

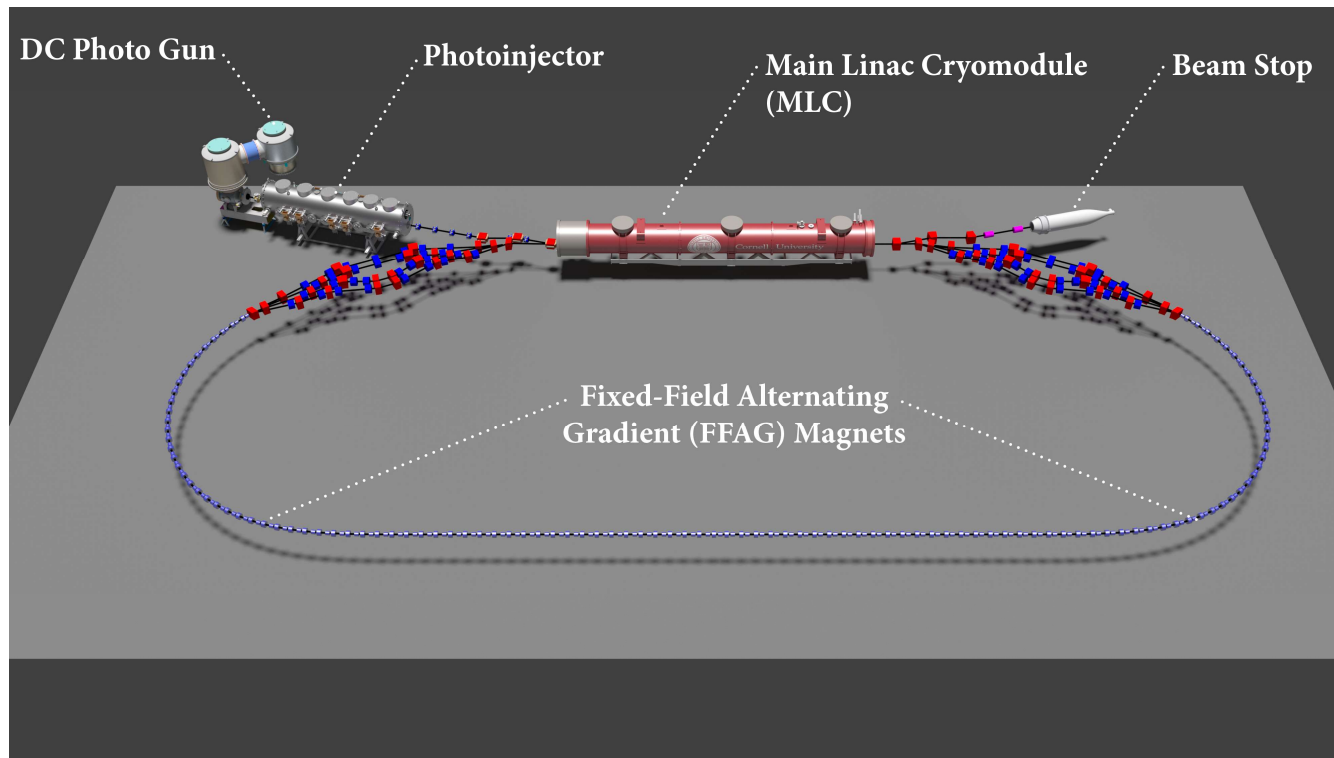
Controlling CBETA with ML

Jason Hua

The problem we are trying to solve

- The CBETA accelerator requires the beam initially injected into the accelerator to have specific phase space distribution.
- This is because CBETA wants to do energy recovery and wants to loose as little particle as possible.
- This means an accelerator controller has to tune the accelerator by dialing various knobs to a very specific state, i.e. the twiss parameters are chosen to particular values.

