Fundamentals of TRF-analyses in Python

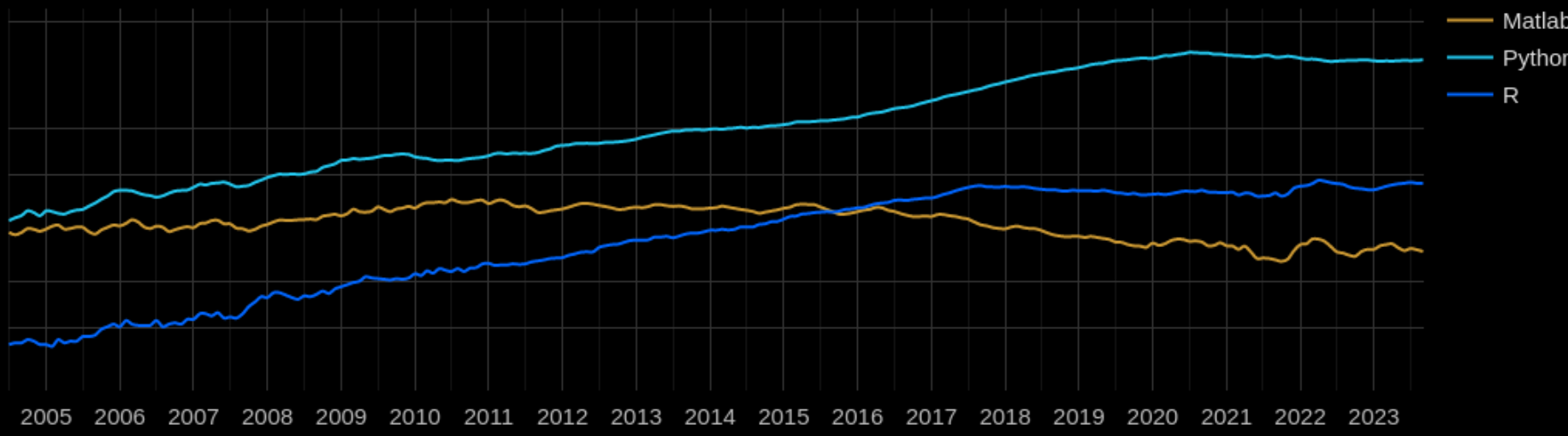
Ole Bialas

September 19, 2023

CNSP 2023

Why use Python?

- It is open source
- It is (one of) the most popular languages
- It is easy to learn
- It is a general purpouse language



PYPL PopularitY of Programming Language (https://pypl.github.io)

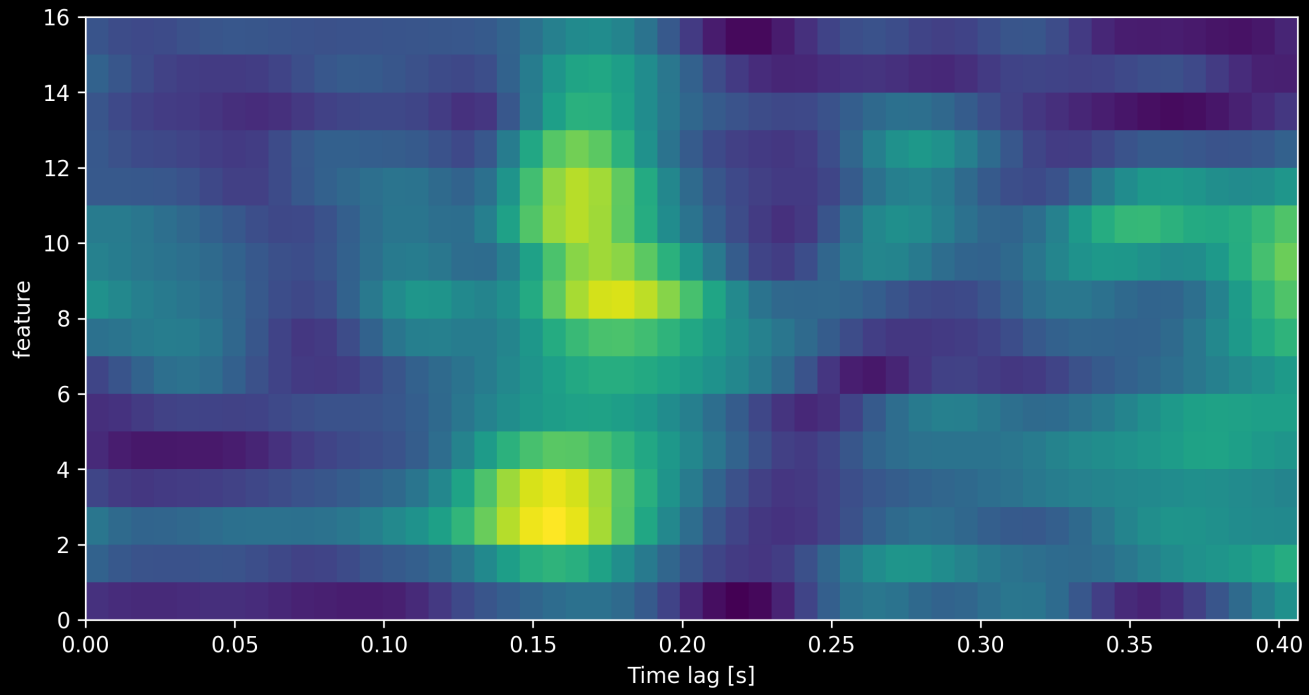
Introducing mTRFpy

Design objectives:

