2) General notes on PINNS and IPINNS

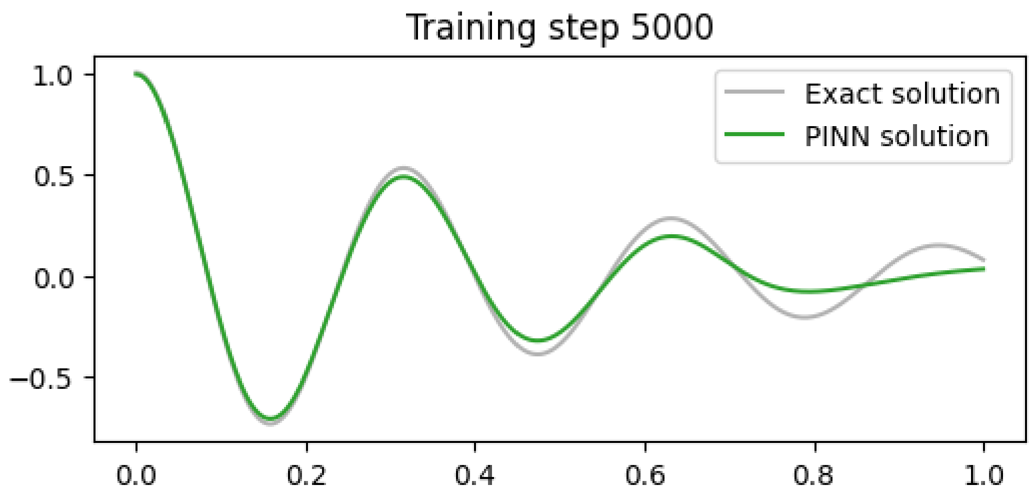
PINNs-
The goal of PINNS is to use Neural Networks to be able to model the solution to a set of differential equations. They accomplish this by embedding the physical differential equations as extra terms to the loss function such as to force the output of the Neural Network not only to match the desired input-output response but for the Neural Network output to also obey the differential equations.

There are terms such as collocation points which are points throughout the domain Neural Network is evaluated at these points and where the differential equations are evaluated. The residuals of the differential equations are summed up and becomes the L_{DE} term in the loss function.

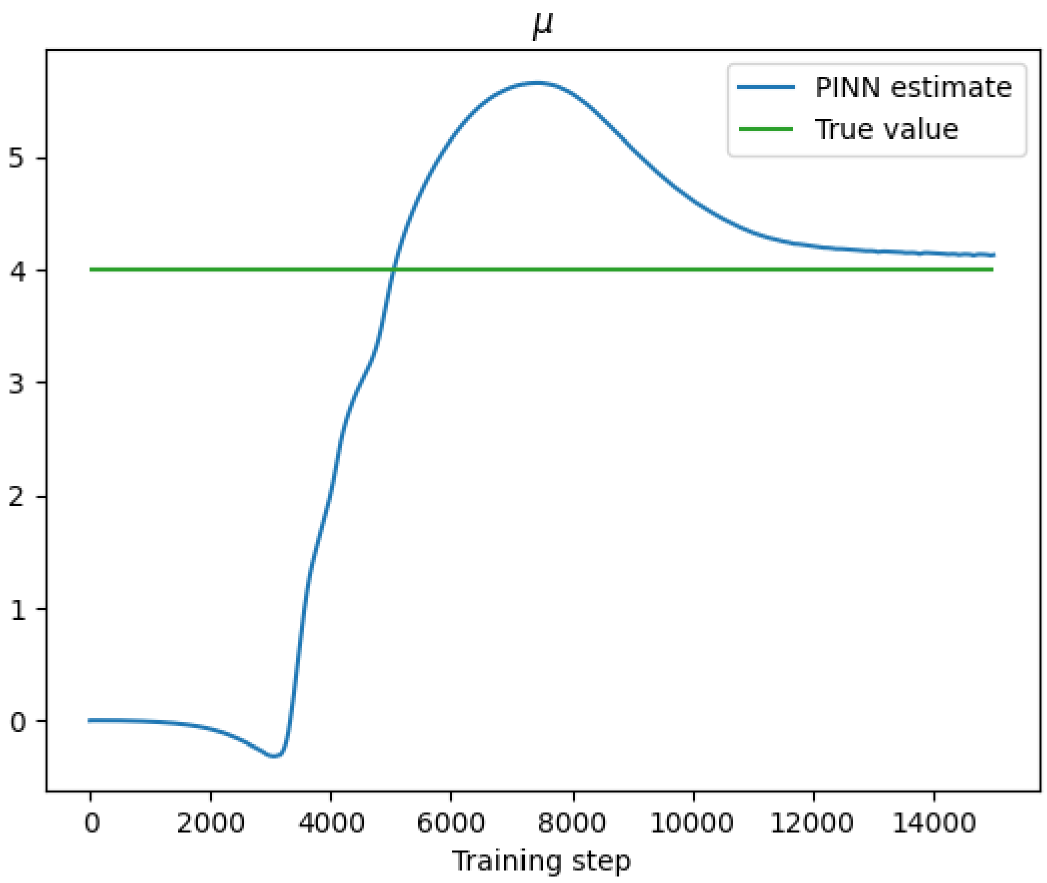
The goal of the PINN (physics informed neural network) is to use the underlying physics equations i.e the differential equations of the system as a part of the loss function to be able to estimate the system response. By defining the differential equations and any other conditions that are known such as $f(0)$ and $f'(0)$, for example it will be able to solve for the response similar to the popular MATLAB function ode45. The PINN will use $f(0)$, $f'(0)$ conditions and the actual physics equations as the loss function for the NN. Note for the PINN the physics coefficients can be defined unlike in the iPINN.