The problem we are trying to solve



The problem we are trying to solve



Twiss Parameters

A good approximation for the beam shape in phase space is an ellipse. Any ellipse can be defined by specifying:

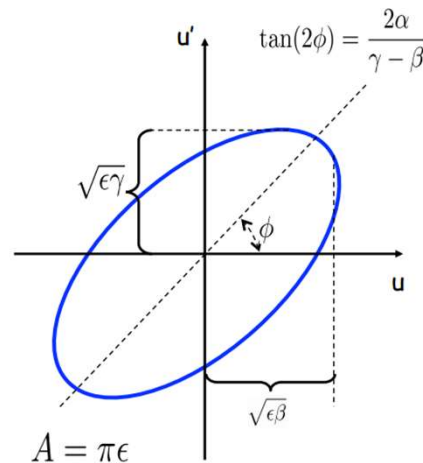
- ✓ Area
- ✓ Shape
- ✓ Orientation

We choose 4 parameters -
3 independent, 1 dependent:

α - related to beam tilt
 β - related to beam shape and size
 ϵ - related to beam size
 γ - dependent on α and β .

These are the "Twiss Parameters" (or "Courant-Snyder Parameters")

Beam Ellipse in Phase Space:



The strategy to solve the problem

- High-fidelity surrogate model.
- Wrap this model inside a minimization algorithm.
- To minimize the distance between the desired twiss parameters and the surrogate model's evaluation.

The strategy to solve the problem

- Why do we need the surrogate model and not just use GPT?
- Given a random starting point in the 6-dimensional space, we need between 10/20 to 1000+ runs of GPT evaluations.
- GPT simulation takes too much time and we want to use this algorithm to control in real life with a push of a button.
- We approach this problem by postulating the conjecture that we can replace the high-fidelity model with a machine learning model.

How do we build the ML-based model?

- First, we clean the data
- Then, we need to select the model. This means that we need to find the correct model complexity. We need a model that is expressive enough to account the complexity of an accelerator yet not so large that we overfit.
- Finally, since we are building this ML-based model as a surrogate model, we need to validate the accuracy of this model.

Data Preprocessing

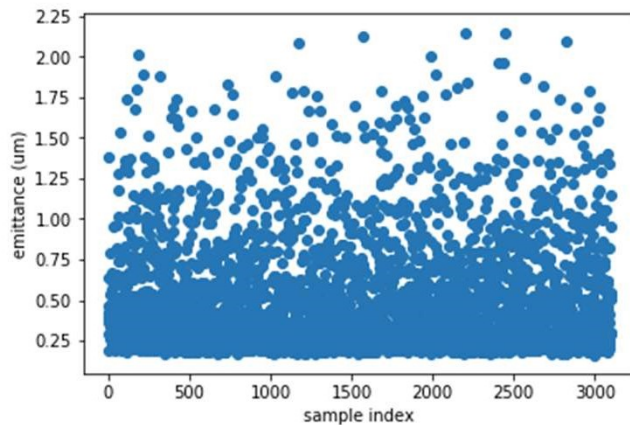
- The nominal operating range for the 6 input parameters.

Variable Name	Minimum Vale	Maximum Vale
Solenoid 1 current	-3.5 A	-2.5 A
Solenoid 2 current	1.8 A	2.8 A
A3 quadrupole 1 current	-8 A	-6 A
A3 quadrupole 2 current	4.5 A	6.5 A
A3 quadrupole 3 current	2.6 A	4.6 A
A3 Quadrupole 4 current	-9.3 A	-7.3 A

- We use those nominal operating range to generate data.