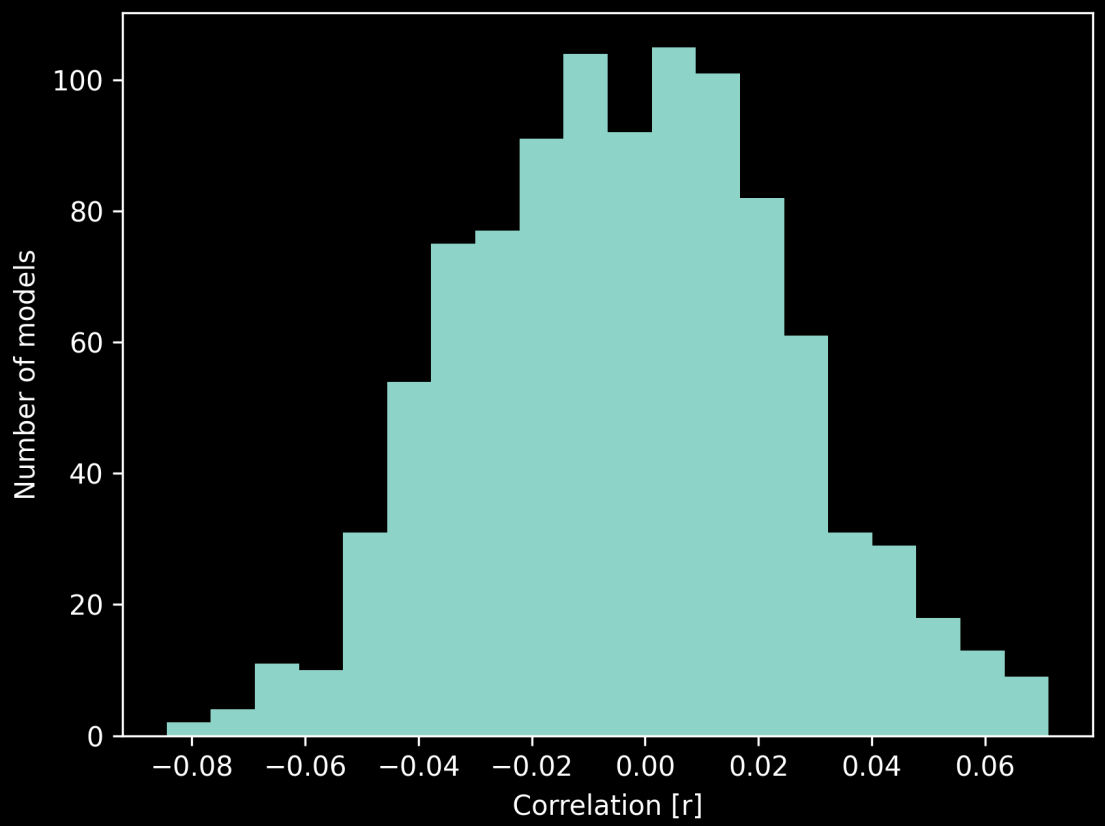
- Similar API as the Matlab mTRF-toolbox
- Object-oriented design
- Continuous integration
- Lightweight with minimal dependencies
- Compatibility with existing pipelines
- Easy to use and learn

```python
from mtrf import (
    TRF, # the main class
    stats, # cross-validation, permutation
    matrices # matrix operations
    )

trf = TRF(
    direction = 1, # 1 or -1
    kind = "multi", # "multi" or "single"
    zeropad = True, # pad time-lag matrix
    method = "ridge", # "tikhonov" or "banded"
    preload = True, # load covariance matrices
    metric = stats.pearsonr # evaluation function
    )

# Attributes
trf.weights # input by time by output matrix
trf.times # array with time lags
trf.regularization # value for lambda
```

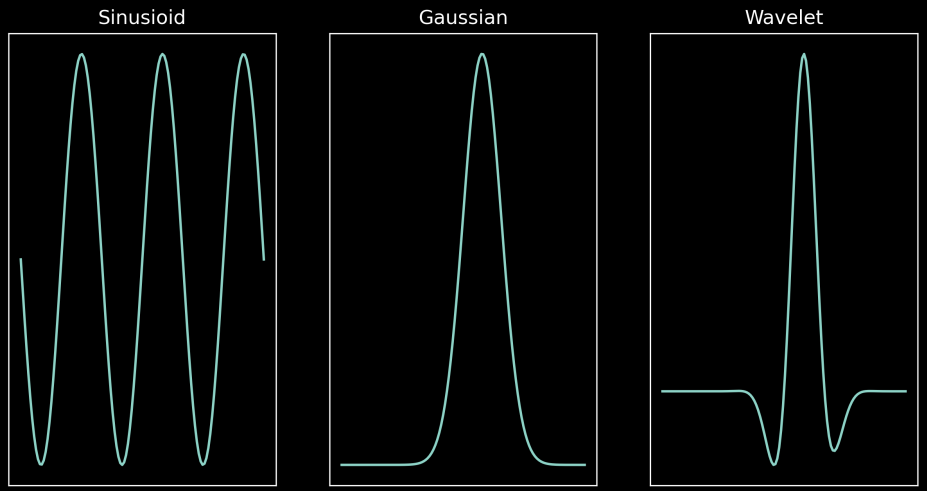
```python
from mtrf import TRF, load_sample_data
stimulus, response, fs = load_sample_data(normalize=True)
trf = TRF() # default forward model
# stimulus and response are lists of arrays (i.e. trials)
trf.train(stimulus, response, fs, tmin=0, tmax=0.35,
    regularization=1000)
trf.plot(channel="gfp", kind="image")
```

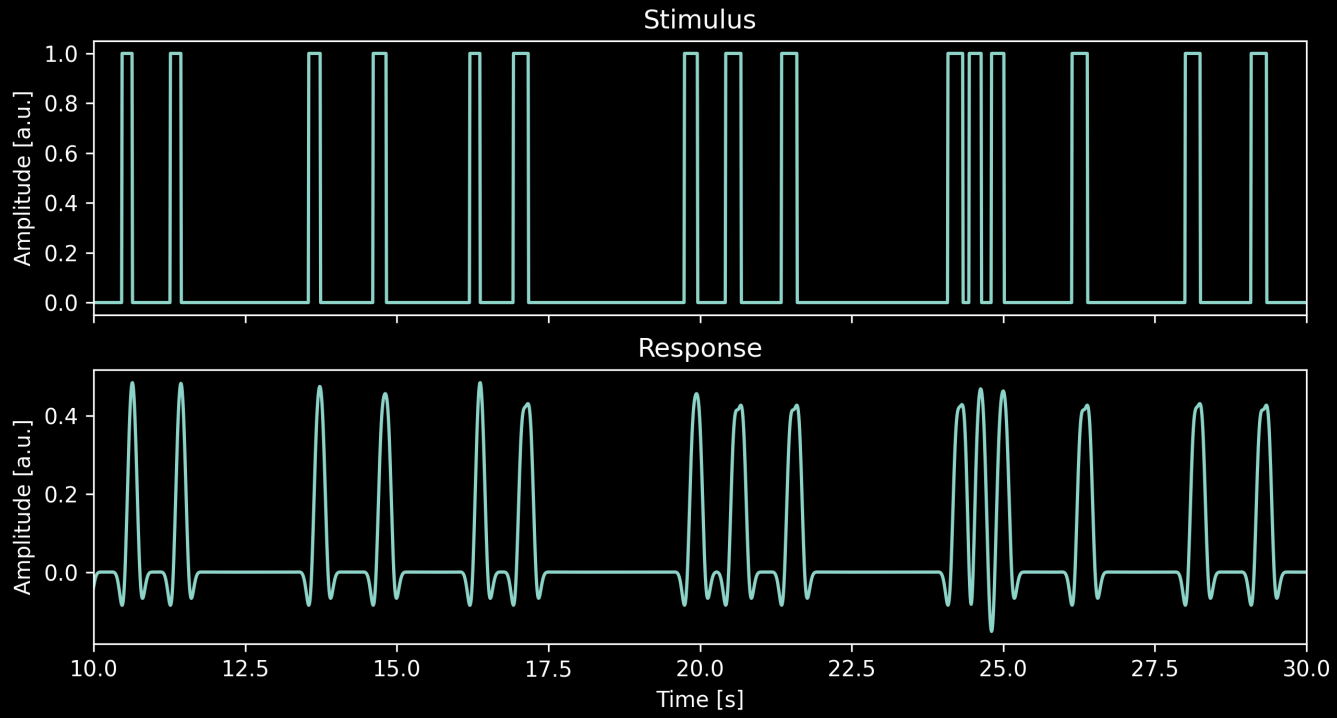


```
from mtrf import TRF, load_sample_data
from mtrf.stats import permutation_distribution
trf = TRF()
r_perm = permutation_distribution(trf, stimulus, response, fs,
    tmin=0, tmax=0.35, regularization=1000, n_permute=1000)
```

