System Identification

Guilherme Beal

May 31, 2023

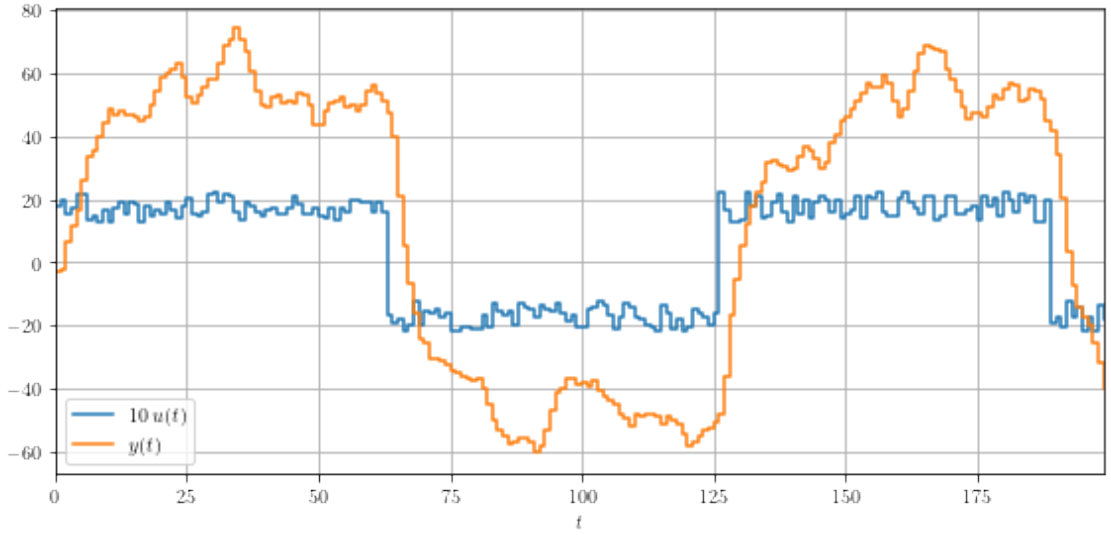
[ ]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

figPath = '../img/'
figExt  = 'eps'

matplotlib.set_loglevel('error')

matplotlib.rcParams.update({
  "text.usetex": True,
  "font.family": "serif",
  "pgf.texsystem": "pdflatex",
  "pgf.rcfonts": False,
})
```

1 Load and View Data

[ ]:
```
file = '../data.csv'
data = pd.read_csv(file, header=None, names=['u', 'y'])
N = len(data)

t = data.index.values
u = data.u.values
y = data.y.values

print('Number of data points:', N)
print(f't in [{t[0]}, {t[-1]}]')

plt.figure(figsize=(8,4))
plt.plot(t, 10*u, label='$10\\,u(t)$', drawstyle='steps-post')
plt.plot(t, y   , label='$y(t)$'     , drawstyle='steps-post')
plt.xlim(t[0], t[-1])
plt.xlabel('$t$')
plt.grid()
plt.legend()
```