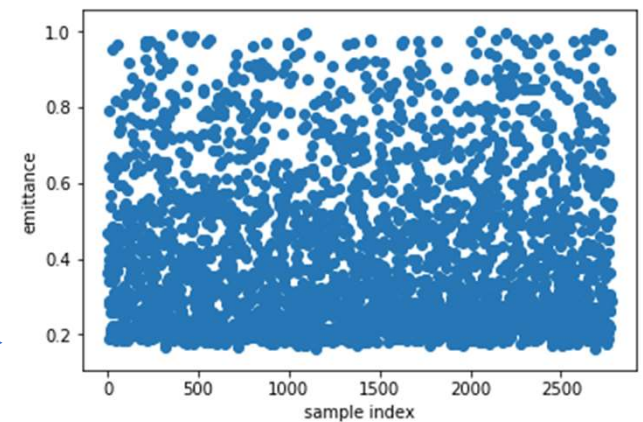
Data Preprocessing

- We don't want the final emittance to be larger than 1 μm .



We cut training data with emittance larger than 1 μm .

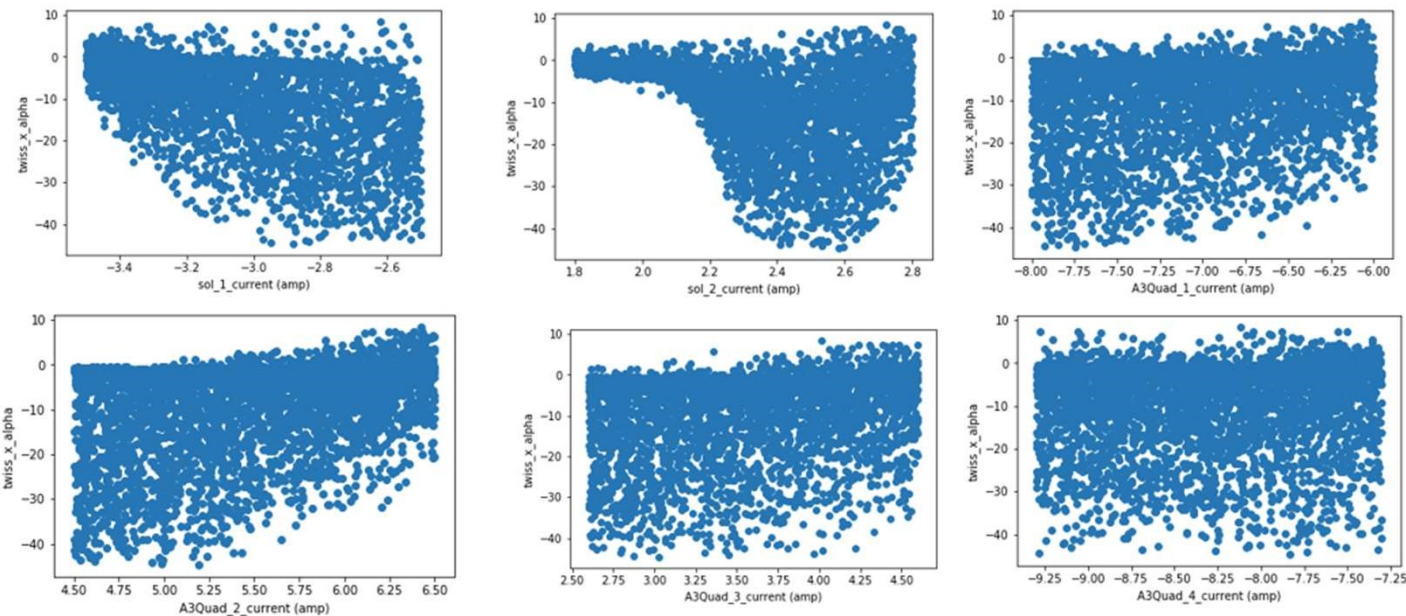
After the cut, we have 2777 data.



- We scale all the input/output data to be between $[-1,1]$

Visualize the data

- Plotting the twiss parameter alpha, α , against all of the 6 varying parameters.



1. If we look at a single graph and pick an independent value, we see that the parameter can vary a lot.
2. There are sparse regions and dense regions.