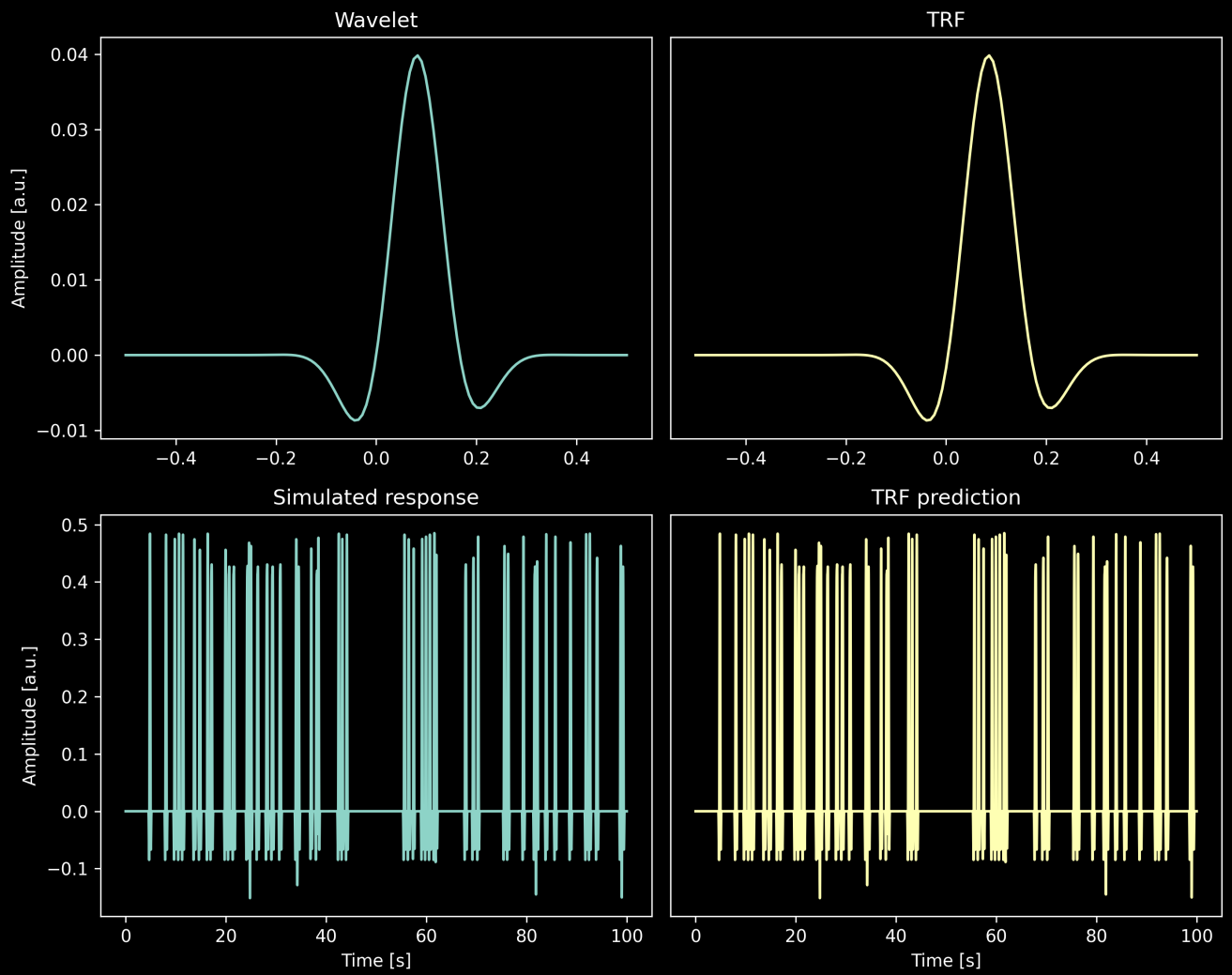


Demo - simulating neural responses with wavelets

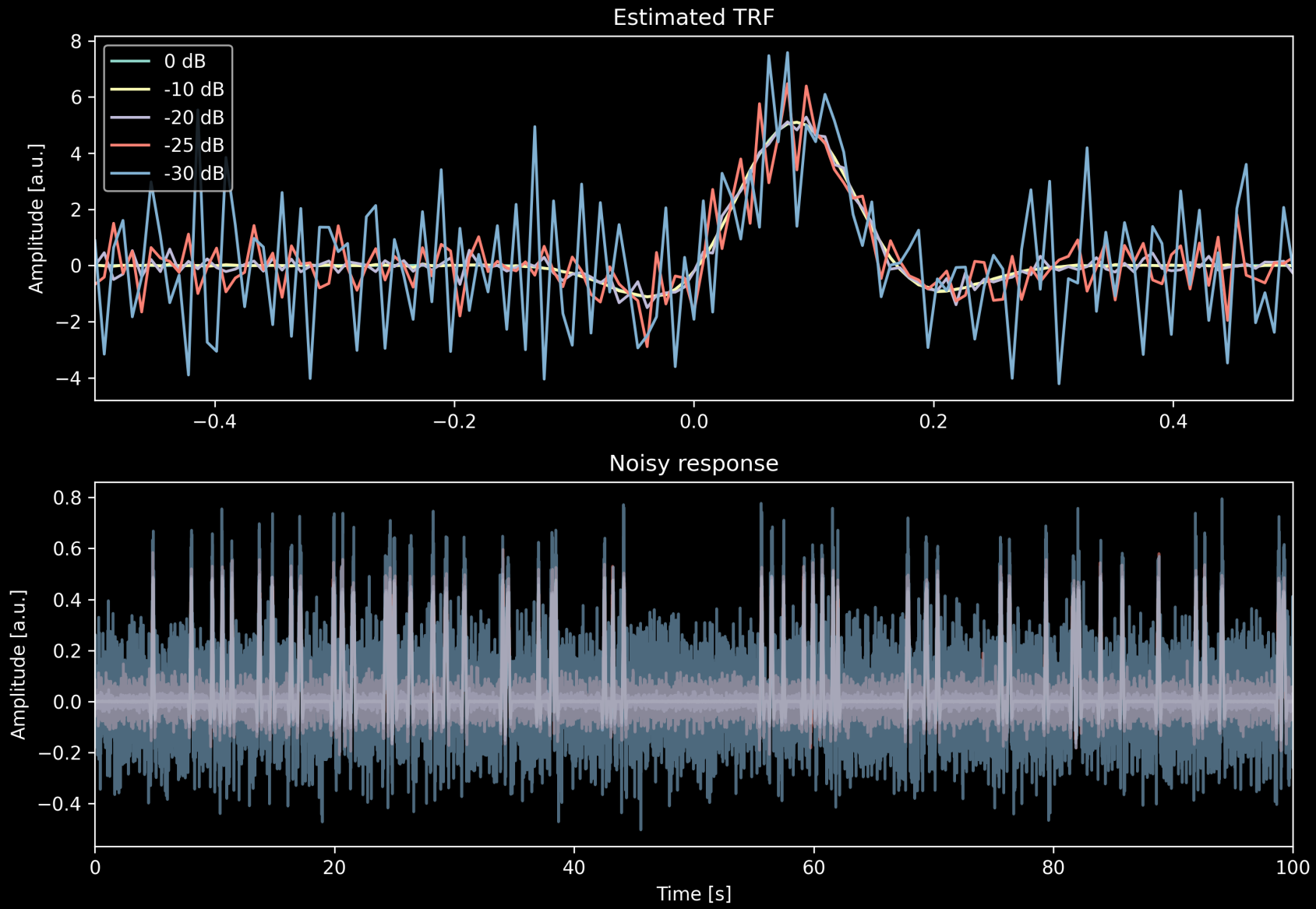

Sinusioid
Gaussian
Wavelet


Stimulus
Amplitude [a.u.]
1.0
0.8
0.6
0.4
0.2
0.0
Response
Amplitude [a.u.]
0.4
0.2
0.0