```
plt.tight_layout()
plt.savefig(figPath + 'data.' + figExt, format=figExt)
plt.show()
```

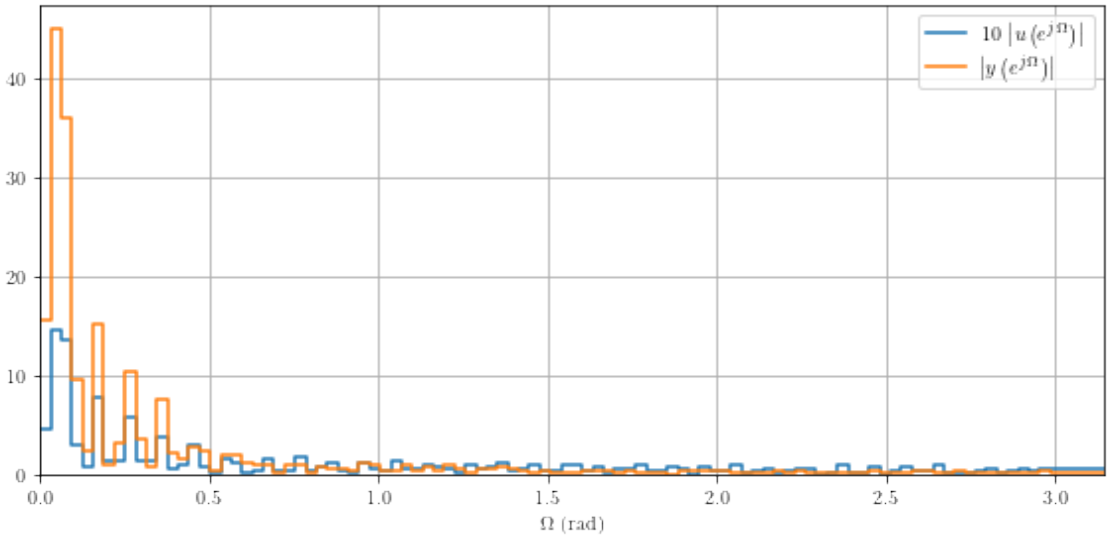
```
Number of data points: 200
t in [0, 199]
```

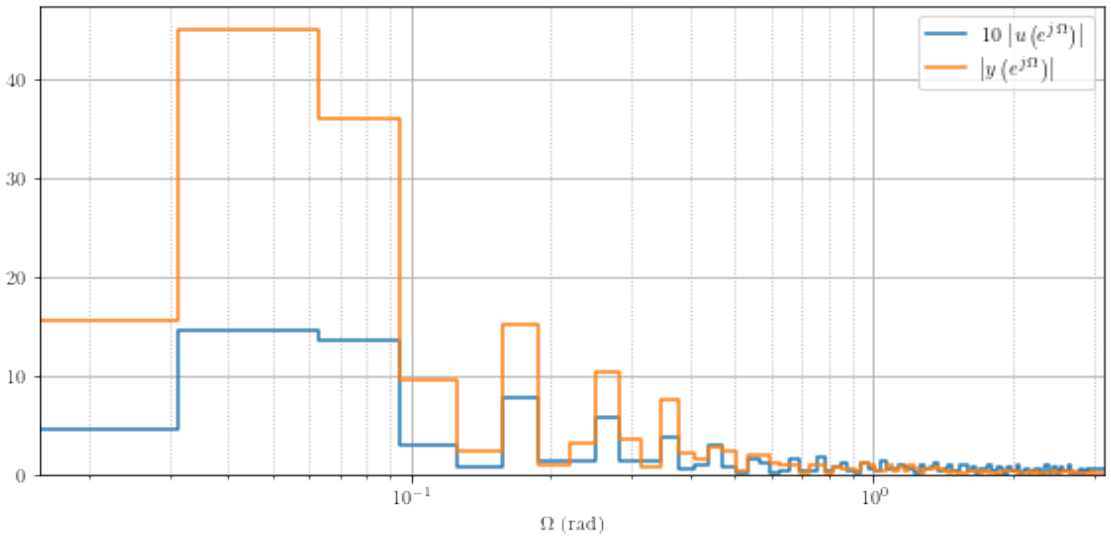


1.1 Input Fourier Transform

[ ]:
```
from scipy import fft

u_rfft = fft.rfft(u, norm='forward')
u_rfft[1:-1] = 2*u_rfft[1:-1]
y_rfft = fft.rfft(y, norm='forward')
y_rfft[1:-1] = 2*y_rfft[1:-1]

u_rfft_mag = np.abs(u_rfft)
y_rfft_mag = np.abs(y_rfft)
Omega = np.linspace(0, np.pi, len(u_rfft_mag))