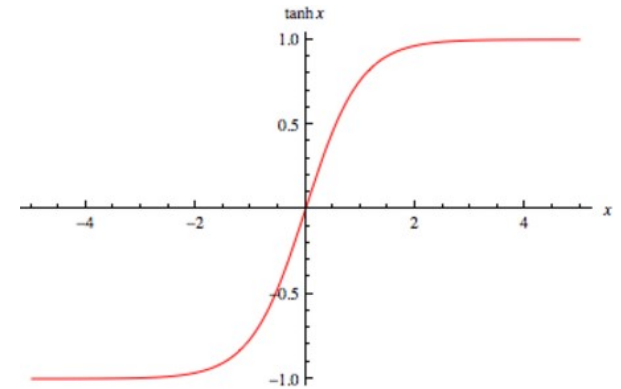
Model Selection

- Based on a rough inspection of the distribution of the data, we probably need a model that can approximate complex functions.
- Fully connected multilayer perceptron with the following architecture.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 64)	448
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 32)	2080
dense_4 (Dense)	(None, 32)	1056
dense_5 (Dense)	(None, 16)	528
dense_6 (Dense)	(None, 16)	272
dense_7 (Dense)	(None, 4)	68
Total params: 8,612		
Trainable params: 8,612		
Non-trainable params: 0		

Model Selection

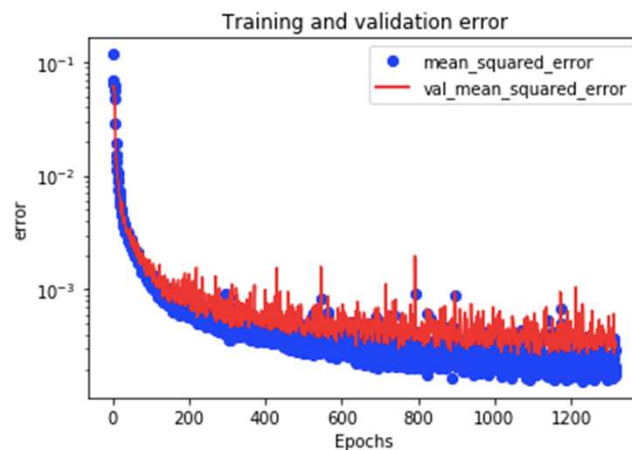
- We use hyperbolic tangent activation function.
- We use no regularization penalties.
- During the training process, we use the Adam optimization algorithm.
- We use early stopping to ensure no overfitting.
- The loss function is mean squared error



$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$


Training the model

- There are 2777 data points. We use 2429 ($7/8$) as training data points and use 348 ($1/8$) as validation data points.



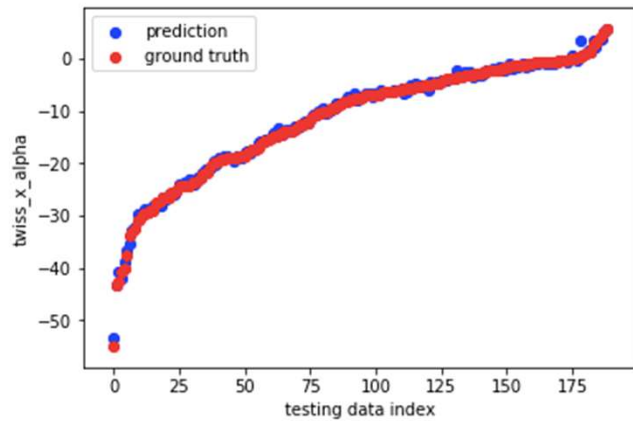
Check model fidelity

- We use GPT to generate a new set of data to test the fidelity of the ML-based model.
- We measure the difference between ground truth, which are twiss parameters calculated by GPT, and the ML prediction by mean absolute error.