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0
Time [s]

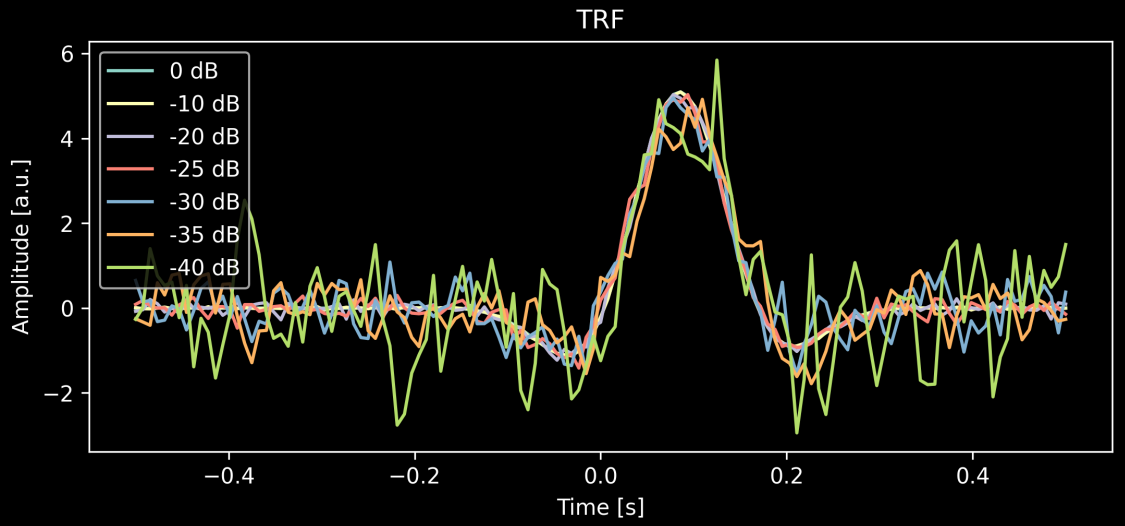
```
from mtrf import TRF
trf = TRF()
trf.train(stimulus, response, fs, tmin=-0.5, tmax=0.5,
    regularization=0)
prediction = trf.predict(stimulus)
```

