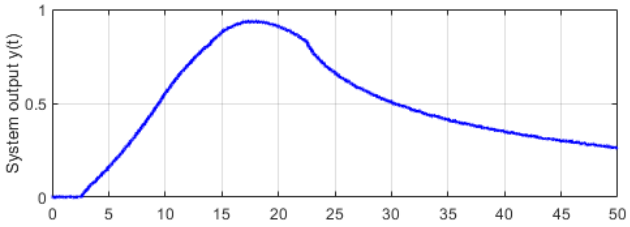
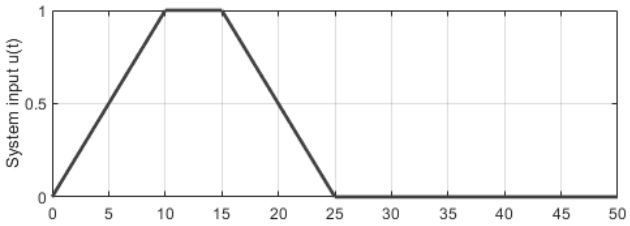
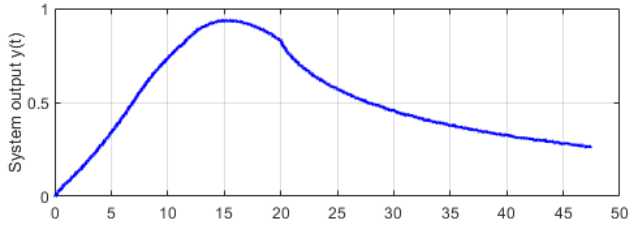
## Time domain IDENTIFICATION comparism between matlab and python fomcon toolbox

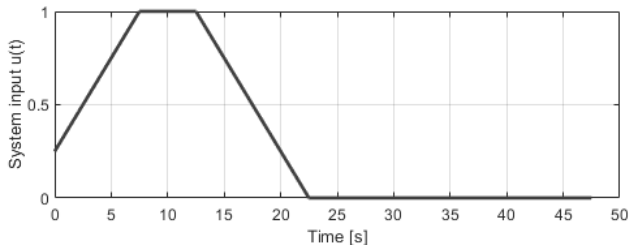
Data set used for comparism is same example set “idsets.mat” used in MATLAB fomcon Toolbox. Plots of Identification data and Validation data are displayed below. Matlab on the **Left**, Python on the **Right.**

