Progress Report

Pranav Bangarbale - 12/21/2021

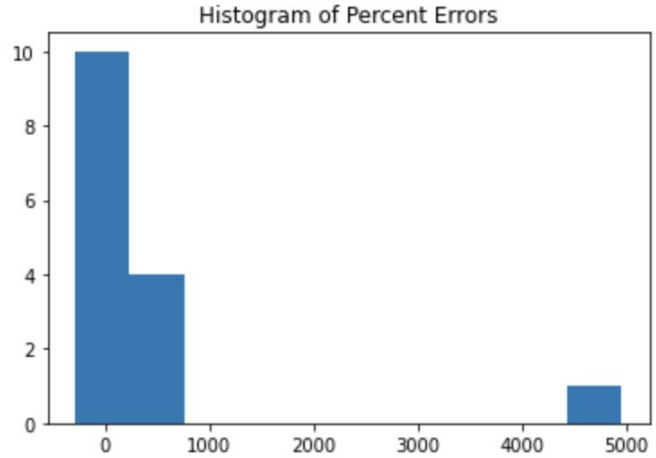
Current Task

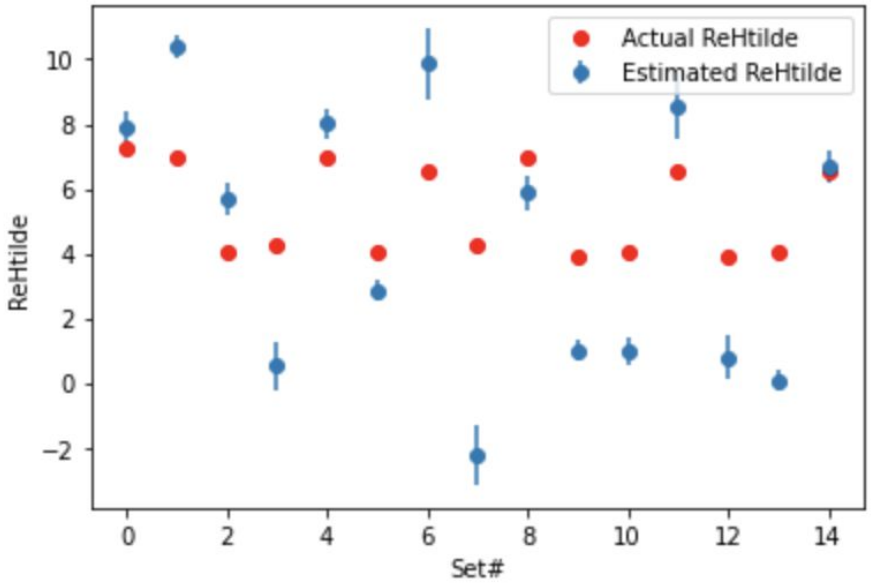
- Have been working on implementing K-means algorithm in order to do intermediate weight averaging
- Implementation has been completed; results have been generated

```
# Fit kmeans clustering to group of weights from set replicas
w = np.array(intermediate_models)
nsamples, nx, ny = w.shape
w1 = w.reshape((nsamples,nx*ny))
kmeans = KMeans(n_clusters=5, random_state=0).fit(w1)
cluster_to_models = {}
for model in set_models:
    input = []
    for x in model[2]:
      for y in x:
        input.append(y)
    input = [input]
    closest = kmeans.predict(np.array(input))[0]
    # print(closest)
    # i = np.where(kmeans.cluster_centers_==closest)
    # print(i)
    if closest not in cluster_to_models:
      cluster_to_models[closest] = list()
    cluster_to_models[closest].append(model)
# have each model for set and which cluster each model belongs to
# want to average weights for models of the same cluster and append these cff values to by_set
for c in cluster_to_models:
  group = cluster_to_models[c] # group of models for a given cluster (given by weights)
  avg_weights = np.mean(group, axis=0) # element-wise weight averaging
  globalModel.set_weights(avg_weights)
  globalModel.fit([setI.Kinematics, setI.XnoCFF], setI.sampleY(), # the sample Y will generate Fs
                epochs=100, verbose=0,
                callbacks=[model_checkpoint_callback])
  globalModel.load_weights(chkpt_path) # load back minimum loss epoch
  cffs = uts.cffs_from_globalModel(globalModel, setI.Kinematics, numHL=NUM_HIDDEN_LAYERS) # get cffs from middle model
  by_set.append(cffs)
```

