Research Meeting Update

Aaryan

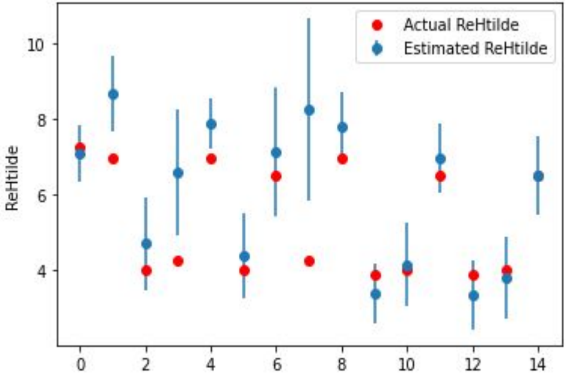
Experimenting with different loss functions

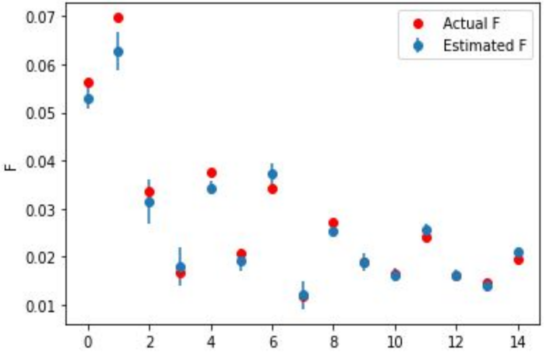
- Why are we getting good fits for F but not CFFs?
  - NN doesn’t care about CFFs only about fitting for F

- What can the loss function be to better fit for the actual CFF
  - F = dvcs + BHUU + IUU
    - IUU is the only component dependant on CFFs

- MSE: F_{act}-F_{est} = IUU_{act} - IUU_{est} → Should get similar results

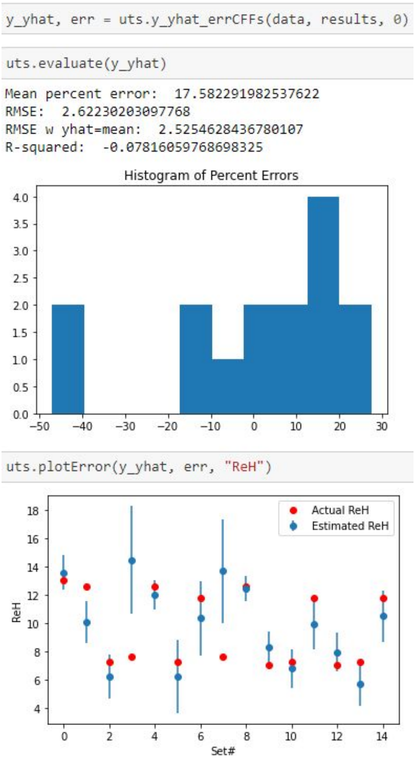
- MAPE: ΔF/F ≠ ΔIUU/IUU





Control - MSE with F loss

ReH



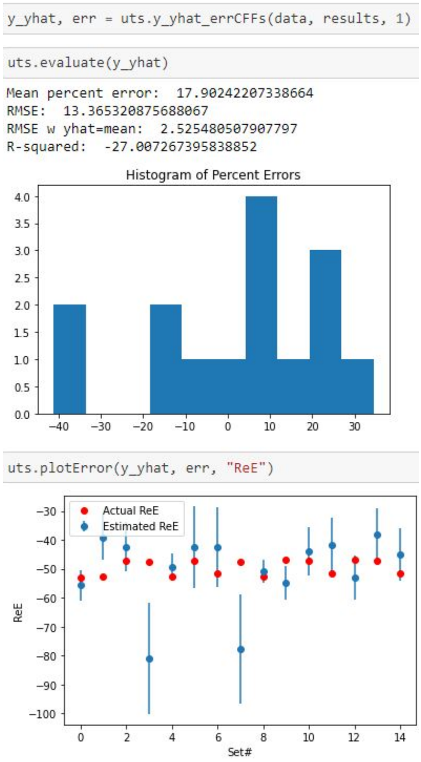
```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 0)
```

```
uts.evaluate(y_yhat)
```

```
Mean percent error:  17.582291982537622
RMSE:  2.62230203097768
RMSE w yhat=mean:  2.525462843678O107
R-squared:  -0.07816059768698325
```

```
uts.plotError(y_yhat, err, "ReH")
```