plt.figure(figsize=(8,4))
plt.plot(Omega, 10*u_rfft_mag,␣
 ↪label='$10\\,\\left|u\\left(e^{{j\,\\Omega}}\\right)\\right|$',␣
 ↪drawstyle='steps-post')
plt.plot(Omega, y_rfft_mag   ,␣
 ↪label='$\\left|y\\left(e^{{j\,\\Omega}}\\right)\\right|$'     ,␣
 ↪drawstyle='steps-post')
plt.xlim(Omega[0], Omega[-1])
plt.ylim(0)
```

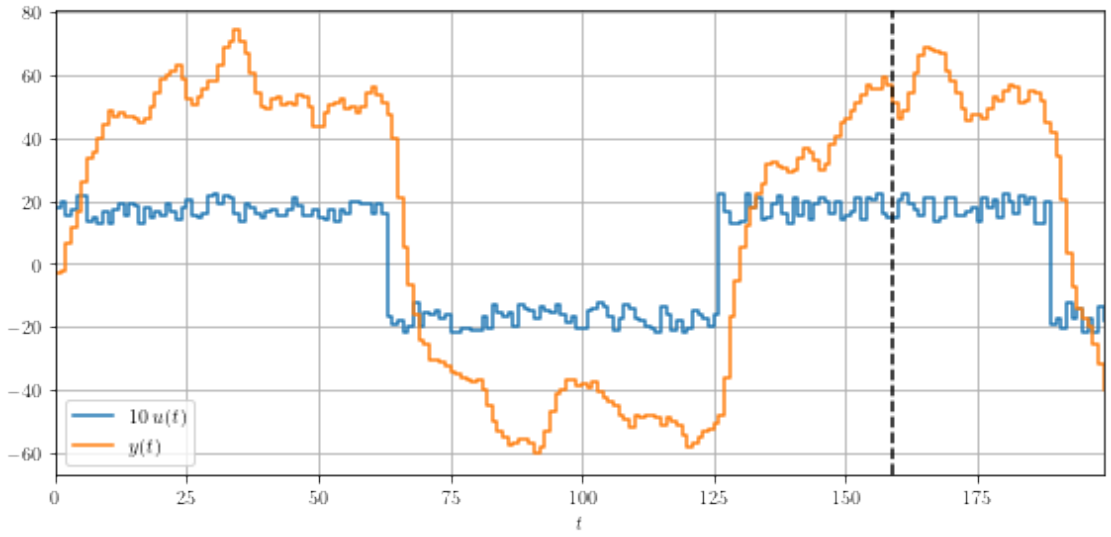
```
plt.xlabel('$\\Omega$ (rad)')
plt.grid()
plt.legend()
plt.tight_layout()
plt.savefig(figPath + 'data_fourier.' + figExt, format=figExt)
plt.show()

plt.figure(figsize=(8,4))
plt.plot(Omega, 10*u_rfft_mag,␣
  ↪label='$10\\,\\left|u\\left(e^{{j\,\\Omega}}\\right)\\right|$',␣
  ↪drawstyle='steps-post')
plt.plot(Omega, y_rfft_mag   ,␣
  ↪label='$\\left|y\\left(e^{{j\,\\Omega}}\\right)\\right|$'     ,␣
  ↪drawstyle='steps-post')
plt.xscale('log')
plt.xlim(Omega[1]/2, Omega[-1])
plt.ylim(0)
plt.xlabel('$\\Omega$ (rad)')
plt.grid(which='major')
plt.grid(which='minor', linestyle=':')
plt.legend()
plt.tight_layout()
plt.savefig(figPath + 'data_fourier_log.' + figExt, format=figExt)
plt.show()
```





1.2 Separate Identification and Validation Data

[ ]:
```
N_fold = 160

t_i = t[:N_fold]
u_i = u[:N_fold]
y_i = y[:N_fold]

t_v = t[N_fold:]
u_v = u[N_fold:]
y_v = y[N_fold:]