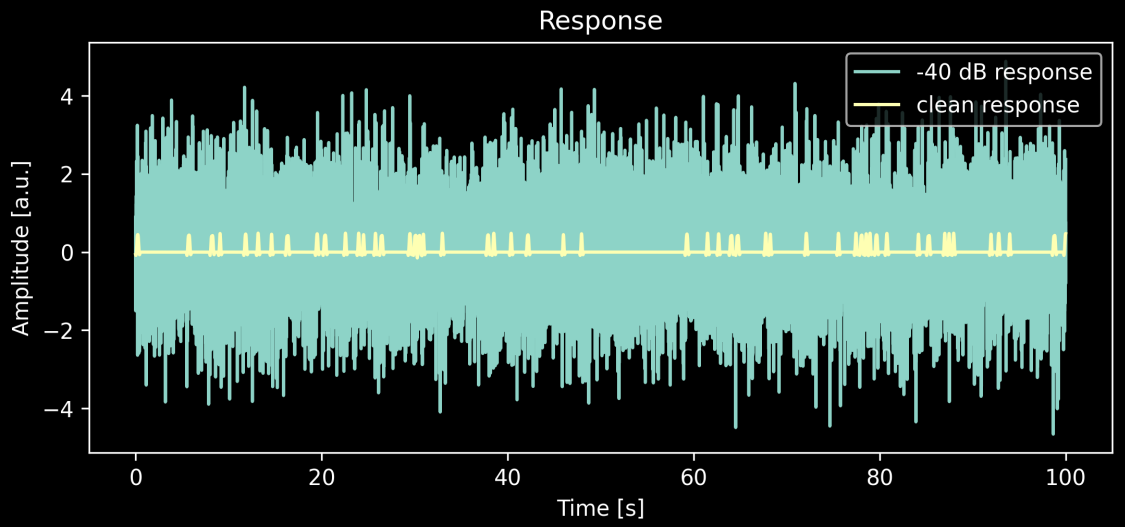


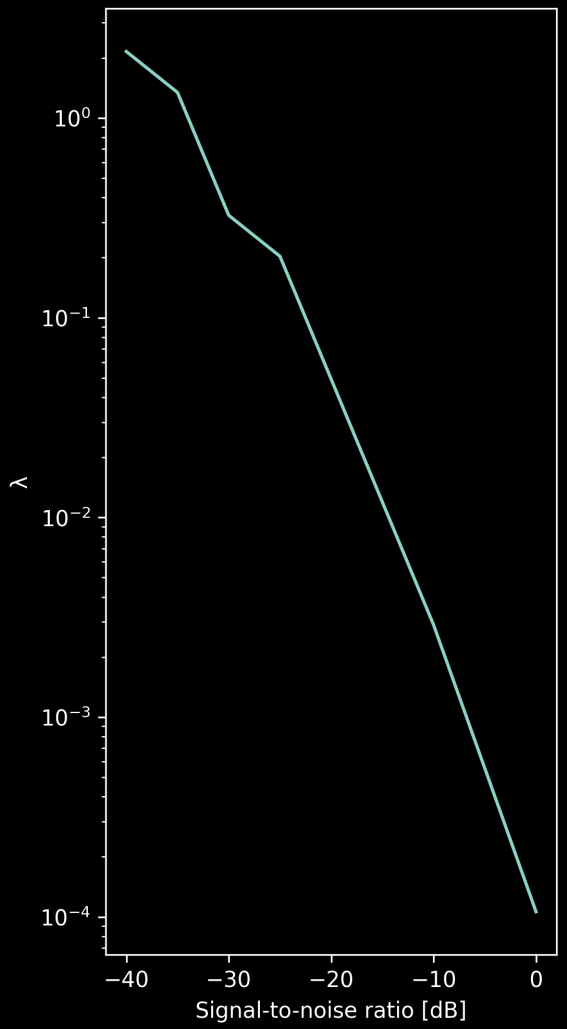
The effect of noise


Estimated TRF
0 dB
-10 dB
-20 dB
-25 dB
-30 dB
Amplitude [a.u.]
8
6
4
2
0
−2
−4
−0.4
−0.2
0.0
0.2
0.4
Noisy response
Amplitude [a.u.]
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
20
40
60
80
100
Time [s]

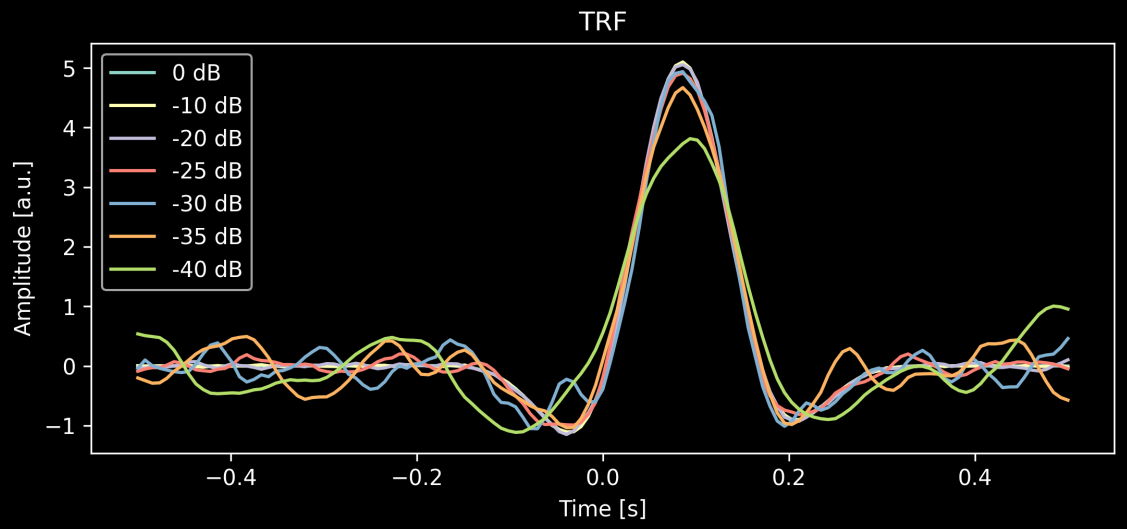
```
trf.train(stimulus, response, fs, tmin=-0.5, tmax=0.5,
    regularization=np.logspace(-5, 3, 40))
```