ReE



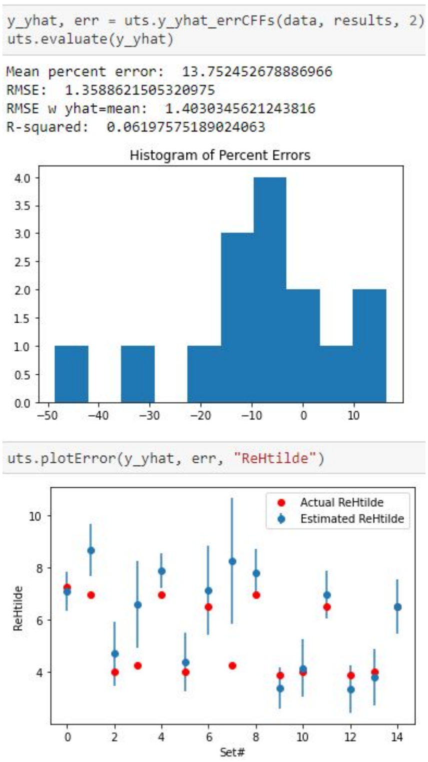
```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 1)
```

```
uts.evaluate(y_yhat)
```

```
Mean percent error:  17.90242207338664
RMSE:  13.365320875688067
RMSE w yhat=mean:  2.525480507907797
R-squared:  -27.007267395838852
```

```
uts.plotError(y_yhat, err, "ReE")
```

ReHtilde

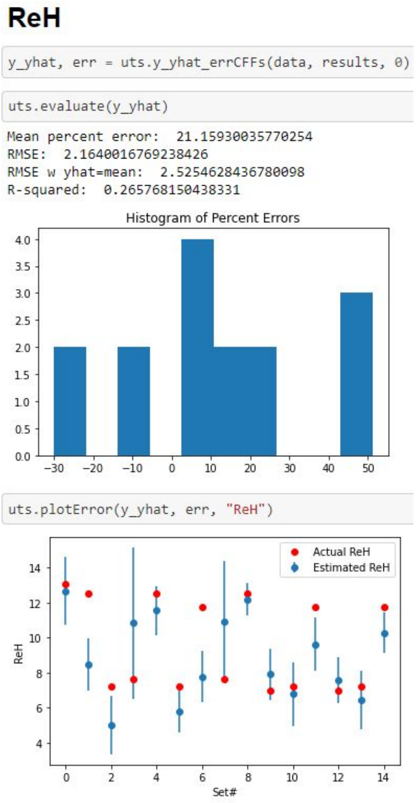


```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 2)
uts.evaluate(y_yhat)
```

```
Mean percent error:  13.752452678886966
RMSE:  1.358862150532O975
RMSE w yhat=mean:  1.4030345621243816
R-squared:  0.06197575189024063
```

```
uts.plotError(y_yhat, err, "ReHtilde")
```

Result of Experimentation - IUU MSE



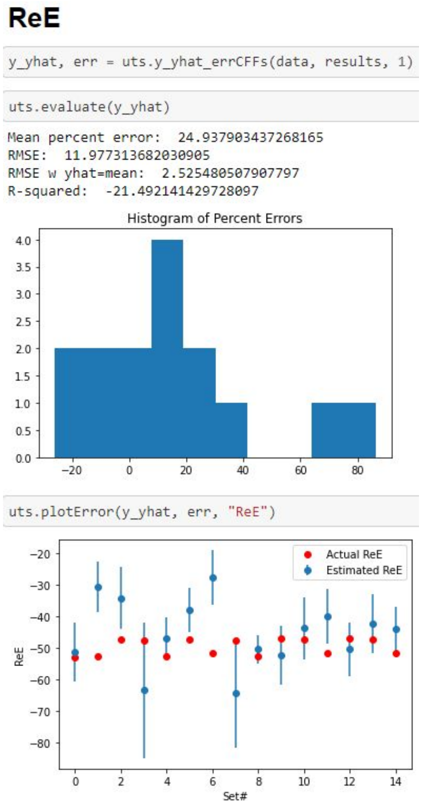
ReH

```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 0)
```

```
uts.evaluate(y_yhat)
```

```
Mean percent error:  21.15930035770254
RMSE:  2.1640016769238426
RMSE w yhat=mean:  2.5254628436780098
R-squared:  0.265768150438331
```

```
uts.plotError(y_yhat, err, "ReH")
```