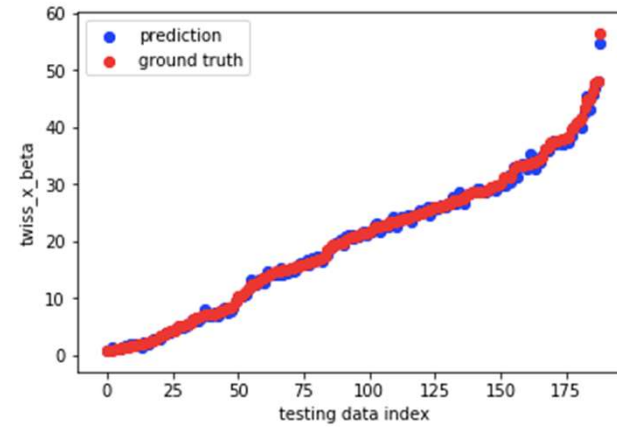

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

©easycalculation.com

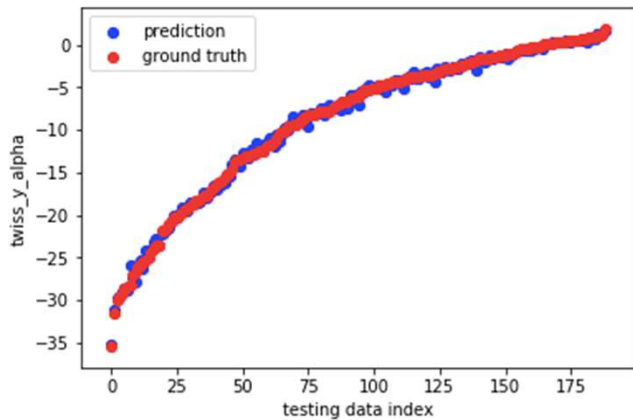
Check model fidelity



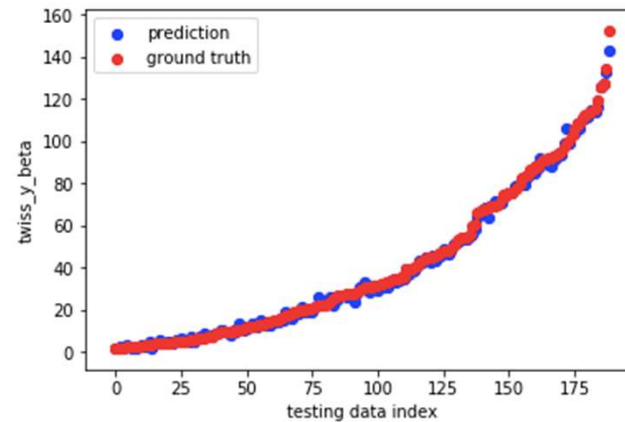
MAE:0.3747



MAE:0.3061



MAE:0.3966



MAE:1.1677

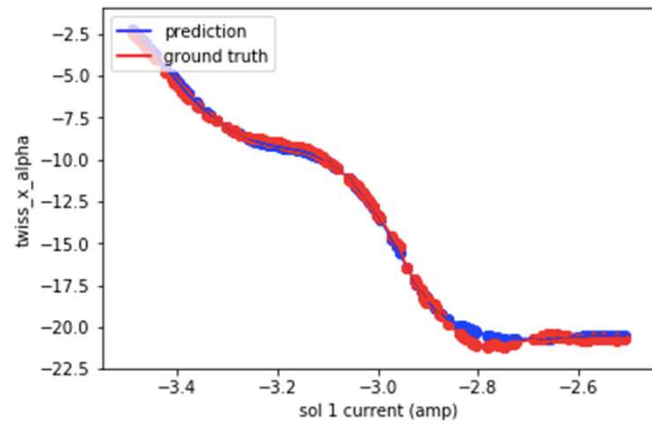
Check model fidelity

- Since the ML-based model is inside a minimization
- We want to make sure the ML-based model has the capability to numerically compute Jacobi or do descent along a direction.