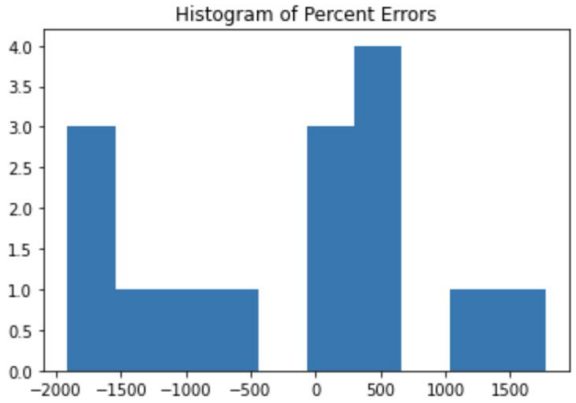
Results (part 1)

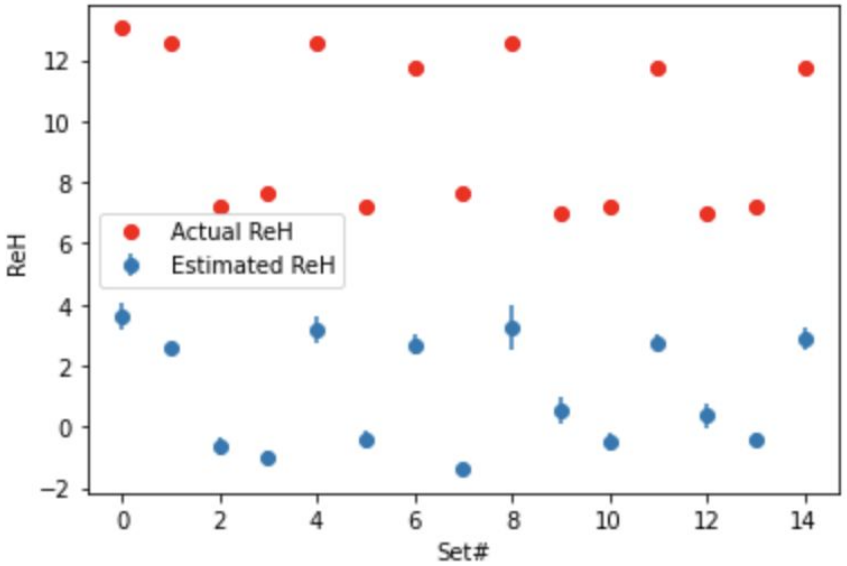
```
Mean percent error:  477.78184628367933
RMSE:  2.976540931551008
RMSE w yhat=mean:  1.4030345621243816
R-squared:  -3.5007716491044656
```