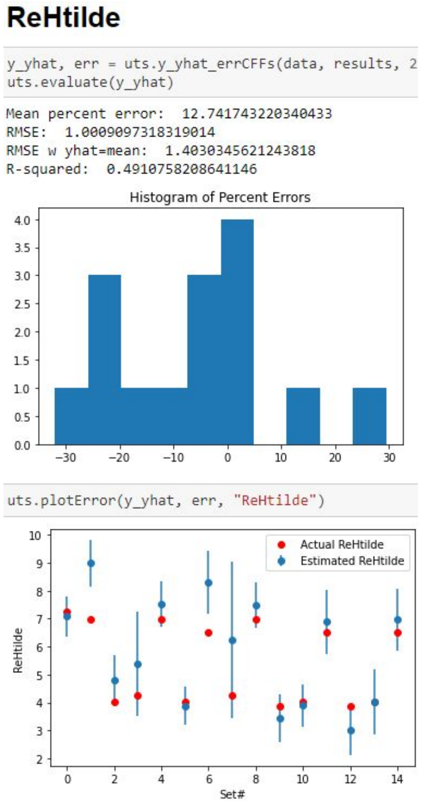
ReE

```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 1)
```

```
uts.evaluate(y_yhat)
```

```
Mean percent error:  24.937903437268165
RMSE:  11.977313682030905
RMSE w yhat=mean:  2.525480507907797
R-squared:  -21.492141429728097
```

```
uts.plotError(y_yhat, err, "ReE")
```



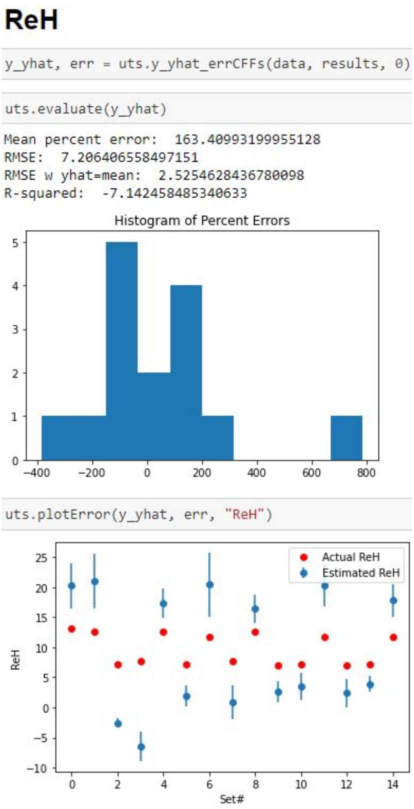
ReHtilde

```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 2
uts.evaluate(y_yhat)
```

```
Mean percent error:  12.741743220340433
RMSE:  1.0009097318319014
RMSE w yhat=mean:  1.4030345621243818
R-squared:  0.4910758208641146
```

```
uts.plotError(y_yhat, err, "ReHtilde")
```

Result of Experimentation - IUU MAPE



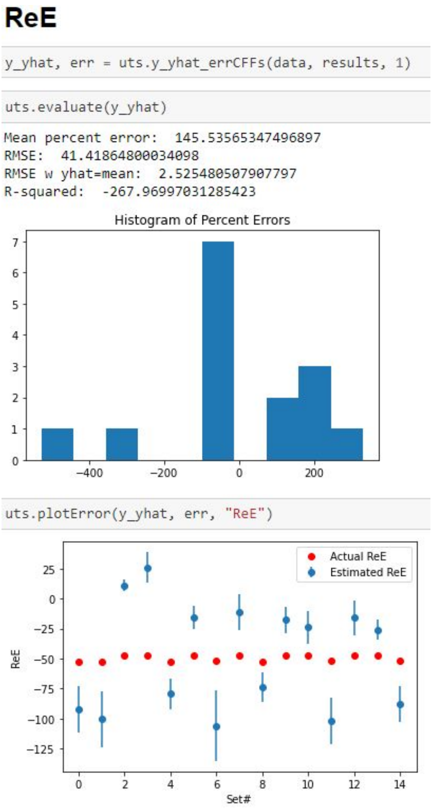
ReH

```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 0)
```

```
uts.evaluate(y_yhat)
```

```
Mean percent error:  163.40993199955128
RMSE:  7.206406558497151
RMSE w yhat=mean:  2.5254628436780098
R-squared:  -7.142458485340633
```

```
uts.plotError(y_yhat, err, "ReH")
```