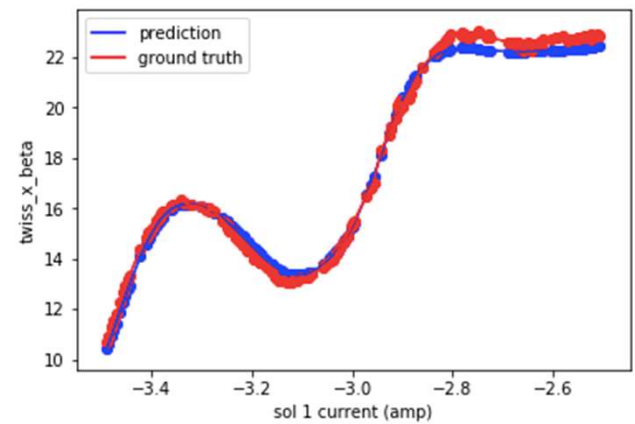
Check model fidelity

- We generated 6 more testing data sets, where for each dataset only one of the 6 input parameters can vary.
- For each testing data sets, we compare 4 output parameters.
- When comparing, we use the MAE as the metric.

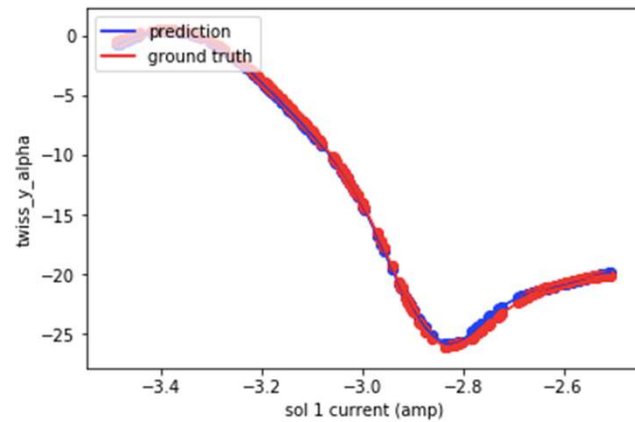
Check model fidelity



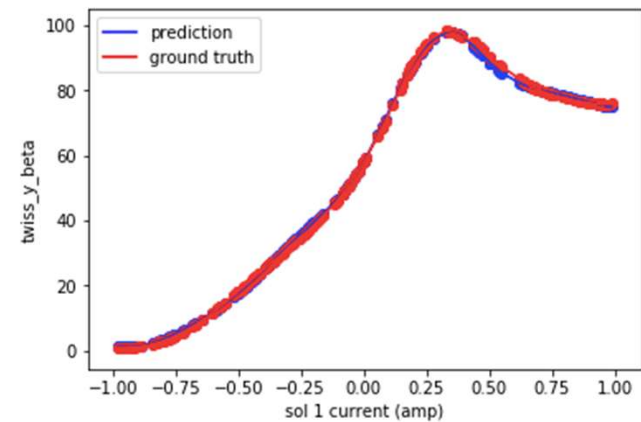
MAE: 0.1913



MAE:0.2681



MAE: 0.1681



MAE:0.4411

Minimization

- We pick a random point in the 6-D input parameter space.
- Then, we ask the ML-based model to simulate what the twiss parameters are associated with this input set.
- Then, we compute a loss function between the twiss parameters simulated by the ML-based model and what the accelerator operator desires. The loss function is the squared error.
- Lastly, we use a truncated newton minimization scheme to find the next point in the 6-D input parameter space.
- Repeat this process, until converge. The convergence criteria is

