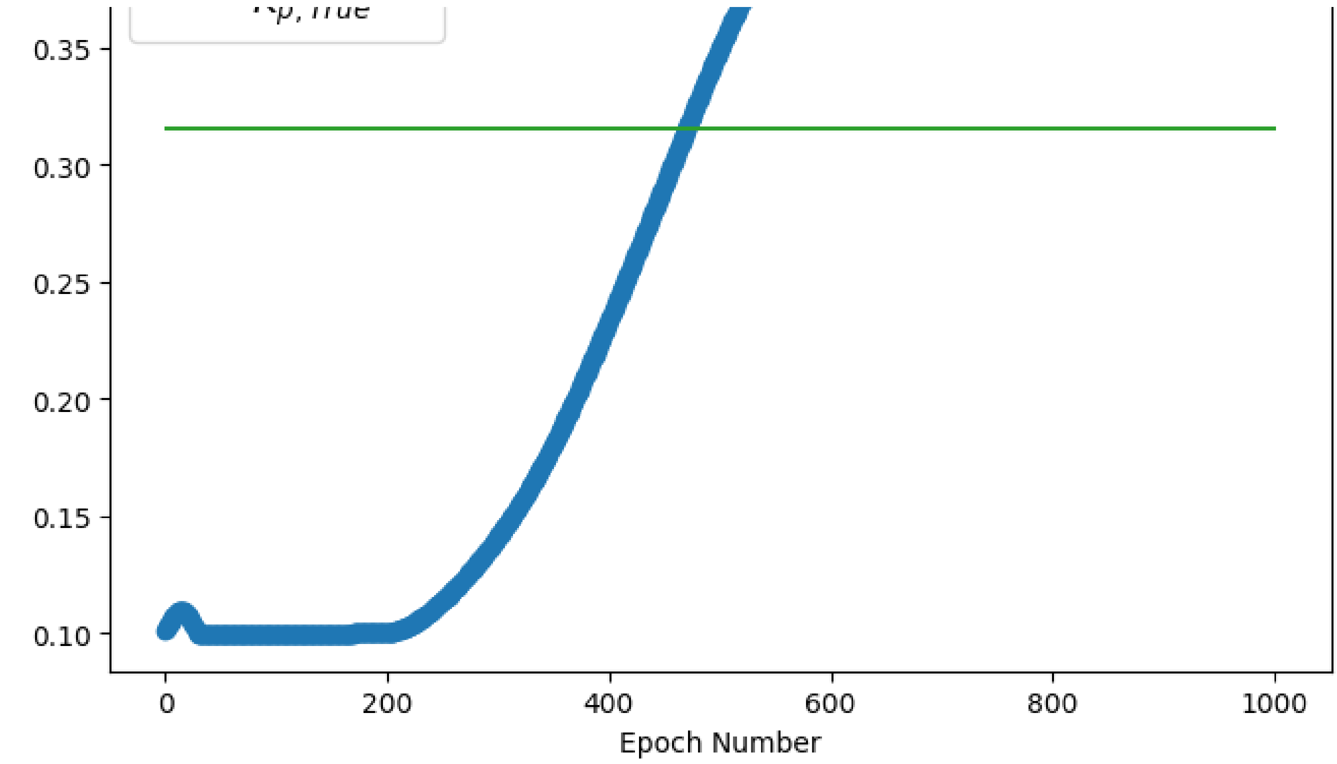

0.20
0.15
0.10
0
200
400
600
800
1000
Epoch Number

Trial 2:

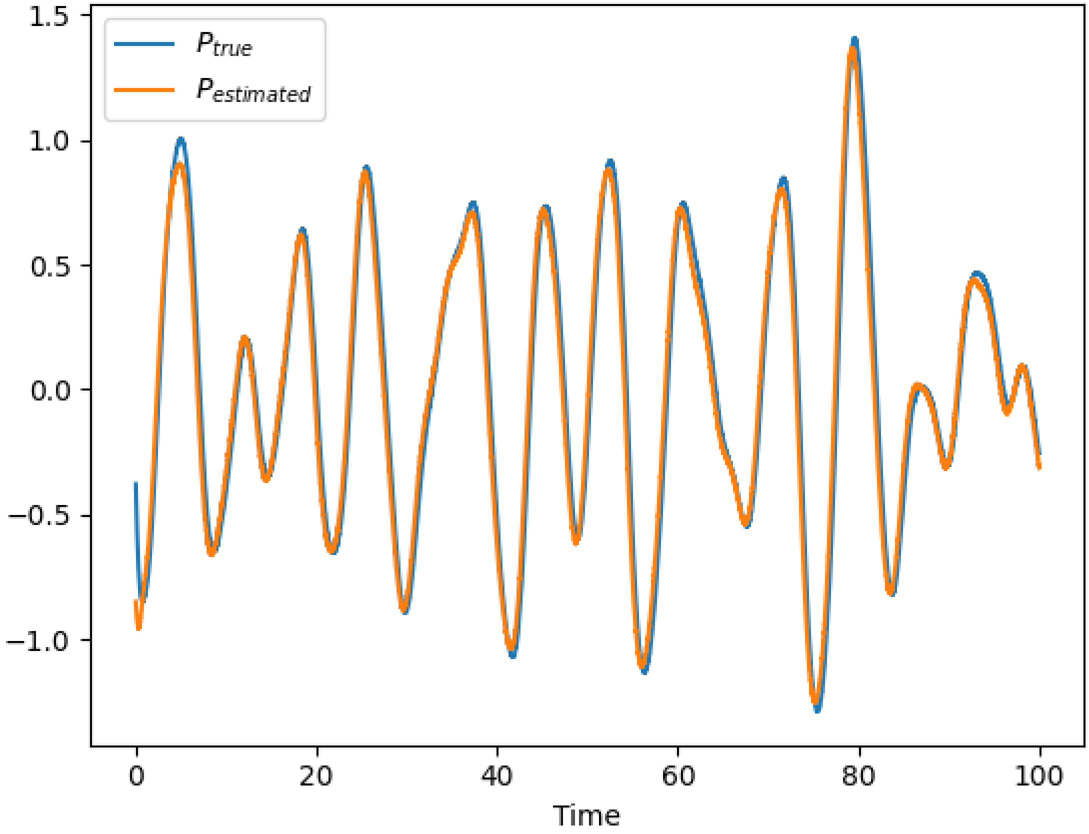

$T_{Lead,\,estimate}$
$T_{Lead,\,True}$
0.18
0.16
0.14
0.12
0.10
0
200
400
600
800
1000
Epoch Number


$T_{Lag,\,estimate}$
$T_{Lag,\,True}$
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0
200
400
600
800
1000
Epoch Number