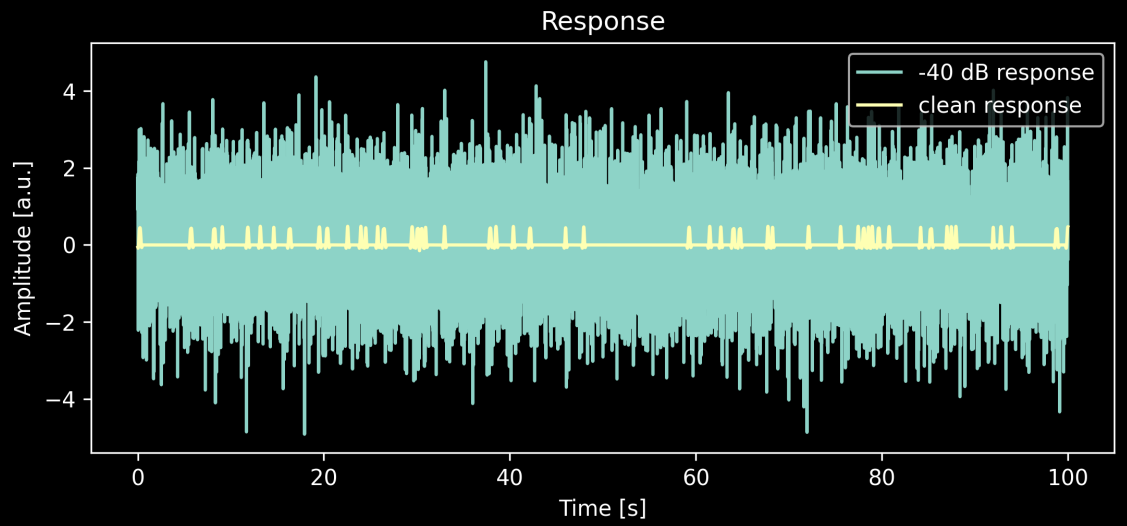


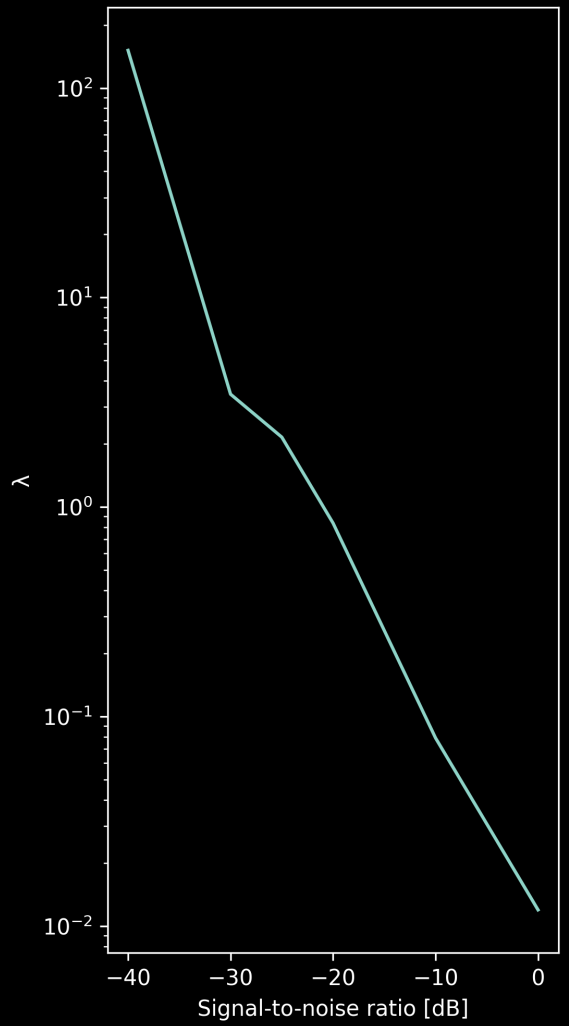




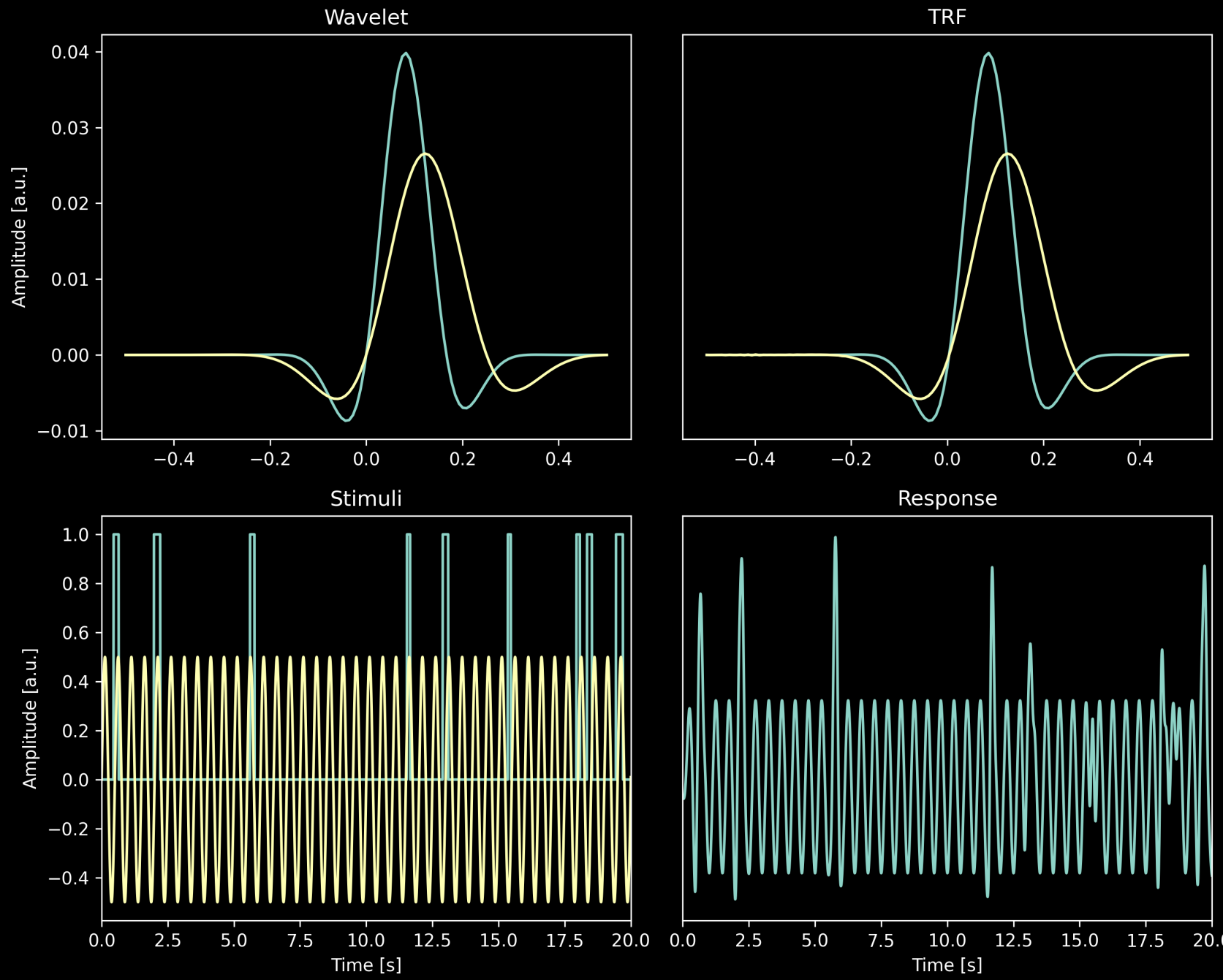
```
trf = TRF(method="tikhonov")
trf.train(stimulus, response, fs, tmin=-0.5, tmax=0.5,
    regularization=np.logspace(-5, 3, 40))
```