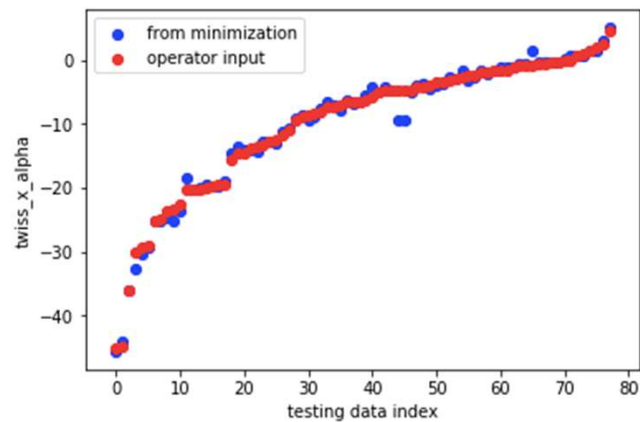
Minimization

- We start the minimization process with 30 different random starting points.
- In the end, we will only return one set of parameters that has the smallest squared error amount the 30 potential answers.

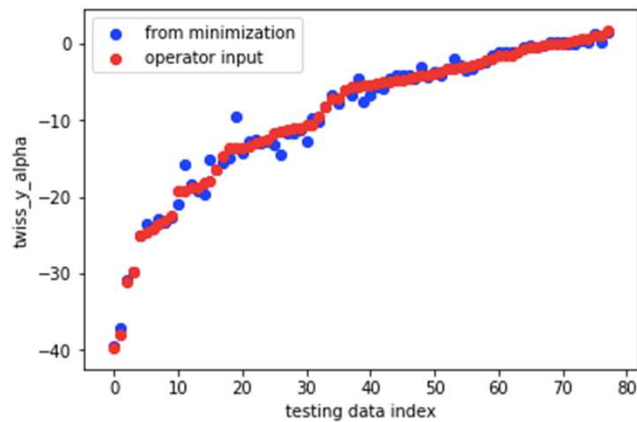
Check that we can operate CBETA with ML

- Because it's likely that for a particular set of twiss parameters, there are more than one solution.
- This means that given 6 input parameters and then ask GPT to simulate twiss parameters based on those inputs and then put the resultant twiss parameters back in the minimization may not return the same 6 input parameters that we started with.
- We need to check the result of the minimization by putting the result back into GPT.

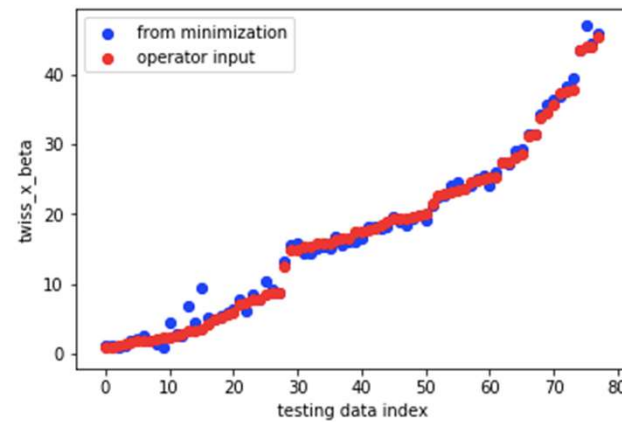
Check that we can operate CBETA with ML



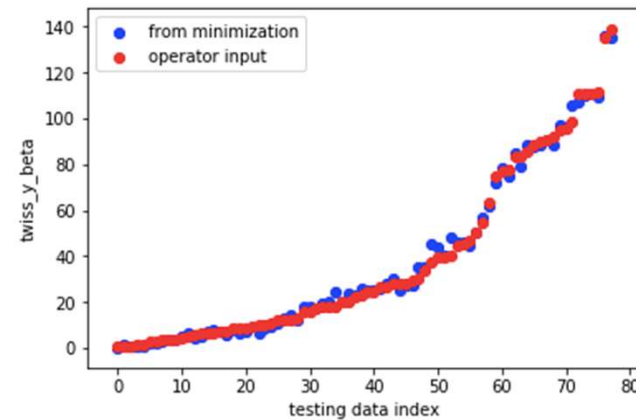
MAE: 0.2485



MAE: 0.7267



MAE: 0.6134



MAE: 0.6688