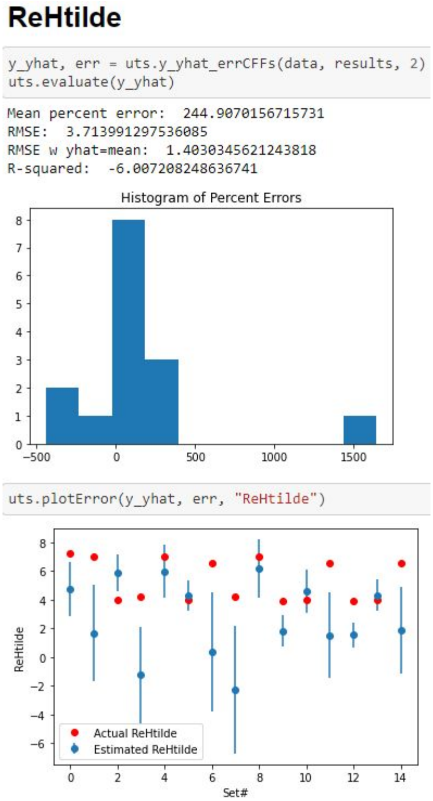
ReE

```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 1)
```

```
uts.evaluate(y_yhat)
```

```
Mean percent error:  145.53565347496897
RMSE:  41.41864800034098
RMSE w yhat=mean:  2.525480507907797
R-squared:  -267.96997031285423
```

```
uts.plotError(y_yhat, err, "ReE")
```



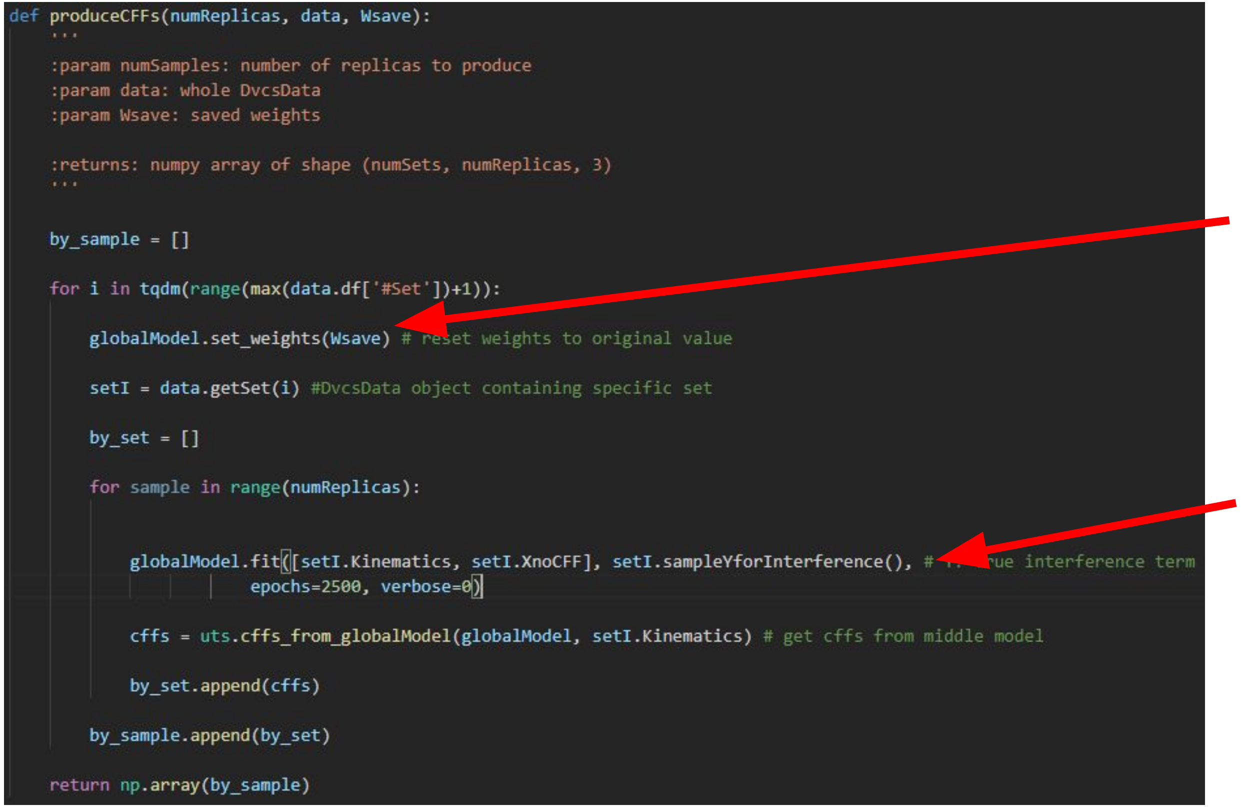
ReHtilde

```
y_yhat, err = uts.y_yhat_errCFFs(data, results, 2)
uts.evaluate(y_yhat)
```

```
Mean percent error:  244.9070156715731
RMSE:  3.713991297536085
RMSE w yhat=mean:  1.4030345621243818
R-squared:  -6.007208248636741
```

```
uts.plotError(y_yhat, err, "ReHtilde")
```