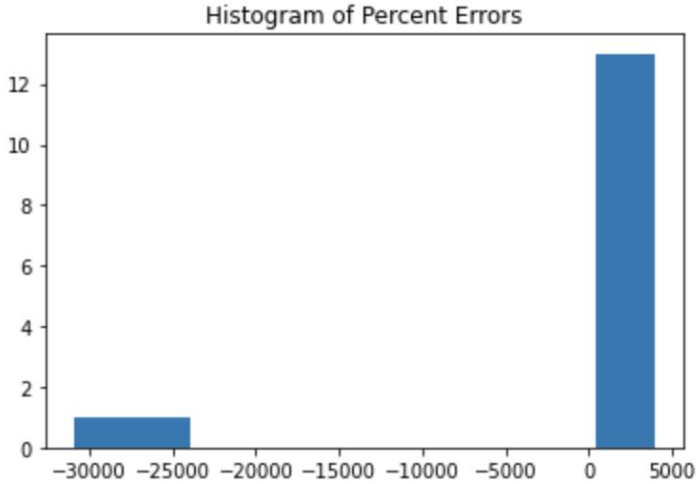


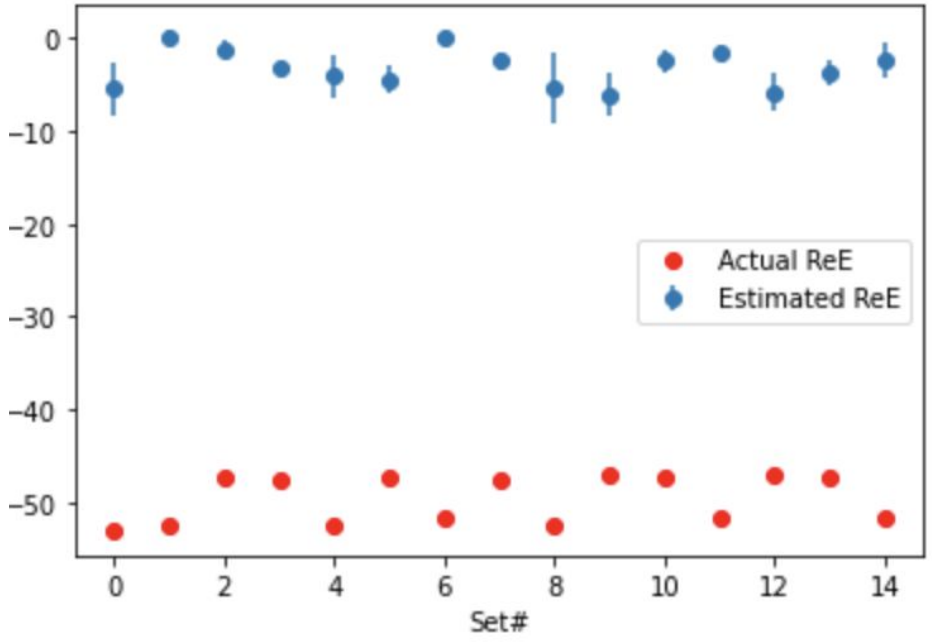
```
Mean percent error:  886.5477930612573
RMSE:  8.49592975595536
RMSE w yhat=mean:  2.5254628436780107
R-squared:  -10.317222635225521
```



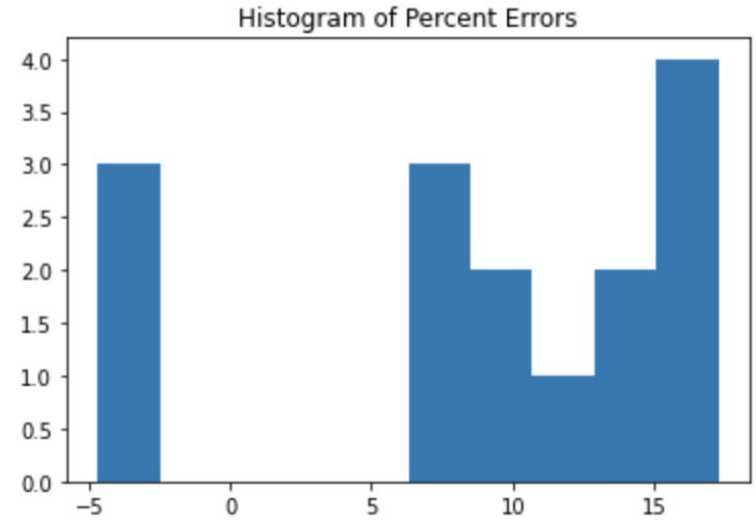


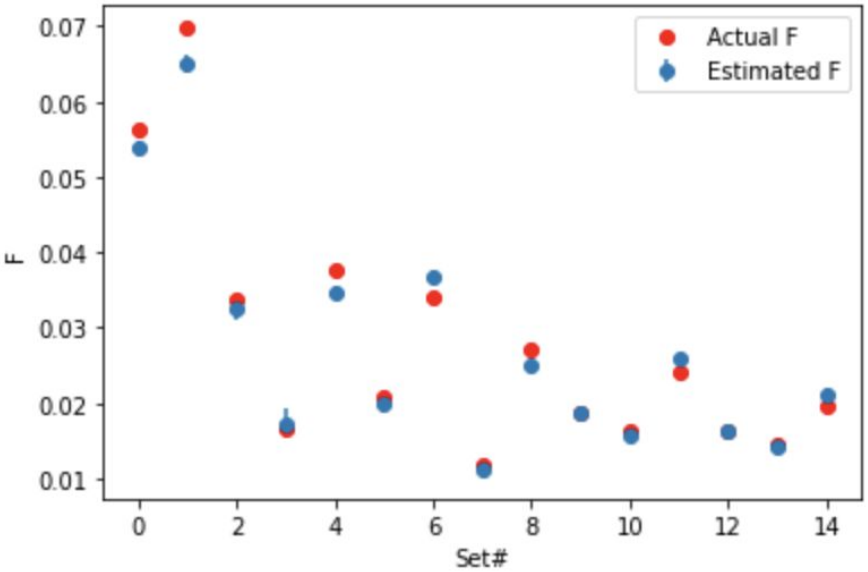
```
Mean percent error:  5167.420371605129
RMSE:  46.58255438657015
RMSE w yhat=mean:  2.525480507907797
R-squared:  -339.2189951795507
```