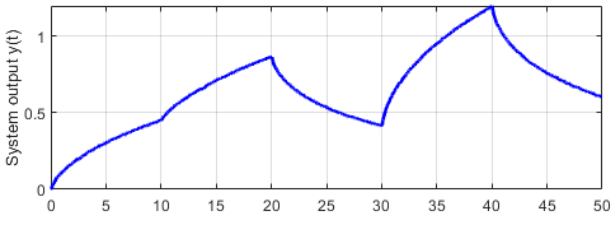
### MATLAB

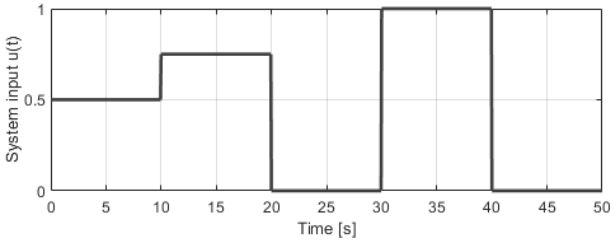


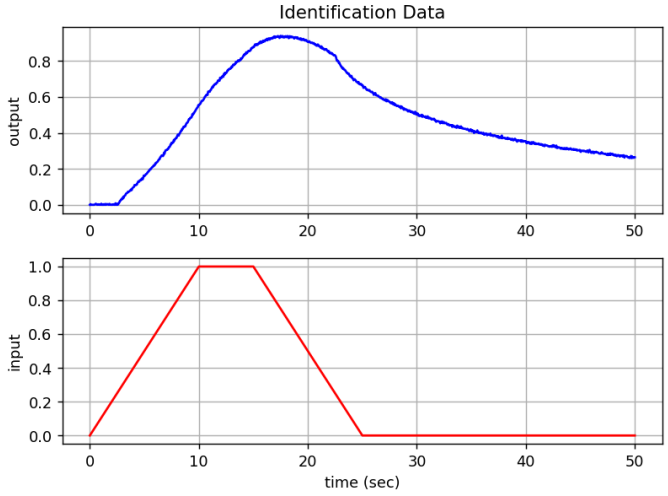



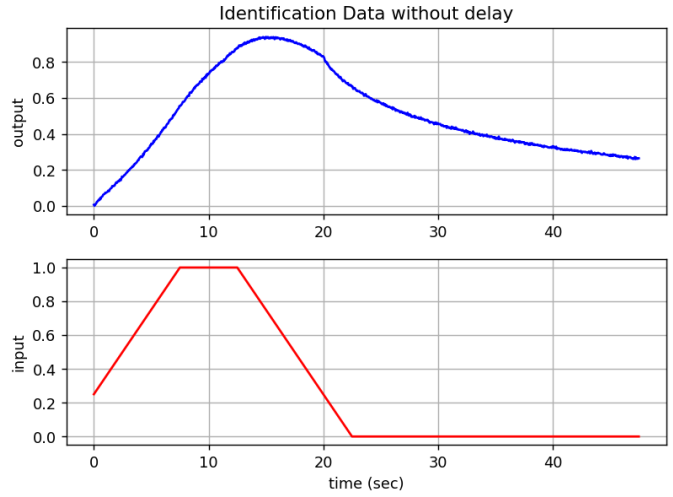



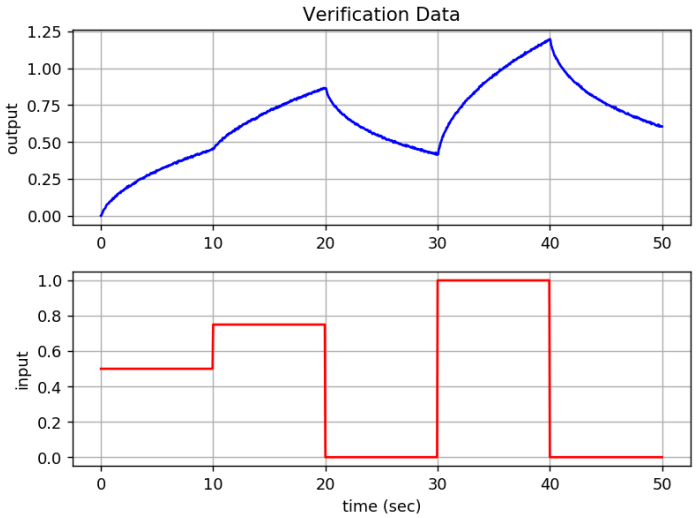




### PYTHON



# Identification using Levenberg Marquardt method

Using data without delay, identification of FOTF was done. The comparism is as below. FOMCON python performs as good as fomcon matlab with noticeable better performance in optimization of Coefficents. Please note that same initial guess was used in both comparism.

**Initial guess:** `newfotf('-2s^{0.63}+4', '2s^{3.501}+3.8s^{2.42}+2.6s^{1.798}+2.5s^{1.31}+1.5', 0)`

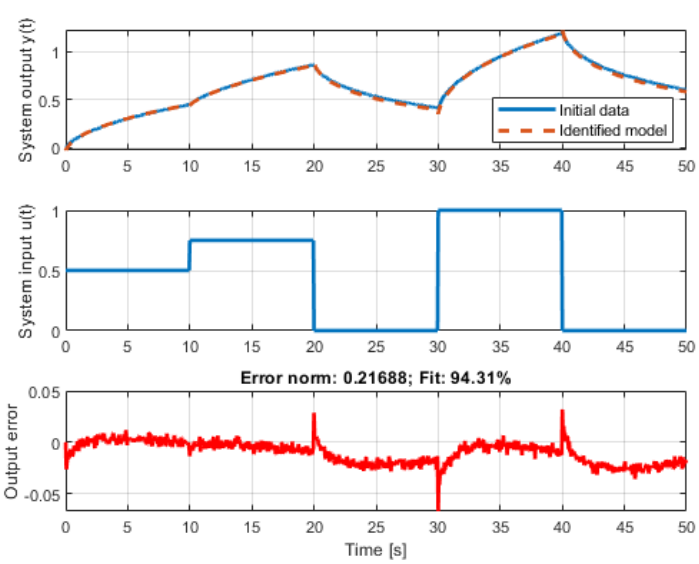
## Free optimization MATLAB (left) vs Python(right)