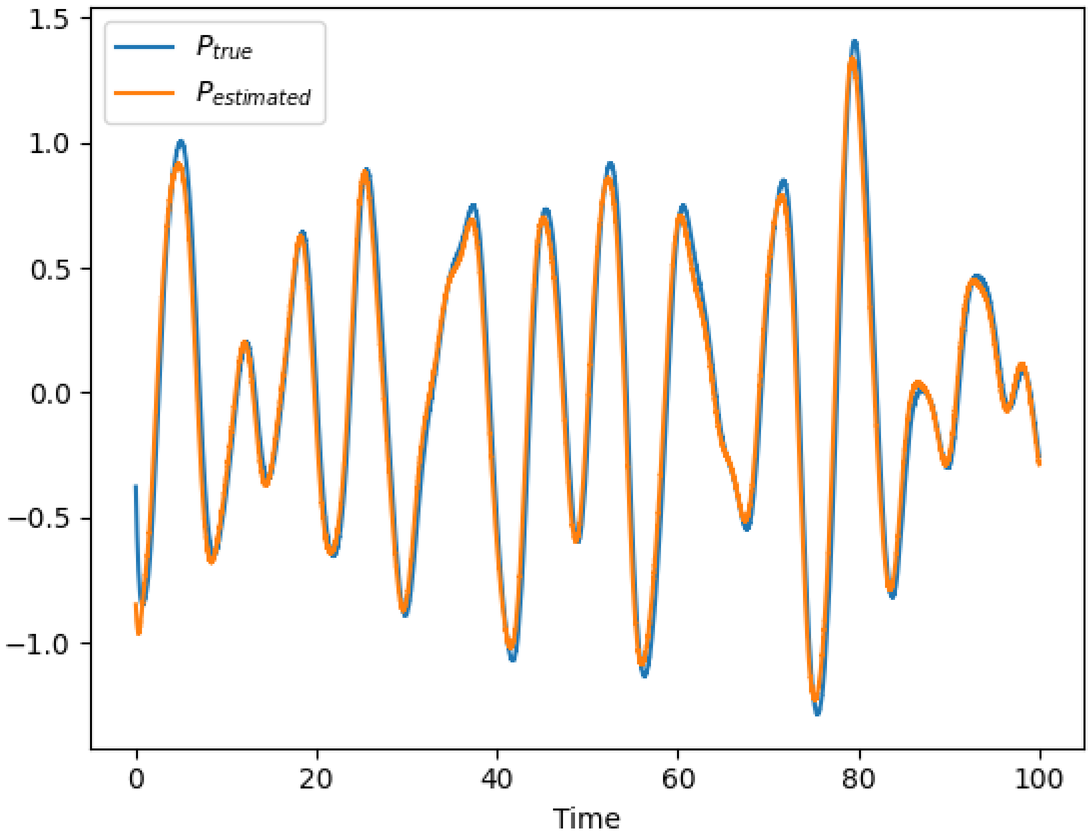

$K_{p,\,estimate}$
$K_{p,\,True}$
0.40


$K_{p,true}$
0.35
0.30
0.25
0.20
0.15
0.10
0
200
400
600
800
1000
Epoch Number

Trial 1:

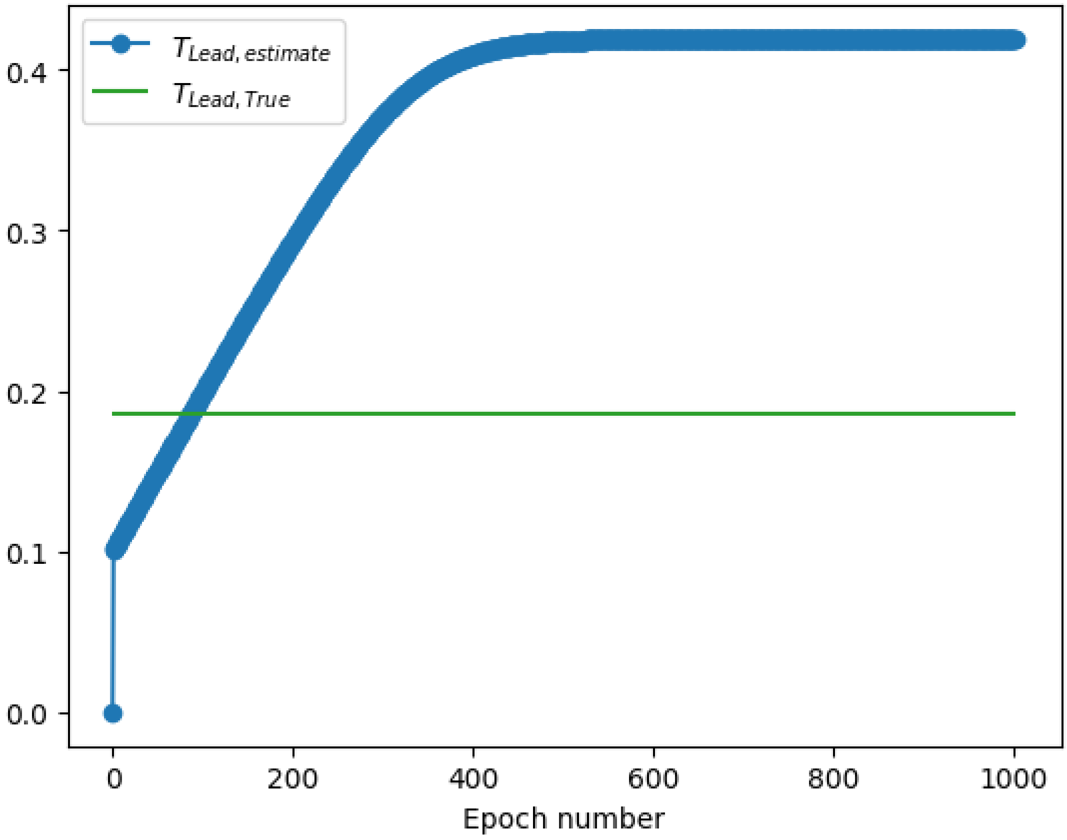

P_{true}
$P_{estimated}$
1.5
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80
100
Time

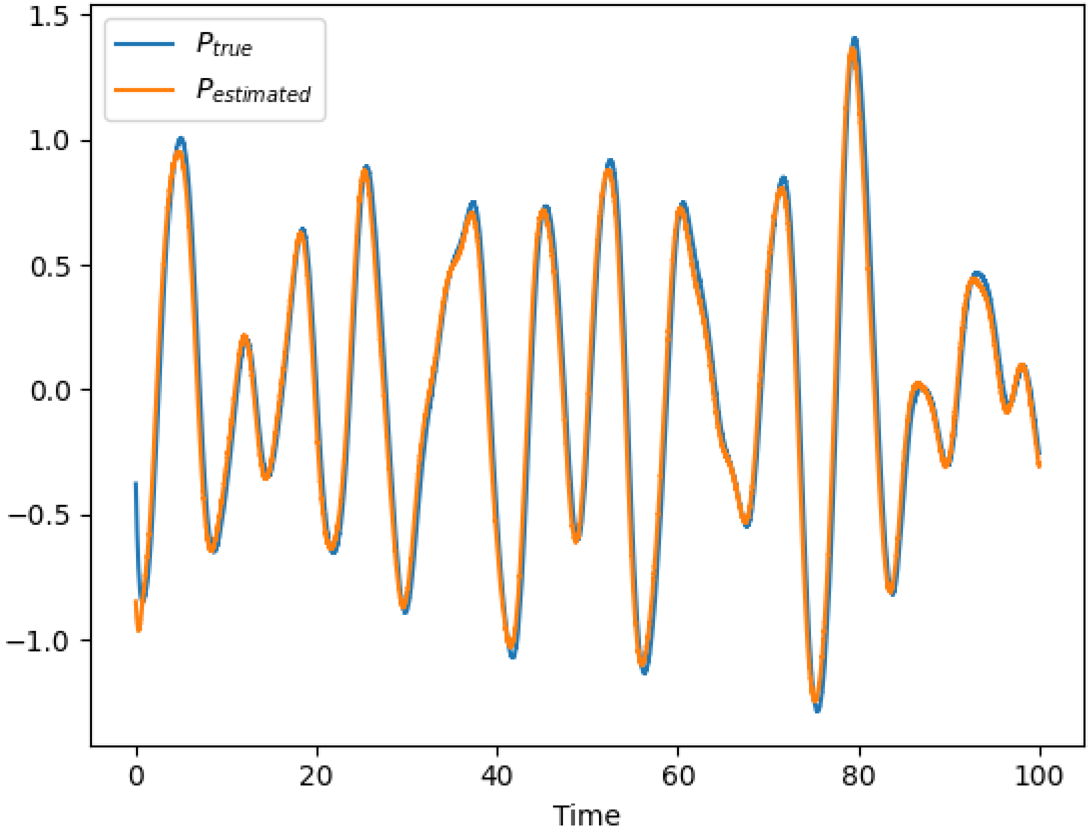
Trial 2:



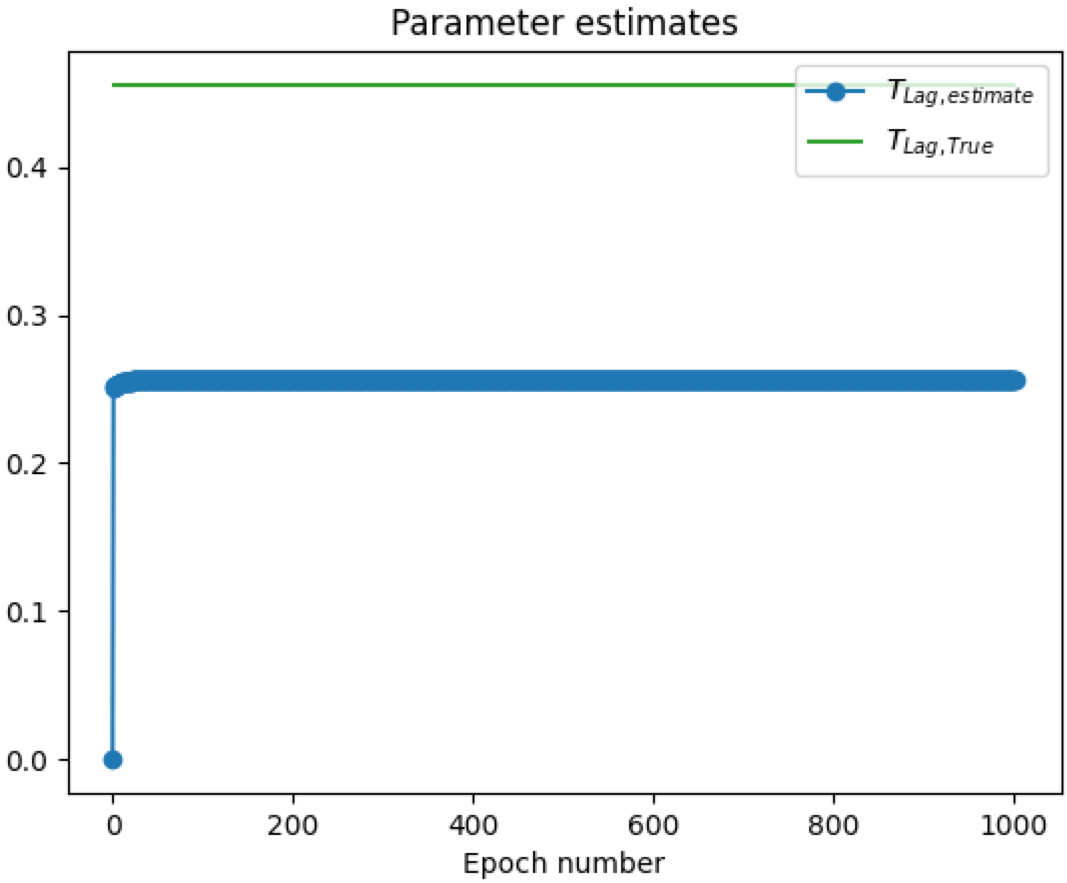
In an attempt to see the algorithm try instead of trying to estimate all 3 parameters at once and run into the issue of non uniqueness in the solution, I instead try and estimate each pilot model parameter 1 at a time (aka set 2 of them to the true value and let the third learn).