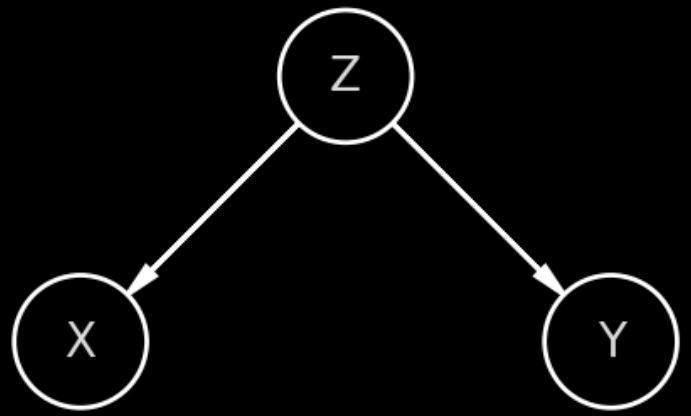


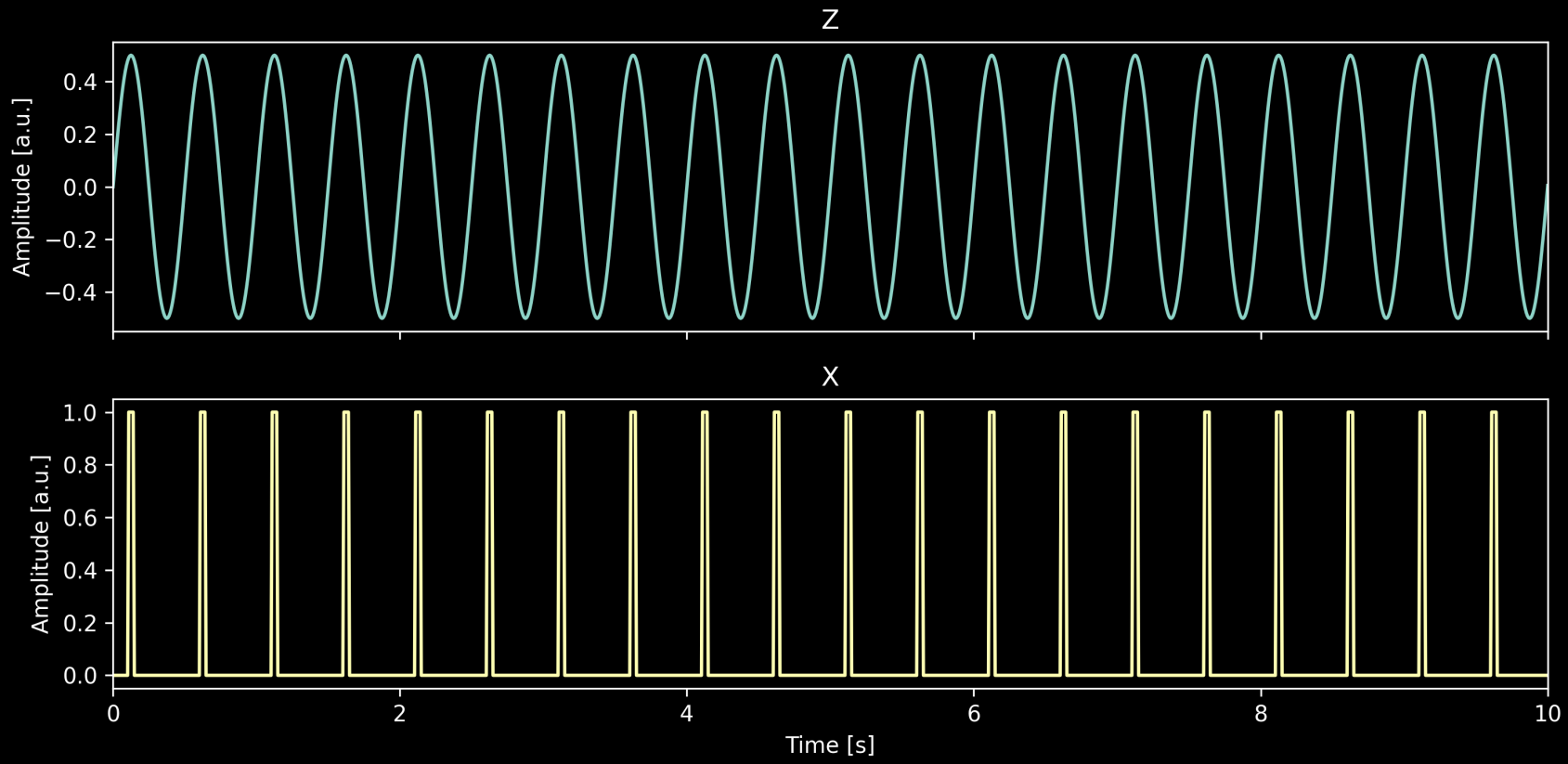


Multiple predictors

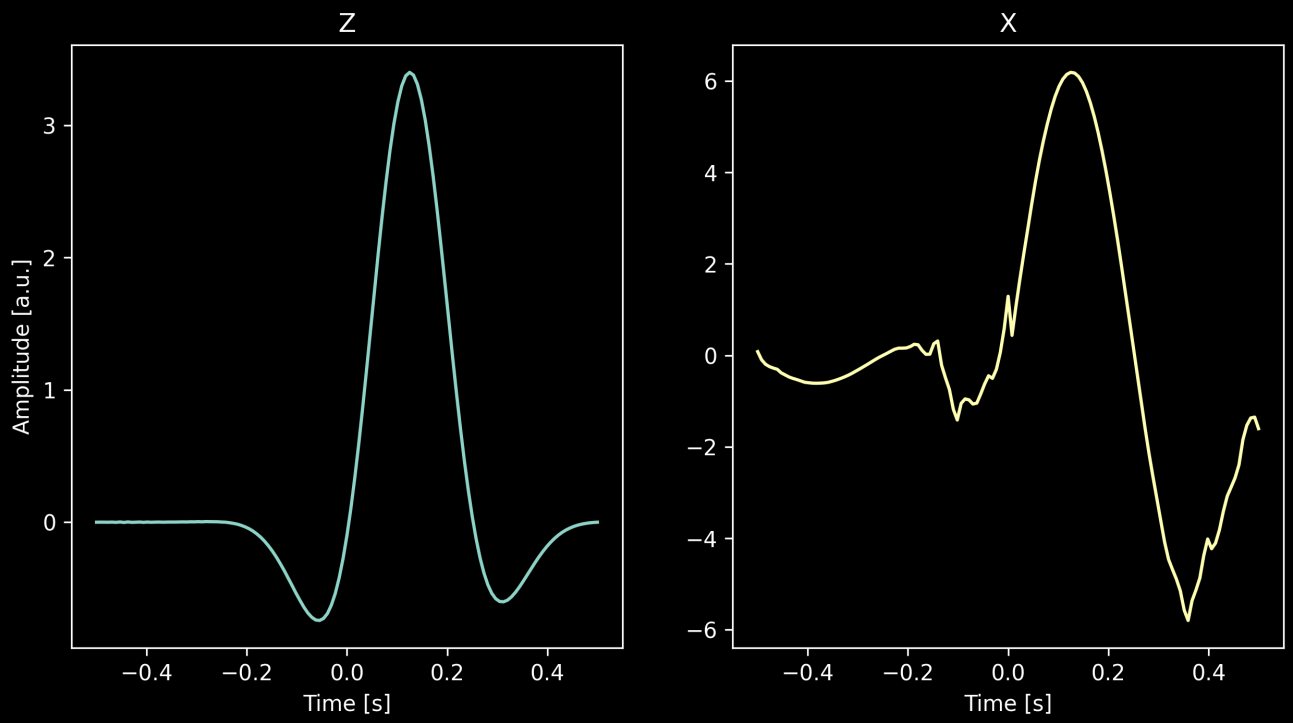

Wavelet
TRF
Amplitude [a.u.]
0.04
0.03
0.02
0.01
0.00
−0.01
−0.4
−0.2
0.0
0.2
0.4
Stimuli
Response
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time [s]

- X and Y are correlated because they are both caused by Z
- The correlation disappears after accounting for Z