plt.figure(figsize=(8,4))
plt.plot(t, 10*u, label='$10\,u(t)$', drawstyle='steps-post')
plt.plot(t, y   , label='$y(t)$'    , drawstyle='steps-post')
plt.axvline(t[N_fold-1], color='black', linestyle='--')
plt.xlim(t[0], t[-1])
plt.xlabel('$t$')
plt.grid()
plt.legend()
plt.tight_layout()
plt.savefig(figPath + 'data_folded.' + figExt, format=figExt)
plt.show()
```



2 Generic Model

$$A(q)\,y(t) = \frac{B(q)}{F(q)}\,u(t) + \frac{C(q)}{D(q)}\,e(t)$$

$$y(t) = G(q)\,u(t) + H(q)\,e(t)$$

$$G(q) = \frac{B(q)}{A(q)\,F(q)} \qquad H(q) = \frac{C(q)}{A(q)\,D(q)}$$

$$A(q) = 1 + a_1\,q^{-1} + \cdots + a_{n_a}\,q^{-n_a}$$

$$B(q) = q^{-n_k}\,\left(b_0 + b_1\,q^{-1} + \cdots + b_{n_b}\,q^{-n_b}\right)$$

$$C(q) = 1 + c_1\,q^{-1} + \cdots + c_{n_c}\,q^{-n_c}$$

$$D(q) = 1 + d_1\,q^{-1} + \cdots + d_{n_d}\,q^{-n_d}$$

$$F(q) = 1 + f_1\,q^{-1} + \cdots + f_{n_f}\,q^{-n_f}$$

2.1 Prediction Error Method

$$\hat{y}(t) = L_u(q)\,u(t) + L_y(q)\,y(t)$$

$$L_u(q) = \frac{G(q)}{H(q)}$$

$$L_y(q) = 1 - \frac{1}{H(q)}$$

2.2 Prediction Cost

$$\hat{J} = \frac{1}{N}\sum_{k=1}^{N}\left(y(t) - \hat{y}(t)\right)^2$$

3 ARX

$$y(t) = G(q)\,u(t) + H(q)\,e(t)$$
$$G(q) = \frac{B(q)}{A(q)} \qquad H(q) = \frac{1}{A(q)}$$
$$A(q) = 1 + a_1\,q^{-1} + \cdots + a_{n_a}\,q^{-n_a}$$
$$B(q) = q^{-n_k}\,\left(b_0 + b_1\,q^{-1} + \cdots + b_{n_b}\,q^{-n_b}\right)$$

$$n_a = \{1, 2, 3, 4\} \qquad n_b = \{0, 1, 2, 3, 4\} \qquad n_k = \{1, 2, 3, 4\}$$

[ ]:
```
from functions import arx

na_range = range(1, 4 + 1)
nb_range = range(0, 4 + 1)
nk_range = range(1, 4 + 1)

models_arx = arx(u_i, y_i, u_v, y_v, na_range, nb_range, nk_range)
```

3.1 Display Best

[ ]:
```
from functions import display_models

columns=[
  'na', 'nb', 'nk',
  'Jv', 'Ji',
  'AICv', 'AICCv',
  'AICi', 'AICCi',
  'A', 'B',
]

for criterion in ['Jv', 'AICCv', 'AICCi']:
  display_models(models_arx.sort_values(by=[criterion]), precision=3, qty=6,␣
  ↪columns=columns)
```

```
    na  nb  nk     Jv     Ji    AICv   AICCv     AICi    AICCi  \
56   3   4   1  5.133  5.296  81.427  86.072  282.699  283.653
36   2   4   1  5.179  5.297  79.783  83.283  280.747  281.484
52   3   3   1  5.376  5.263  81.274  84.774  279.705  280.442
32   2   3   1  5.410  5.264  79.531  82.077  277.740  278.289
64   4   1   1  5.419  5.611  79.600  82.145  287.946  288.495
68   4   2   1  5.503  5.167  82.208  85.708  276.753  277.490