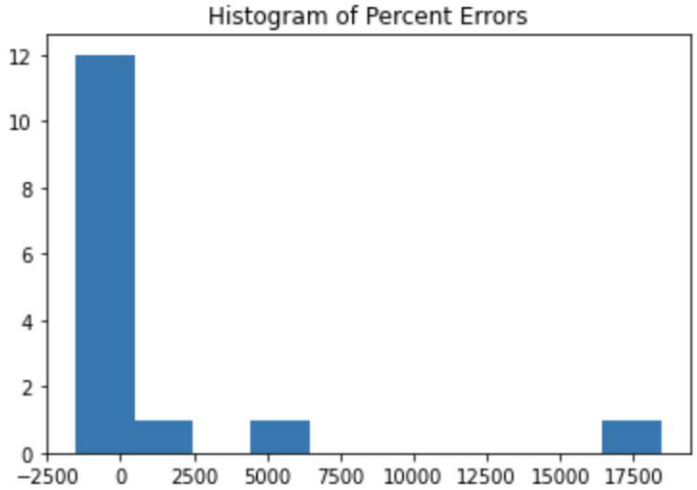
```
Mean percent error:  10.533317631106346
RMSE:  0.003918341964841446
RMSE w yhat=mean:  0.015855649879790133
R-squared:  0.9389287827470483
```

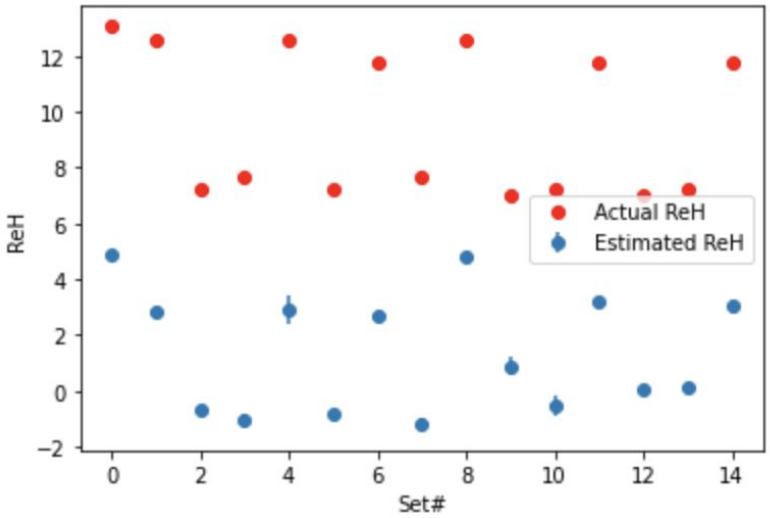




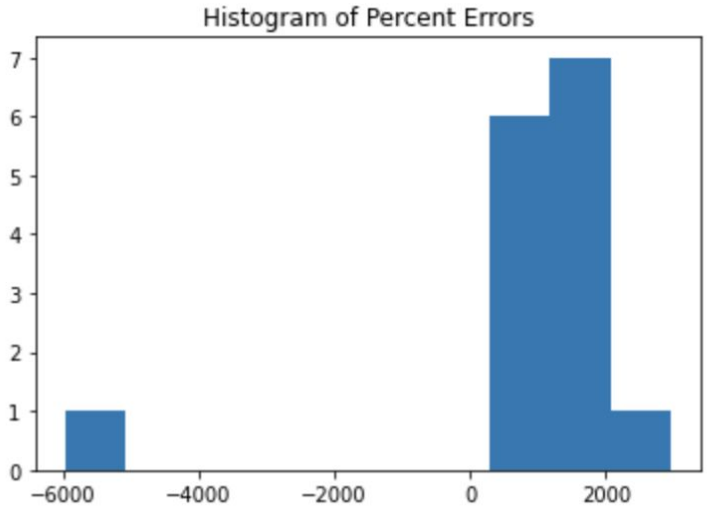
Results (part 2)