IPINNs-
The goal of the IPINN (inverse physics informed neural network) is the inverse of the PINN where the IPINN will estimate/learn any unknown parameters to the system such as the coefficients in the differential equations, example: the K and b in the spring mass damper equation. This process is done similar to the case of the PINN This is done the same by defining the differential equations and any other conditions such as $f(0)$, $f'(0)$ and and any data terms into the loss function of the NN.

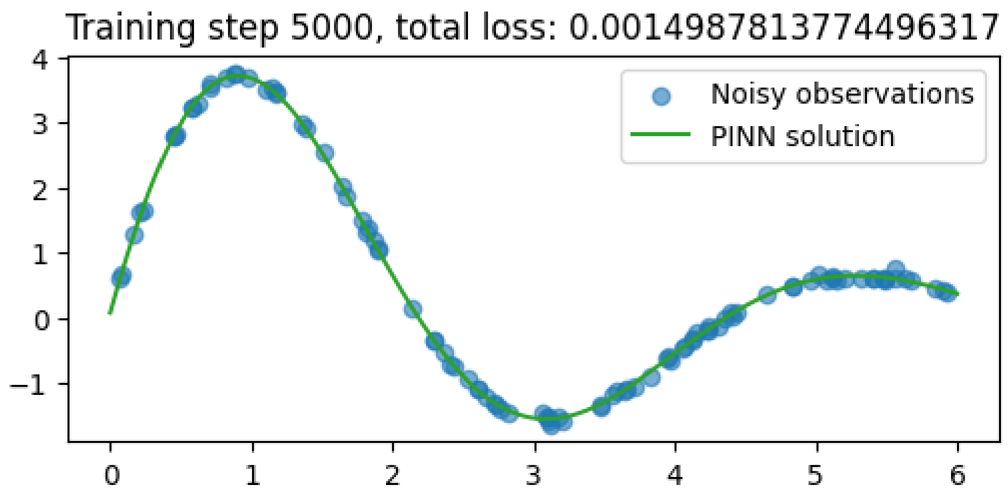
General Results/Limitations-
PINNs: can not really find limitations of the PINN, the solution has shown to consistently converge with minimal error. See example below below for damped harmonic oscillator example after 5000 training steps