Neurons Updates for each Replica/Sample

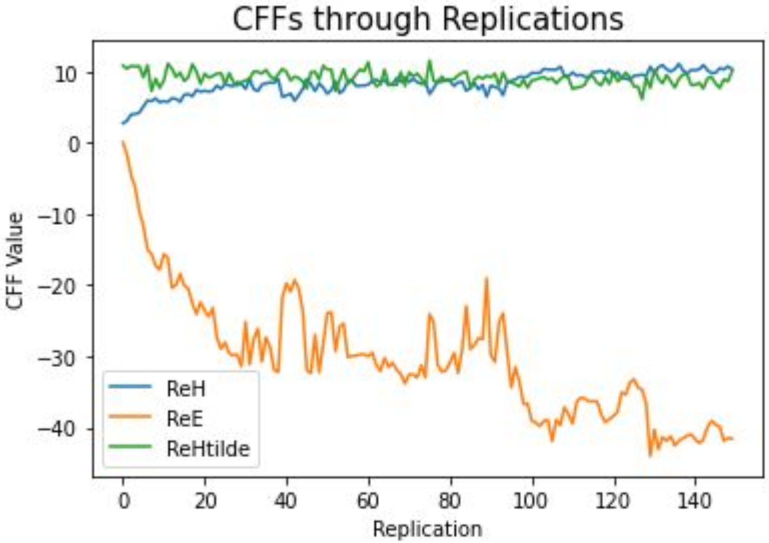


```
def produceCFFs(numReplicas, data, Wsave):
    '''
    :param numSamples: number of replicas to produce
    :param data: whole DvcsData
    :param Wsave: saved weights

    :returns: numpy array of shape (numSets, numReplicas, 3)
    '''

    by_sample = []

    for i in tqdm(range(max(data.df['#Set'])+1)):

        globalModel.set_weights(Wsave) # reset weights to original value

        setI = data.getSet(i) #DvcsData object containing specific set

        by_set = []

        for sample in range(numReplicas):

            globalModel.fit([setI.Kinematics, setI.XnoCFF], setI.sampleYforInterference(), # F: true interference term
                            epochs=2500, verbose=0)

            cffs = uts.cffs_from_globalModel(globalModel, setI.Kinematics) # get cffs from middle model

            by_set.append(cffs)

        by_sample.append(by_set)

    return np.array(by_sample)
```

Weights being reset for each new sample

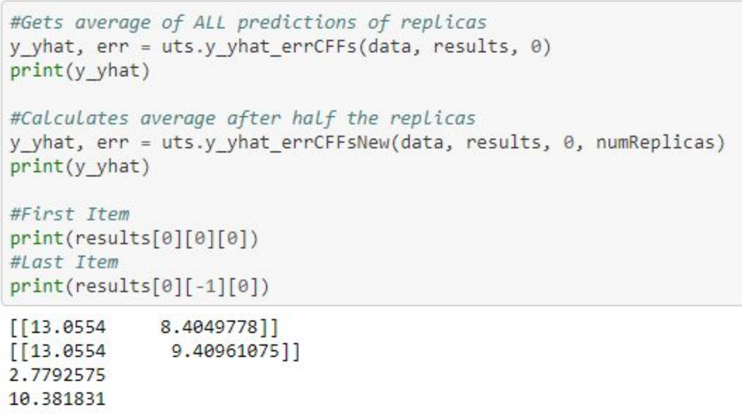
Weights being continually updated for each replica