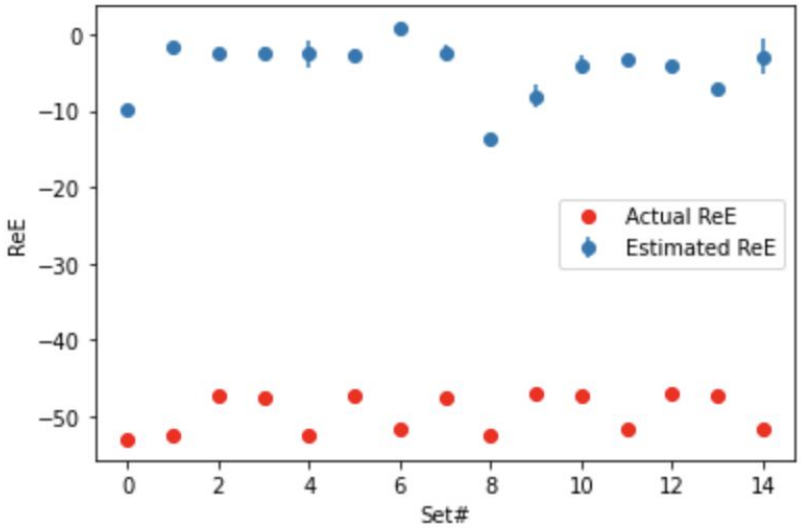
```
Mean percent error:  2156.0630772409563
RMSE:  8.2486403045738
RMSE w yhat=mean:  2.5254628436780107
R-squared:  -9.667994061785775
```



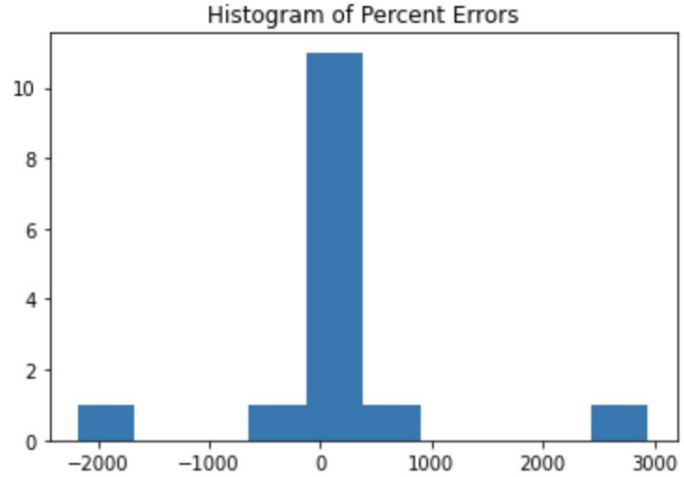


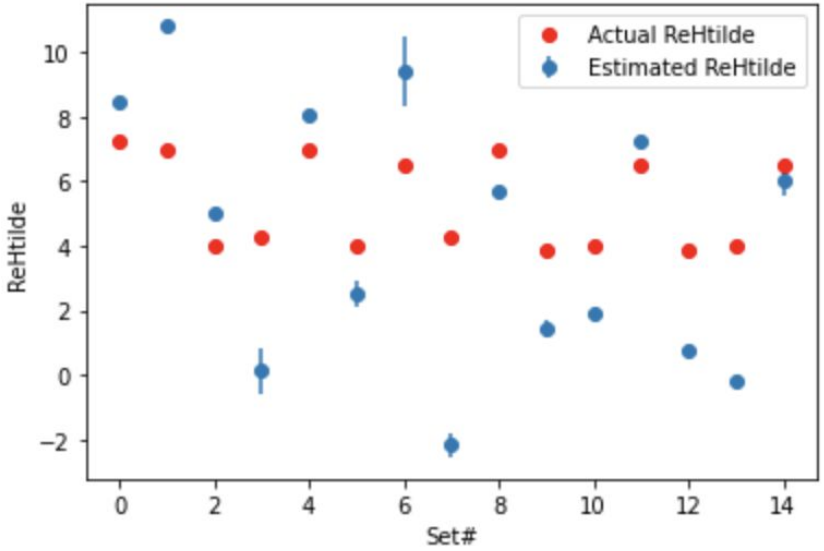
```
Mean percent error:  1679.9991196579774
RMSE:  45.3661428441653
RMSE w yhat=mean:  2.525480507907797
R-squared:  -321.68269095984874
```