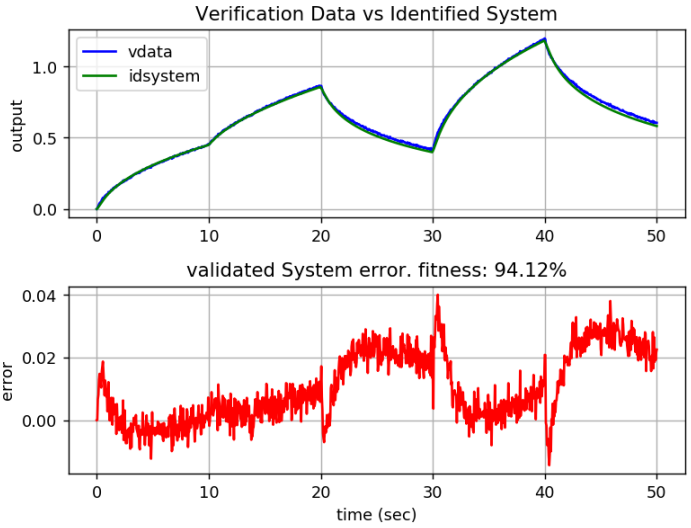
*Matlab identified FOTF: 94.31% fitness (better)*

$$\frac{-0.34637s^{0.68923} + 1.3186s^{5.2672e-14}}{1.45s^{0.59846} + 1.6647s^{0.59379} + 2.0018s^{0.59337} + 0.74478s^{1.5836e-6} - 0.42169s^{9.8837e-9}}$$

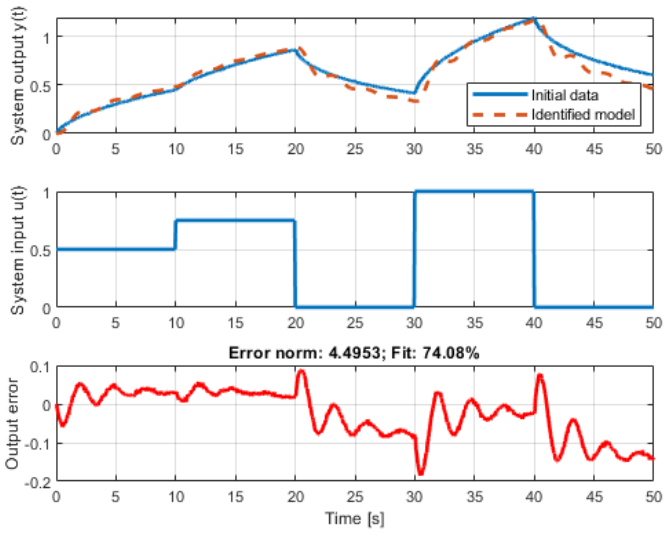
*Python Identified FOTF: 94.12% fitness*

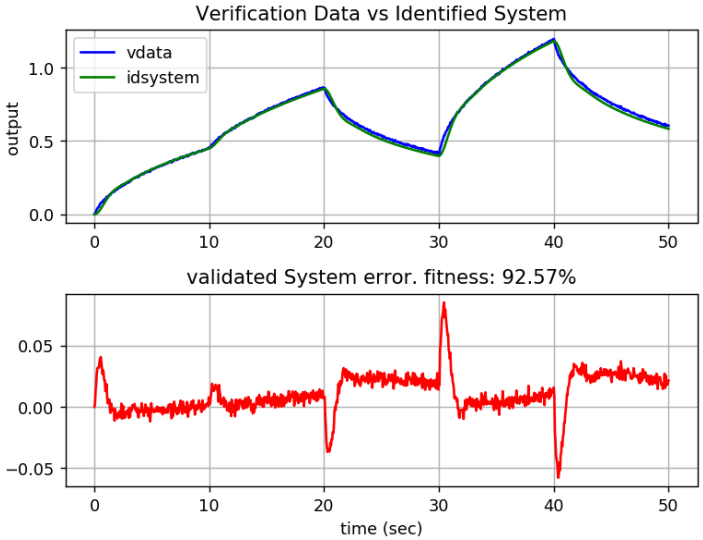
$$\frac{-1.72s\text{^}\{0.54\} + 3.92s\text{^}\{0.67\}}{0.84s\text{^}\{2.29\} + 3.64s\text{^}\{1.34\} + 2.03s\text{^}\{2.22\} + 1.72s\text{^}\{1.35\} + 4.01s\text{^}\{1.33\}}$$