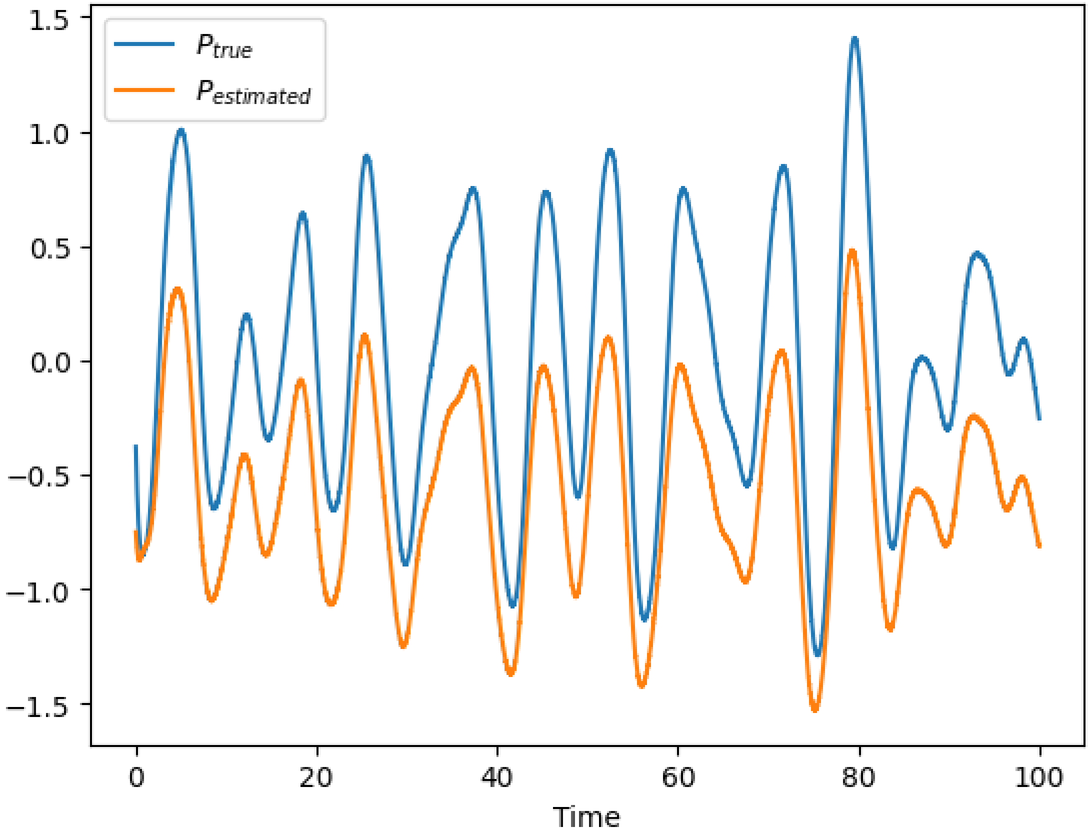
1-Holding T_{Lag}, K_p as true and learning T_{Lead}