                                      A  \
56          [ 1.    -1.419  0.514 -0.015]
```

```
36                [ 1.    -1.409  0.489]
52         [ 1.    -1.418  0.511 -0.013]
32                   [ 1.   -1.41  0.49]
64  [ 1.    -1.356  0.452  0.114 -0.101]
68  [ 1.    -1.399  0.398  0.189 -0.109]
```

```
                                                B
56  [ 0.     2.132  1.632 -1.689  0.274 -0.061]
36  [ 0.     2.127  1.646 -1.695  0.23  -0.044]
52         [ 0.     2.13   1.628 -1.685  0.24 ]
32         [ 0.     2.126  1.642 -1.69   0.21 ]
64                          [0.    2.147 0.994]
68                [ 0.     2.165  1.714 -1.594]
```

```
    na  nb  nk     Jv     Ji    AICv   AICCv     AICi    AICCi  \
24   2   1   1  5.801  5.750  78.320  79.463  287.883  288.142
44   3   1   1  5.556  5.740  78.594  80.358  289.581  289.971
28   2   2   1  5.589  5.276  78.832  80.596  276.114  276.503
20   2   0   1  6.384  5.910  80.153  80.820  290.269  290.423
40   3   0   1  6.074  5.875  80.160  81.303  291.314  291.572
32   2   3   1  5.410  5.264  79.531  82.077  277.740  278.289
```

```
                                A                                        B
24         [ 1.    -1.31   0.417]                    [0.    2.142 0.941]
44  [ 1.    -1.352  0.503 -0.046]                    [0.    2.143 0.873]
28         [ 1.    -1.407  0.483]          [ 0.     2.162  1.611 -1.602]
20         [ 1.    -1.371  0.464]                           [0.    2.656]
40  [ 1.    -1.415  0.557 -0.051]                           [0.    2.605]
32           [ 1.   -1.41  0.49]  [ 0.     2.126  1.642 -1.69   0.21 ]
```

```
    na  nb  nk     Jv     Ji    AICv   AICCv     AICi    AICCi  \
28   2   2   1  5.589  5.276  78.832  80.596  276.114  276.503
68   4   2   1  5.503  5.167  82.208  85.708  276.753  277.490
32   2   3   1  5.410  5.264  79.531  82.077  277.740  278.289
48   3   2   1  5.634  5.275  81.153  83.698  278.076  278.625
72   4   3   1  5.611  5.134  84.986  89.631  277.739  278.693
52   3   3   1  5.376  5.263  81.274  84.774  279.705  280.442
```

```
                                    A                                      B
28                [ 1.    -1.407  0.483]          [ 0.     2.162  1.611 -1.602]
68  [ 1.    -1.399  0.398  0.189 -0.109]          [ 0.     2.165  1.714 -1.594]
32                   [ 1.   -1.41  0.49]  [ 0.     2.126  1.642 -1.69   0.21 ]
48         [ 1.    -1.399  0.463  0.012]          [ 0.     2.156  1.624 -1.623]
72  [ 1.    -1.421  0.441  0.188 -0.124]  [ 0.     2.134  1.686 -1.821  0.449]
52         [ 1.    -1.418  0.511 -0.013]  [ 0.     2.13   1.628 -1.685  0.24 ]
```

4 ARMAX

$$y(t) = G(q)\,u(t) + H(q)\,e(t)$$
$$G(q) = \frac{B(q)}{A(q)} \qquad H(q) = \frac{C(q)}{A(q)}$$
$$A(q) = 1 + a_1\,q^{-1} + \cdots + a_{n_a}\,q^{-n_a}$$
$$B(q) = q^{-n_k}\,\left(b_0 + b_1\,q^{-1} + \cdots + b_{n_b}\,q^{-n_b}\right)$$
$$C(q) = 1 + c_1\,q^{-1} + \cdots + c_{n_c}\,q^{-n_c}$$

$$n_a = \{1,2,3,4\} \qquad n_b = \{0,1,2,3,4\} \qquad n_c = \{1,2,3,4\} \qquad n_k = \{1,2,3,4\}$$

[ ]:
```
from functions import armax