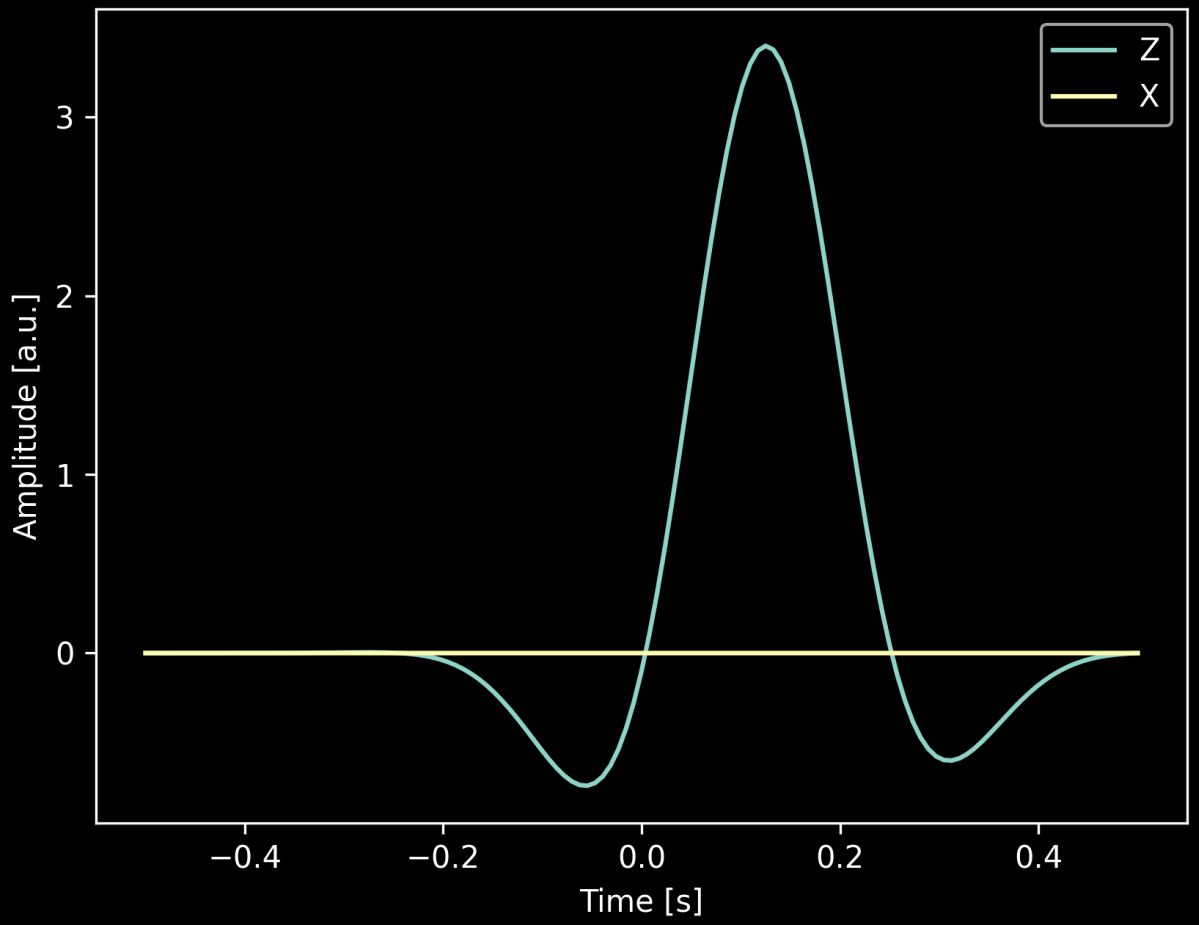




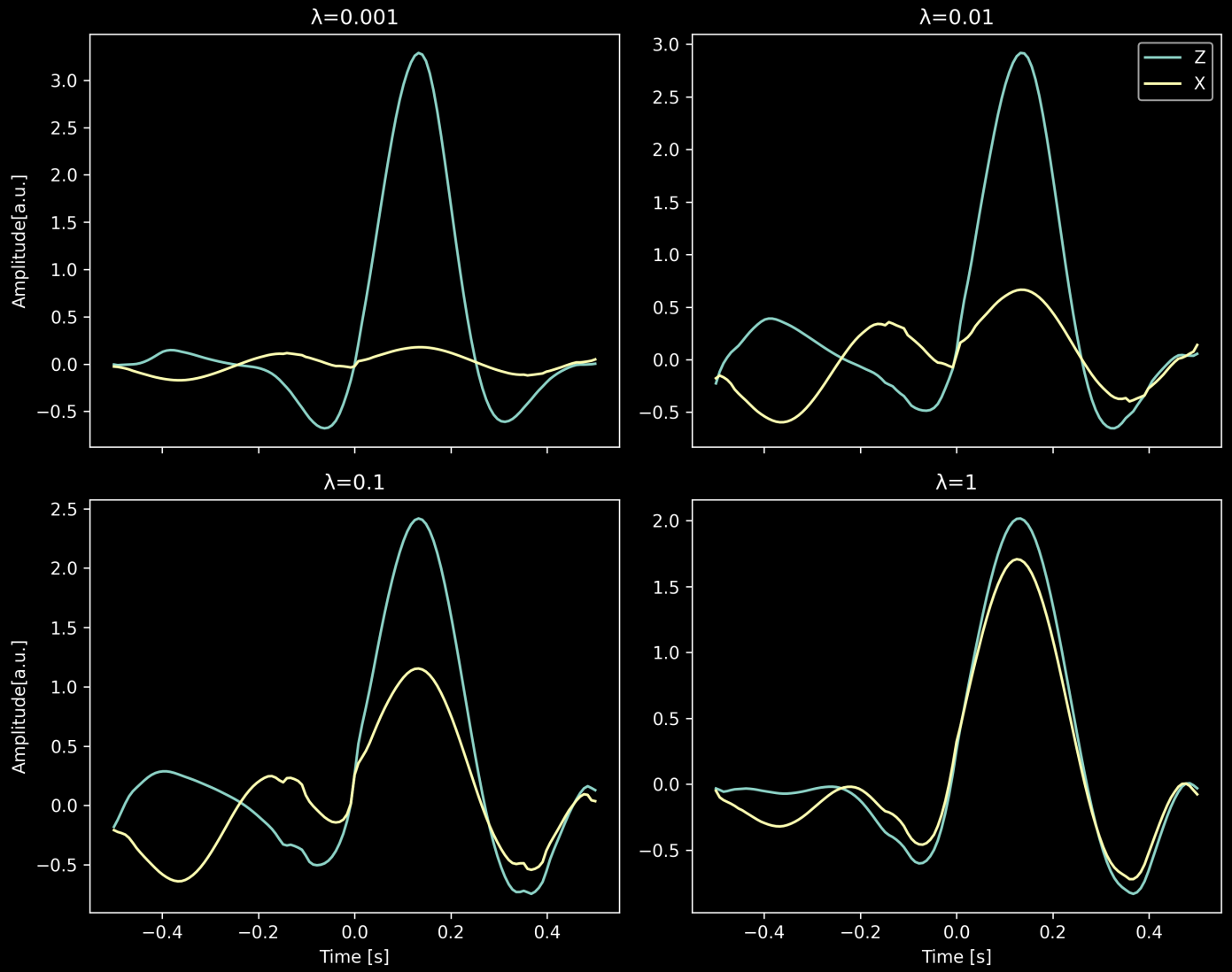
```
trf_Z, trf_X = TRF(), TRF()
trf_Z.train(stimulus_Z, response, fs, tmin=-0.5, tmax=0.5,
    regularization=0)
trf_X.train(stimulus_X, response, fs, tmin=-0.5, tmax=0.5,
    regularization=0)
```



```
stimulus = np.stack([stim_Z, stim_X], axis=1])
trf.train(stimulus, response, fs, tmin=-0.5, tmax=0.5,
    regularization=0)
```



```
trf = TRF(method="tikhonov")
for reg in [0.001, 0.01, 0.1, 1]:
    trf.train(stimulus, response, fs, tmin=-0.5, tmax=0.5,
        regularization=reg)
```



Thank you for your attention!

- If you are interested in contributing to the project please get in touch!
- Check out the online documentation at mtrfpy.readthedocs.io