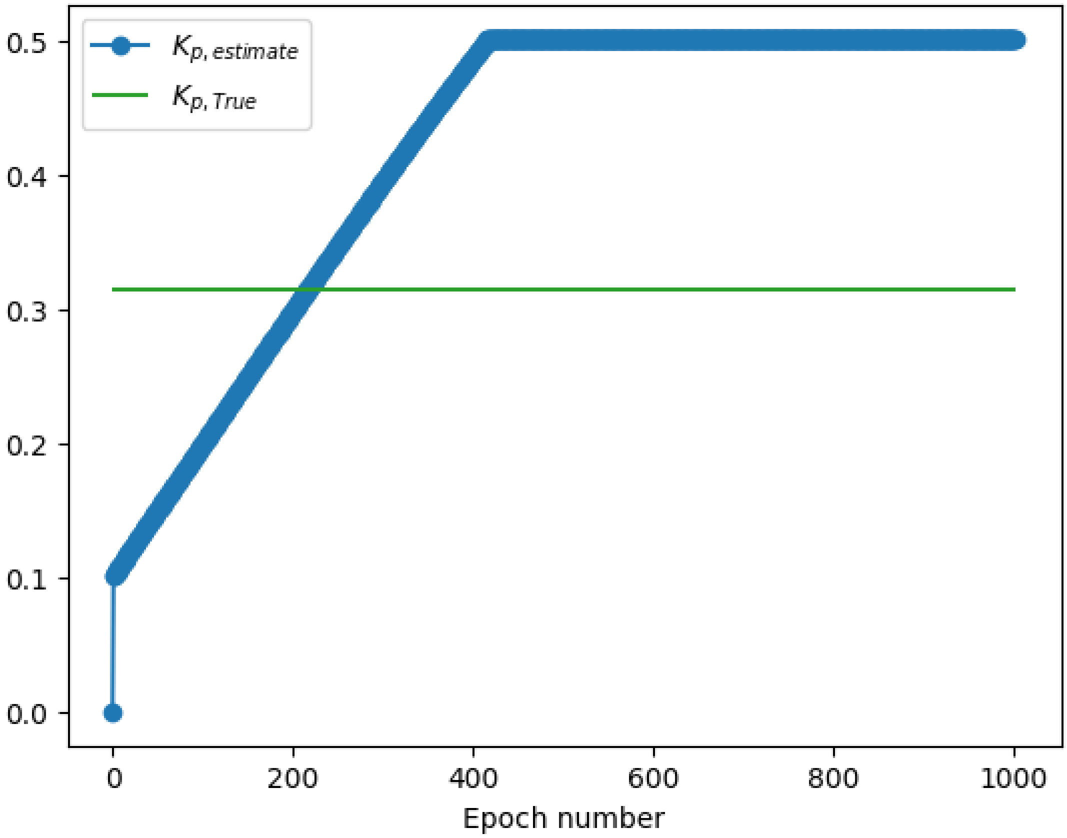


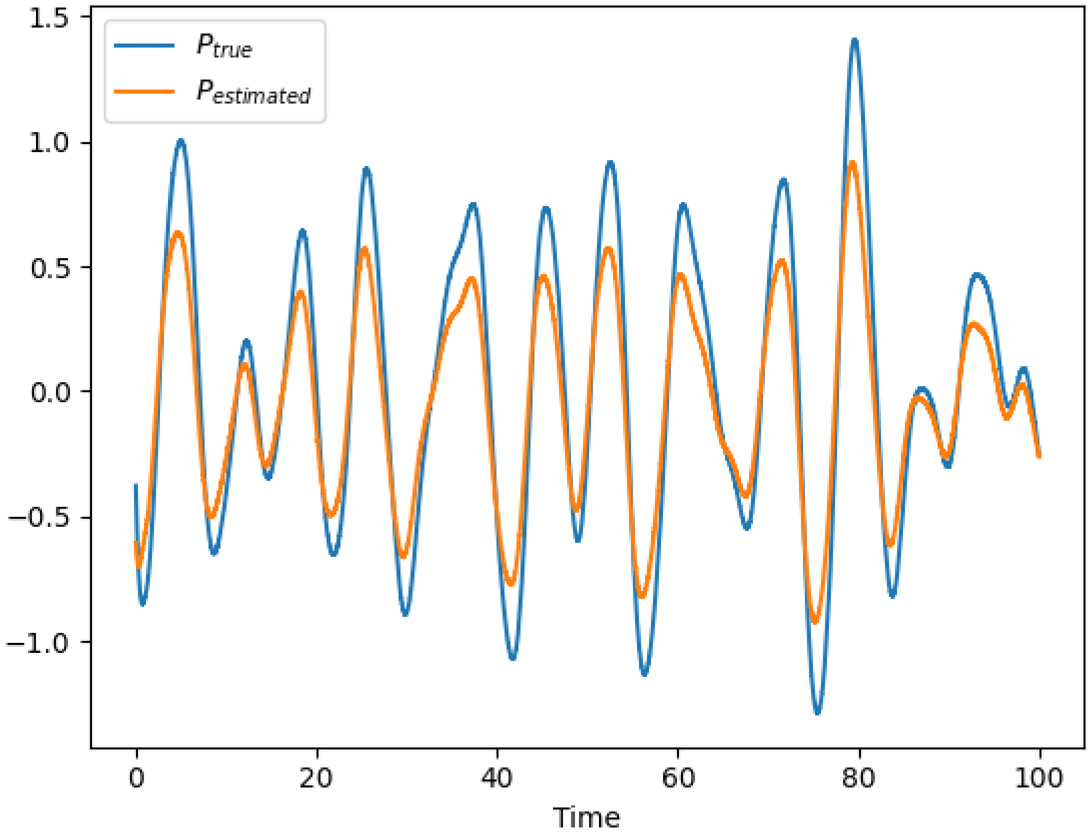
2-Learning T_{Lag}





3-Learning K_p


$K_{p,estimate}$
$K_{p,True}$
Epoch number


P_{true}
$P_{estimated}$
Time