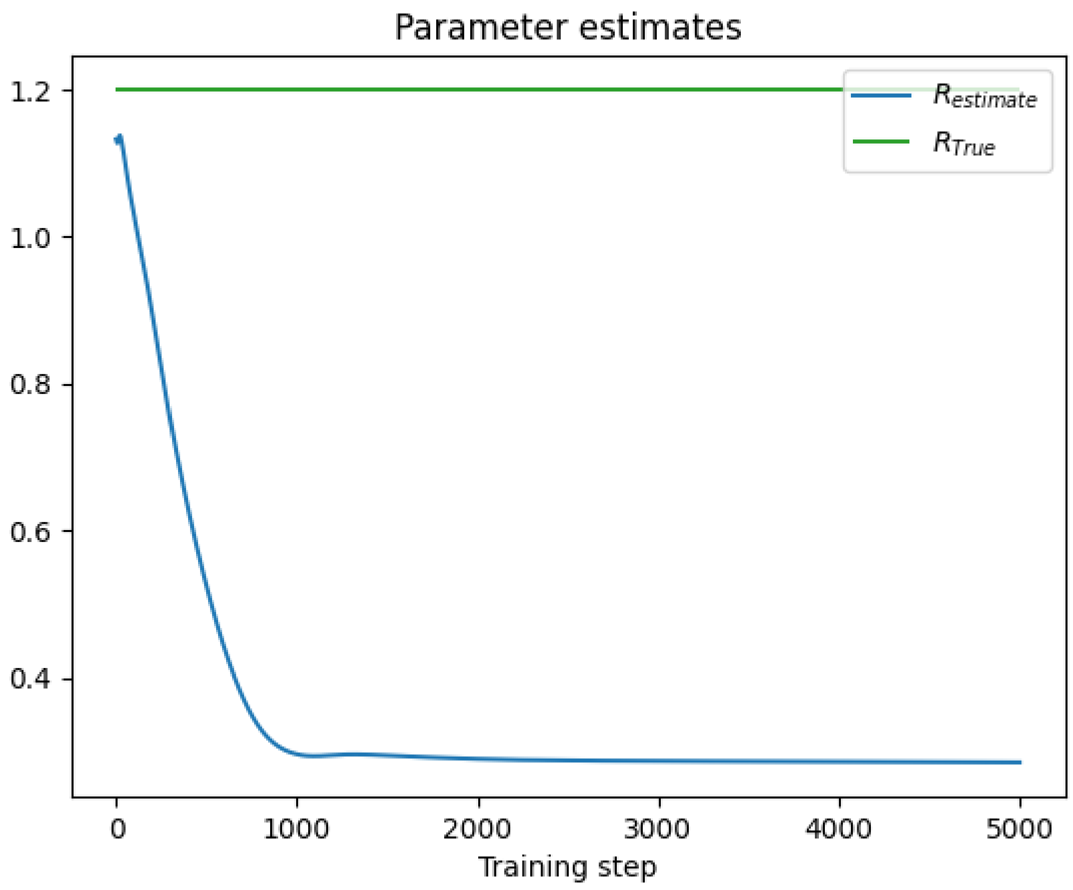
IPINNs: limitations found have been that the output may converge but the parameters may not. For example for a system with 1 parameter the system output and parameter can converge with little error. See example below from damped harmonic oscillator,