## Exponent only Optimization. MATLAB (left) vs Python(right)



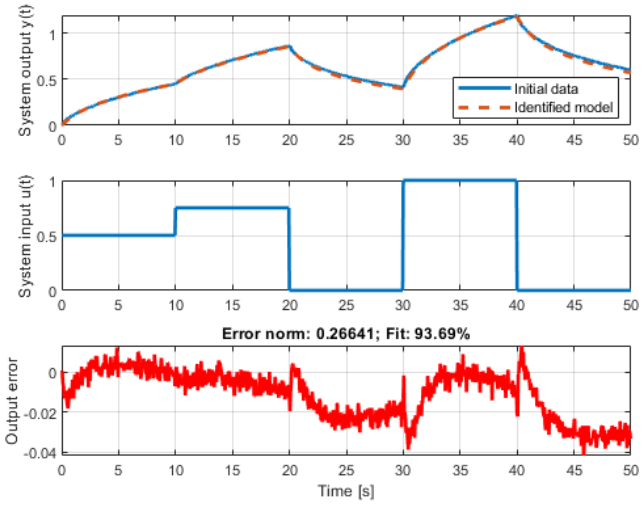
*Matlab identified FOTF: 74.08% fitness*

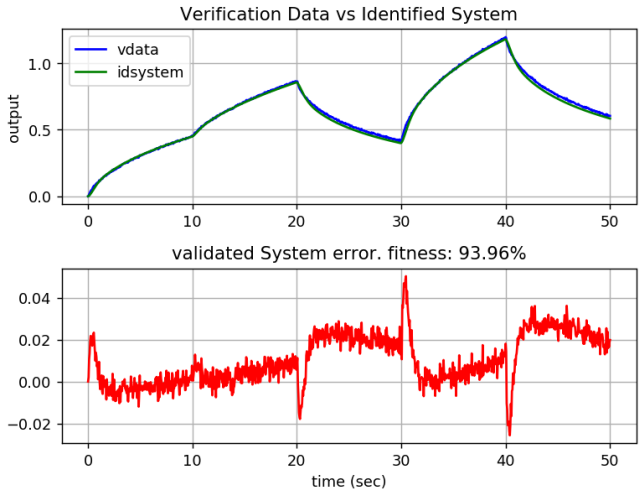
$$\frac{-2s^{0.24298} + 4s^{5.6925e-17}}{2s^{2.4913} + 3.8s^{0.63127} + 2.5s^{0.63125} + 2.6s^{0.63123} + 1.5}$$

*Python Identified FOTF: 92.57% fitness (better)*

$$\frac{-2s^{\{0.54\}} + 4s^{\{0.64\}}}{2s^{\{3.01\}} + 3.80s^{\{1.76\}} + 2.60s^{\{1.25\}} + 2.50s^{\{1.25\}} + 1.50s^{\{1.25\}}}$$

## **Coefficient only** Optimization. MATLAB (left) vs Python(right)




*Matlab identified FOTF 93.69% fitness*

$$\frac{0.53794s^{0.5} - 0.18905}{2.44s^{2} + 0.67006s^{1.5} + 1.3547s - 0.49482s^{0.5} - 0.029193}$$

*Python Identified FOTF: 93.96% fitness (better)*

$$\frac{64.30s^{\{0.63\}} - 4.96}{0.52s^{\{3.50\}} + 79.78s^{\{2.42\}} - 54.24s^{\{1.80\}} + 311.09s^{\{1.31\}} - 1.52}$$

# Identification using Trust Region Reflective method

The implemented optimization algorithm for TRR was also compared for matlab and python. It is also noticed that performance was almost saem,  and sometimes better. One thing to note is that the time for Free optimization in Python was much longer

```
Initial guess used:
newfotf('2s^{0.63}+4',2s^{3.501}+3.8s^{2.42}+2.6s^{1.798}+2.5s^{1.31}+1.5', 0)
```

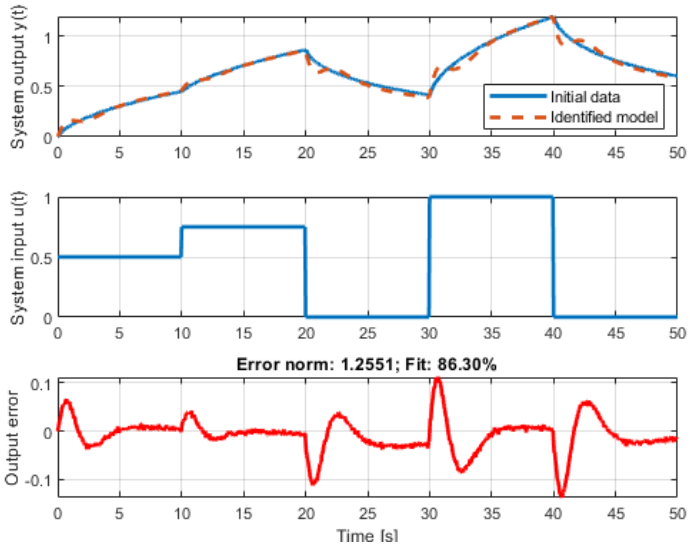
## Free optimization Mode. MATLAB (left) vs Python(right)