na_range = range(1, 4 + 1)
nb_range = range(0, 4 + 1)
nc_range = range(1, 4 + 1)
nk_range = range(1, 4 + 1)

models_armax = armax(u_i, y_i, u_v, y_v, na_range, nb_range, nc_range, nk_range)
```

4.1 Display Best

[ ]:
```
from functions import display_models

columns=[
  'na', 'nb', 'nc', 'nk',
  'Jv', 'Ji',
  'AICv', 'AICCv',
  'AICi', 'AICCi',
  'A', 'B', 'C',
]

for criterion in ['Jv', 'AICCv', 'AICCi']:
  display_models(models_armax.sort_values(by=[criterion]), precision=3, qty=6,␣
 ↪columns=columns)
```

```
     na  nb  nc  nk     Jv     Ji    AICv   AICCv     AICi    AICCi  \
64    1   4   1   1  5.199  5.922  79.935  83.435  298.594  299.330
144   2   4   1   1  5.250  5.299  82.332  86.977  282.814  283.768
256   4   1   1   1  5.277  5.337  80.531  84.031  281.932  282.669
272   4   2   1   1  5.473  5.147  83.994  88.640  278.136  279.090
128   2   3   1   1  5.476  5.266  82.015  85.515  279.812  280.548
172   3   0   4   1  5.568  5.472  84.678  89.323  287.945  288.898
```

```
                                        A  \
64                          [ 1.    -0.848]
144                   [ 1.    -1.405  0.485]
256  [ 1.    -0.755 -0.382  0.415 -0.114]
272  [ 1.    -1.218  0.146  0.276 -0.109]
128                   [ 1.    -1.406  0.486]
172           [ 1.    -1.619  0.986 -0.277]

                                              B  \
64   [ 0.     2.262  2.875 -0.327 -0.13  -0.202]
144  [ 0.     2.075  1.709 -1.778  0.308 -0.044]
256                           [0.    1.906 2.829]
272                   [ 0.     2.092  2.16  -1.528]
128           [ 0.     2.073  1.706 -1.777  0.286]
172                                 [0.    2.619]

                                        C
64                            [1.    0.429]
144                           [1.    0.013]
256                           [1.    0.627]
272                           [1.    0.201]
128                           [1.    0.013]
172  [ 1.    -0.252  0.298 -0.05   0.081]

     na  nb  nc  nk     Jv     Ji    AICv   AICCv     AICi    AICCi  \
80    2   0   1   1  6.244  5.905  81.264  82.406  292.128  292.387
160   3   0   1   1  5.853  5.802  80.675  82.440  291.308  291.698
112   2   2   1   1  5.579  5.277  80.757  83.302  278.136  278.685
64    1   4   1   1  5.199  5.922  79.935  83.435  298.594  299.330
32    1   2   1   1  6.021  5.902  81.809  83.574  294.060  294.450
256   4   1   1   1  5.277  5.337  80.531  84.031  281.932  282.669

                                        A  \
80                    [ 1.    -1.4    0.489]
160           [ 1.    -1.582  0.822 -0.158]
112                   [ 1.    -1.395  0.473]
64                          [ 1.    -0.848]
32                            [ 1.   -0.84]
256  [ 1.    -0.755 -0.382  0.415 -0.114]