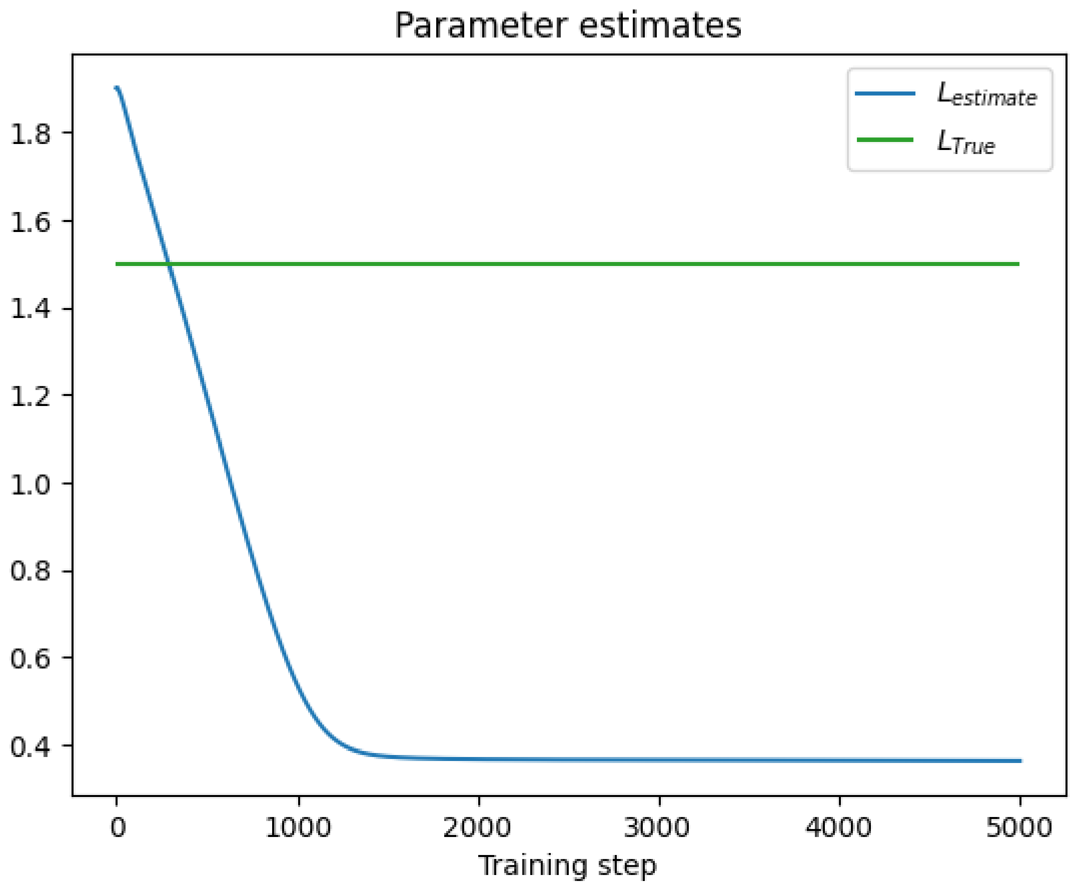


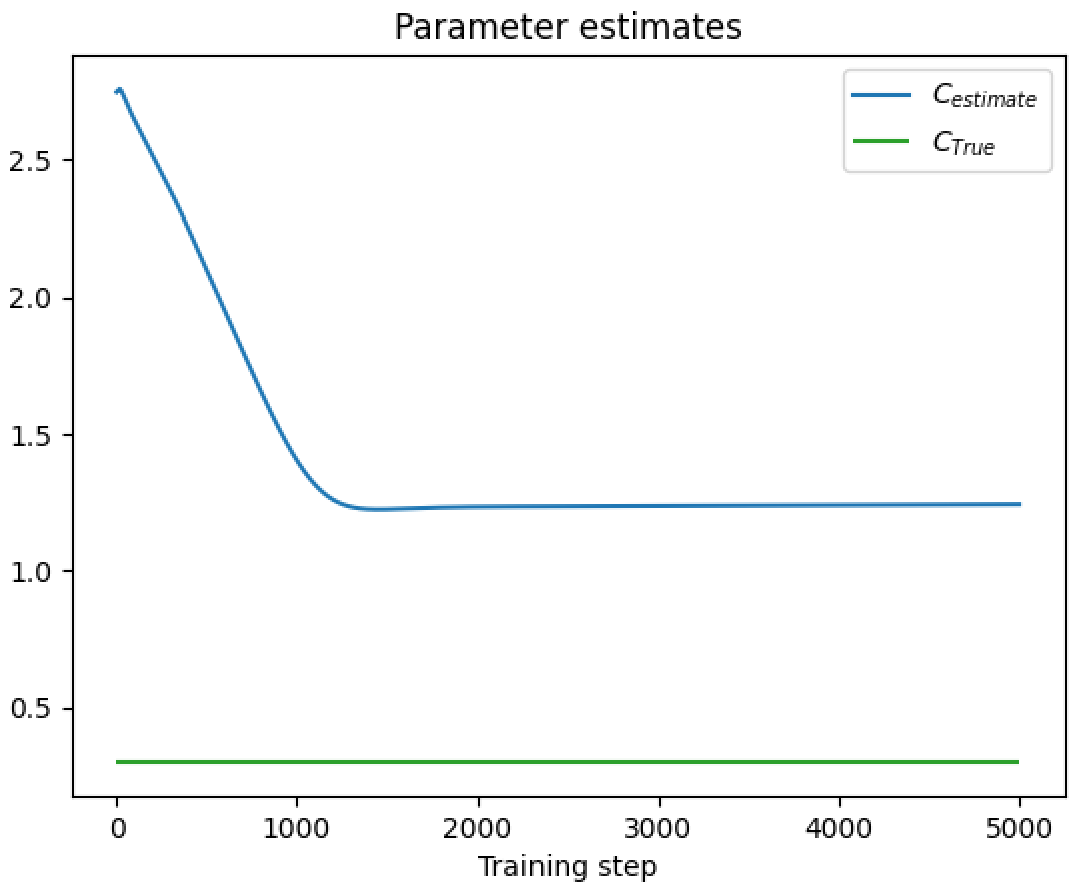
But with multiple parameters the output has been shown to converge with little error but the parameters may not. This can maybe be due to how there could be multiple solutions to the output (or multiple solutions to the output for the example done so far) so many different configurations of the parameters satisfy the differential equations/initial conditions. See example below IPINN to estimate parameters for an RLC circuit while also trying to estimate the solution for an RLC circuit where we are estimating R, L, C,


Training step 5000, total loss: 0.0014987813774496317
Noisy observations
PINN solution
4
3
2
1
0
−1
0
1
2
3
4
5
6

xaxis=time


Parameter estimates
$R_{estimate}$
R_{True}
1.2
1.0
0.8
0.6
0.4
0
1000
2000
3000
4000
5000
Training step


Parameter estimates
$L_{estimate}$
L_{True}
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
1000
2000
3000
4000
5000
Training step



The parameter estimates potentially flatten around training step 1000 because around step 1000 the loss function is so low that not much can be backpropagated to adjust the weights to generate a new estimate.

Main drawbacks of PINNS/IPINNS in general is that the way they are setup in the code as well as seen in the literature, they are solved for and solved for a specific case/a specific set of parameters or set of initial conditions and this may be good for solving for complex set of equations like Navier Stokes, Nonlinear Schrodinger, etc. but this will lack large learning capabilities and generality when it comes to for example modeling different pilots characteristics.