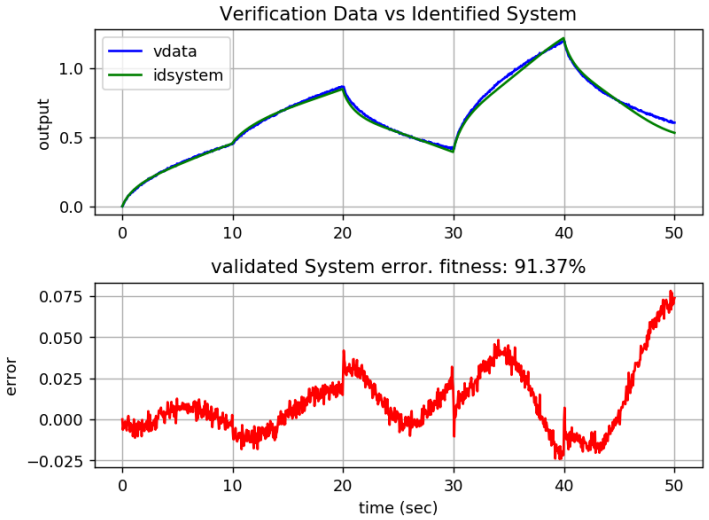
*Matlab identified FOTF: 86.30% fitness*

$$\frac{2.0448s^{2.9534} + 1.4113s^{1.1878}}{1.3947s^{4.2549} + 2.6111s^{3.3586} + 4.4203s^{2.7325} + 2.7416s^{1.8808} + 2.5548s^{1.5302}}$$

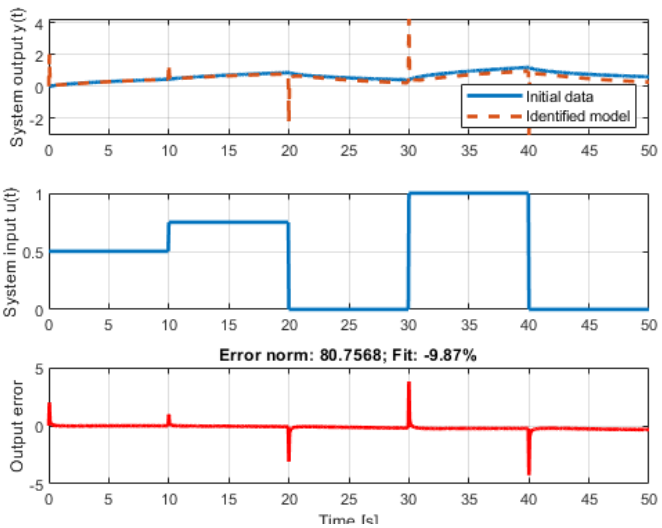
*Python Identified FOTF: 91.37% fitness (better)*

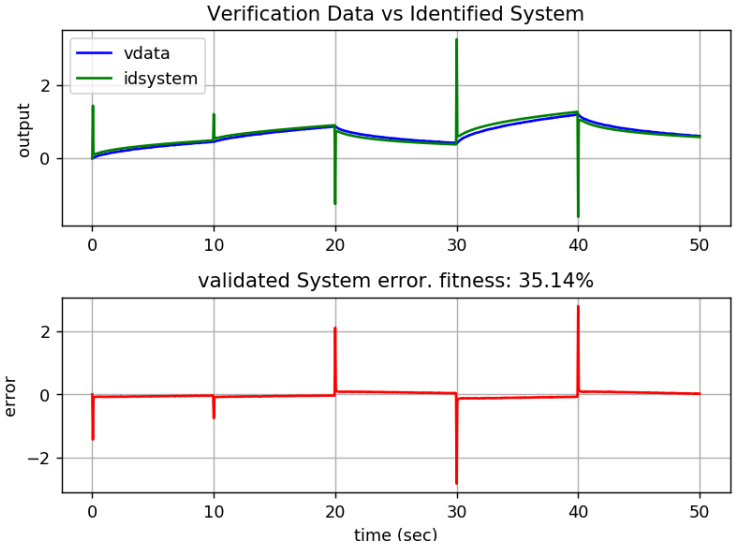
$$\frac{0.76s^{\{0.33\}} + 4.40s^{\{2.37\}}}{7.07s^{\{2.73\}} + 3.32s^{\{0.90\}} + 5.07s^{\{2.61\}} + 10.08s^{\{3.34\}} + 0.04s^{\{0.02\}}}$$