Predictions of CFFs throughout Replicas



Predictions of CFFs throughout Replicas

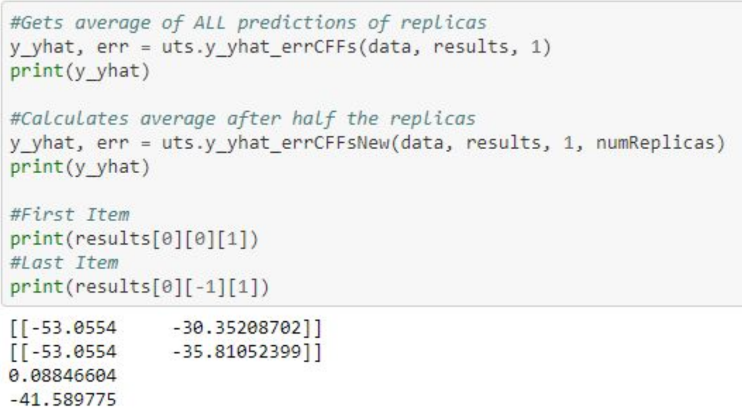
ReH



```
#Gets average of ALL predictions of replicas
y_yhat, err = uts.y_yhat_errCFFs(data, results, 0)
print(y_yhat)

#Calculates average after half the replicas
y_yhat, err = uts.y_yhat_errCFFsNew(data, results, 0, numReplicas)
print(y_yhat)

#First Item
print(results[0][0][0])
#Last Item
print(results[0][-1][0])
```

```
[[13.0554     8.4049778]]
[[13.0554      9.40961075]]
2.7792575
10.381831
```

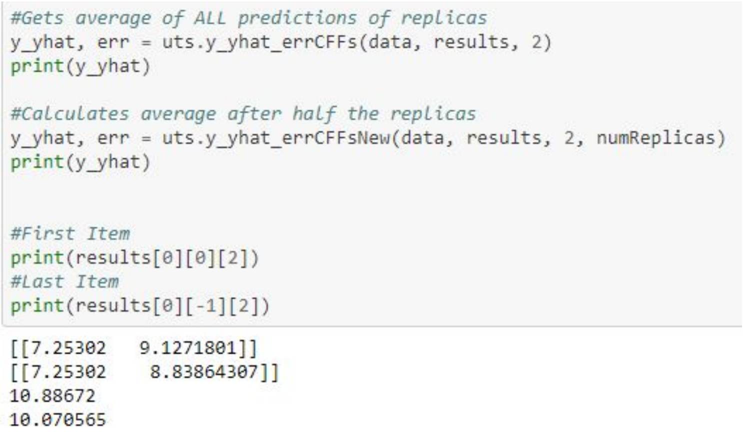
ReE



```
#Gets average of ALL predictions of replicas
y_yhat, err = uts.y_yhat_errCFFs(data, results, 1)
print(y_yhat)

#Calculates average after half the replicas
y_yhat, err = uts.y_yhat_errCFFsNew(data, results, 1, numReplicas)
print(y_yhat)

#First Item
print(results[0][0][1])
#Last Item
print(results[0][-1][1])
```

```
[[-53.0554     -30.35208702]]
[[-53.0554     -35.81052399]]
0.08846604
-41.589775
```

ReHtilde



```
#Gets average of ALL predictions of replicas
y_yhat, err = uts.y_yhat_errCFFs(data, results, 2)
print(y_yhat)

#Calculates average after half the replicas
y_yhat, err = uts.y_yhat_errCFFsNew(data, results, 2, numReplicas)
print(y_yhat)


#First Item
print(results[0][0][2])
#Last Item
print(results[0][-1][2])
```

```
[[7.25302    9.1271801]]
[[7.25302     8.83864307]]
10.88672
10.070565
```