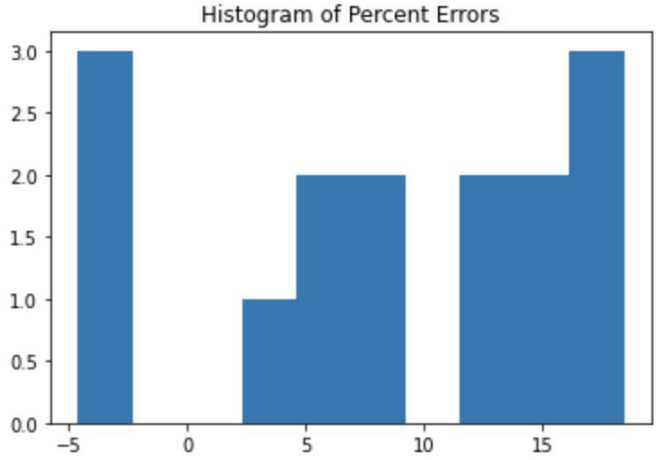


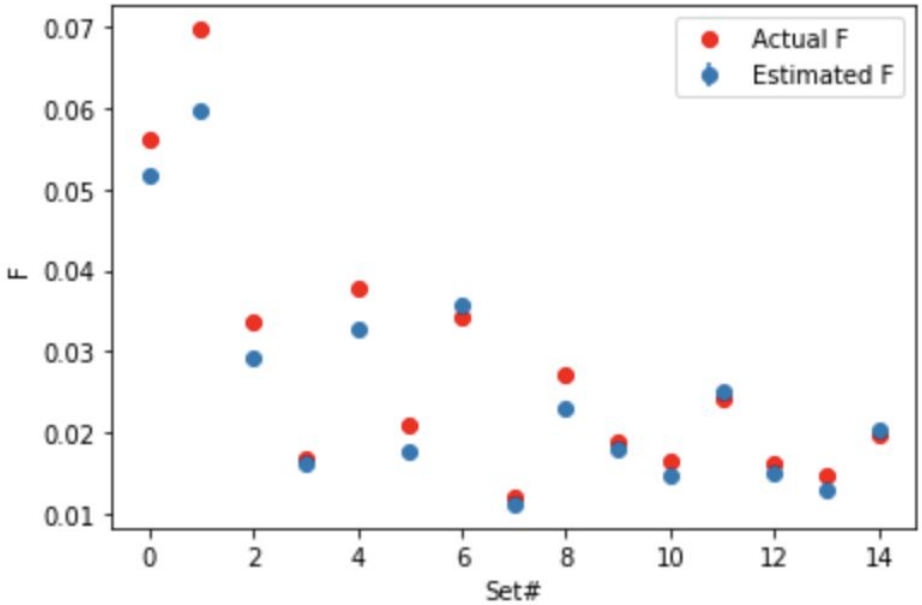
Mean percent error: 423.2346910967616
RMSE: 2.91907201445781
RMSE w yhat=mean: 1.4030345621243816
R-squared: -3.3286540662181823





```
Mean percent error:  10.217115631864447
RMSE:  0.0037381169089202248
RMSE w yhat=mean:  0.015855649879790133
R-squared:  0.9444175526949976
```





Summary

- For both methods, the fit on the CFFs was especially poor, while the fit on F was very good
- Perhaps after a limited number of epochs, the weights are not trained well enough to be averaged (currently 400/100 split)
- Experimentation with different epoch amounts is necessary, as well as different numbers of replicas and clusters created from each amount of replicas
- A small number of replicas was used, which could be the reason for k-means not generating an optimal result