                                              B                C
80                                  [0.    2.537]    [ 1.   -0.04]
160                                 [0.    2.368]  [ 1.    -0.188]
112                   [ 0.     2.102  1.725 -1.602]    [1.    0.022]
64   [ 0.     2.262  2.875 -0.327 -0.13  -0.202]    [1.    0.429]
32                    [ 0.     2.239  2.843 -0.368]    [1.    0.434]
256                           [0.    1.906 2.829]    [1.    0.627]

     na  nb  nc  nk        Jv     Ji     AICv    AICCv     AICi    AICCi  \
```

```
124   2   2   4   1  5137.121  4.239  359.770  365.770  249.074  250.274
232   3   4   3   1  7611.760  4.183  379.498  388.927  250.981  252.765
296   4   3   3   1  6086.247  4.214  370.551  379.980  252.143  253.926
236   3   4   4   1  6614.672  4.221  375.882  387.437  254.401  256.523
20    1   1   2   1    12.835  5.109  112.086  113.851  270.950  271.339
24    1   1   3   1    12.067  5.067  111.619  114.164  271.633  272.182
```

```
                                        A  \
124                [ 1.    -1.767  0.773]
232         [ 1.    -2.052  1.304 -0.245]
296  [ 1.    -1.99   1.097 -0.023 -0.078]
236         [ 1.    -1.697  0.656  0.048]
20                       [ 1.    -0.825]
24                       [ 1.    -0.821]
```

```
                                              B  \
124                [ 0.     2.305  0.972 -3.096]
232  [ 0.     2.146  0.498 -3.251  1.102 -0.318]
296         [ 0.     2.219  0.536 -3.317  0.721]
236  [ 0.     2.206  1.197 -2.998 -0.028 -0.176]
20                           [0.    2.112 2.98 ]
24                           [0.    2.134 3.023]
```

```
                                        C
124  [ 1.    -0.425 -0.188 -0.447 -0.189]
232         [ 1.    -0.724 -0.069 -0.389]
296         [ 1.    -0.661 -0.185 -0.335]
236  [ 1.    -0.362 -0.221 -0.458 -0.23 ]
20                    [1.    0.554 0.404]
24               [1.    0.619 0.487 0.115]
```

5 Output Error

$$y(t) = G(q)\,u(t) + H(q)\,e(t)$$

$$G(q) = \frac{B(q)}{F(q)} \qquad H(q) = 1$$

$$B(q) = q^{-n_k}\,\left(b_0 + b_1\,q^{-1} + \cdots + b_{n_b}\,q^{-n_b}\right)$$

$$F(q) = 1 + f_1\,q^{-1} + \cdots + f_{n_f}\,q^{-n_f}$$

$$n_b = \{0,1,2,3,4\} \qquad n_f = \{1,2,3,4\} \qquad n_k = \{1,2,3,4\}$$

[ ]:
```
from functions import oe
```

```
nb_range = range(0, 4 + 1)
nf_range = range(1, 4 + 1)
nk_range = range(1, 4 + 1)

models_oe = oe(u_i, y_i, u_v, y_v, nb_range, nf_range, nk_range)
```

5.1 Display Best

[ ]:
```
from functions import display_models

columns=[
  'nb', 'nf', 'nk',
  'Jv', 'Ji',
  'AICv', 'AICCv',
  'AICi', 'AICCi',
  'B', 'F'
]

for criterion in ['Jv', 'AICCv', 'AICCi']:
  display_models(models_oe.sort_values(by=[criterion]), precision=3, qty=6,␣
  ↪columns=columns)
```

```
    nb  nf  nk       Jv      Ji     AICv    AICCv     AICi    AICCi  \
69   4   2   2  108.924  48.942  201.626  205.126  636.502  637.238
65   4   1   2  112.710  49.103  200.993  203.538  635.027  635.576
68   4   2   1  125.264  44.195  207.217  210.717  620.178  620.915
77   4   4   2  125.821  36.652  211.394  217.394  594.237  595.437
73   4   3   2  125.835  36.655  209.399  214.044  592.248  593.202
78   4   4   3  126.315  68.113  211.551  217.551  693.387  694.587