## Exponent only optimization mode. MATLAB (left) vs Python(right)



*Matlab identified FOTF: -9.87% fitness*

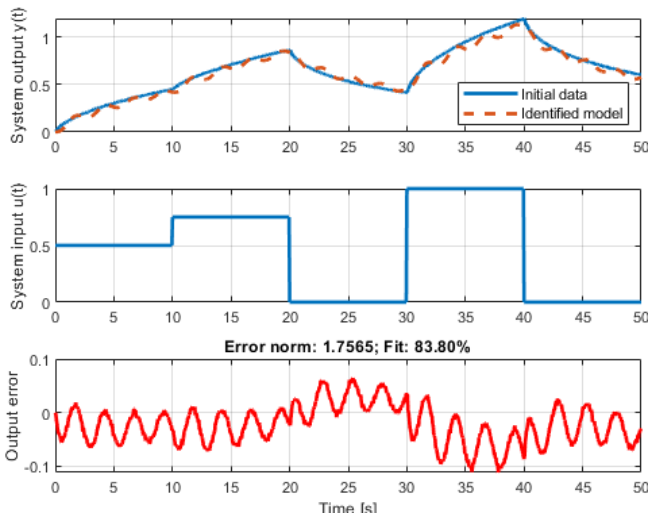
$$\frac{4s\text{^}1.7435 + 2s\text{^}0.01}{3.8s^{1.0134} + 2.5s^{0.90386} + 2s^{0.89697} + 2.6s^{0.89609} + 1.5s^{9.9997e-10}}$$

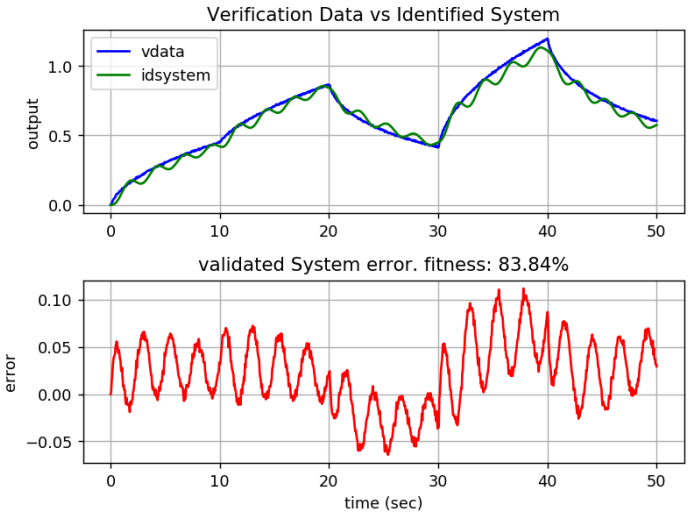
*Python Identified FOTF: 35.14% fitness (better)*

$$\frac{2s^{\{2.80\}} + 4s^{\{1.42\}}}{2s^{\{1.85\}} + 3.80s^{\{1.85\}} + 2.60s^{\{1.85\}} + 2.50s^{\{1.85\}} + 1.50s^{\{1.85\}}}$$

I will like to note how python optimization swaps the position of the coefficients though they are still same.

## Coefficient only Optimization Matlab(left) vs Python(right)




*Matlab identified FOTF 83.80% fitness*

$$\frac{3.411s^{0.63} + 7.1559e - 07}{2.6723s^{3.501} + 1.5457s^{2.42} + 2.1668e^{-06}s^{1.798} + 19.996s^{1.31} + 0.11252}$$

*Python Identified FOTF 83.84% fitness (better)*

$$\frac{3.41s\text{^}\{0.63\} + 0}{2.67s^{\{3.50\}} + 1.54s^{\{2.42\}} + 0s^{\{1.80\}} + 20s^{\{1.31\}} + 0.11}$$

# CONCLUSION

From the comparison, same results are obtained from python as compared to MATLAB. It is also noted that python performs better in more cases. Generally using trust region reflective method, and specifically in exponent optimization

# OTHER UPDATES

At the moment, I started learning QT5 to try implement the GUI (as stated in the third phase of the task document). Codes are also available on git for your perusal.