                                                      B  \
69  [ 0.000e+00  0.000e+00  7.955e+00 -6.364e+00  …
65  [ 0.      0.      7.95   -3.135   0.035   0.423 -2…
68         [ 0.      3.728 -0.551 -1.656   0.938 -1.252]
77  [  0.       0.       7.145 -15.096  10.089  -2.0…
73  [  0.       0.       7.144 -15.703  11.388  -2.8…
78  [ 0.      0.      0.     10.506 -1.954 -5.907  4…

                                      F
69                 [ 1.    -1.295  0.353]
65                        [ 1.    -0.897]
68                 [ 1.    -1.47   0.509]
77  [ 1.    -2.658  2.339 -0.596 -0.072]
73          [ 1.    -2.746  2.58  -0.823]
78  [ 1.    -0.541 -0.734  0.62  -0.219]
    nb  nf  nk       Jv      Ji     AICv    AICCv     AICi    AICCi  \
65   4   1   2  112.710  49.103  200.993  203.538  635.027  635.576
```

```
69   4   2   2  108.924  48.942  201.626  205.126  636.502  637.238
66   4   1   3  126.671  68.237  205.664  208.209  687.678  688.227
68   4   2   1  125.264  44.195  207.217  210.717  620.178  620.915
70   4   2   3  126.771  68.234  207.695  211.195  689.671  690.408
64   4   1   1  136.532  45.141  208.662  211.208  621.566  622.115
```

```
                                                    B                        F
65  [ 0.     0.     7.95  -3.135  0.035  0.423 -2…          [ 1.    -0.897]
69  [ 0.000e+00  0.000e+00  7.955e+00 -6.364e+00  …  [ 1.    -1.295  0.353]
66  [ 0.     0.     0.    10.487 -5.604  0.365 -0…          [ 1.    -0.895]
68        [ 0.     3.728 -0.551 -1.656  0.938 -1.252]  [ 1.    -1.47   0.509]
70  [ 0.     0.     0.    10.487 -5.256  0.178 -0…  [ 1.    -0.862 -0.029]
64        [ 0.     3.681  1.748 -0.588  0.952 -2.418]           [ 1.   -0.89]
```

```
    nb  nf  nk       Jv      Ji     AICv    AICCv     AICi    AICCi  \
72   4   3   1  139.654  33.552  213.567  218.212  578.094  579.048
44   2   4   1  172.372  34.054  219.986  223.486  578.474  579.211
76   4   4   1  139.650  33.531  215.566  221.566  579.993  581.193
40   2   3   1  195.958  34.973  223.116  225.661  580.733  581.282
60   3   4   1  171.498  34.030  221.783  226.428  580.361  581.314
57   3   3   2  155.141  36.544  215.773  219.273  589.765  590.501
```

```
                                                    B  \
72        [ 0.     2.959 -3.458 -1.011  1.689  0.148]
44                       [ 0.     3.369 -6.335  3.172]
76        [ 0.     2.884 -0.265 -4.806  0.786  2.043]
40                       [ 0.     4.629 -8.623  4.305]
60              [ 0.     3.005 -4.027 -0.22   1.531]
57  [  0.      0.      6.991 -14.95    9.947  -1.655]
```

```
                                    F
72         [ 1.    -2.745  2.58  -0.824]
44  [ 1.    -3.121  3.616 -1.802  0.314]
76  [ 1.    -1.775 -0.084  1.683 -0.802]
40         [ 1.    -2.759  2.61  -0.841]
60  [ 1.    -2.863  2.906 -1.131  0.098]
57         [ 1.    -2.75   2.588 -0.827]
```

6 Box-Jenkins

$$y(t) = G(q)\,u(t) + H(q)\,e(t)$$

$$G(q) = \frac{B(q)}{F(q)} \qquad H(q) = \frac{C(q)}{D(q)}$$

$$B(q) = q^{-n_k}\left(b_0 + b_1\,q^{-1} + \cdots + b_{n_b}\,q^{-n_b}\right)$$

$$C(q) = 1 + c_1\,q^{-1} + \cdots + c_{n_c}\,q^{-n_c}$$

$$D(q) = 1 + d_1\,q^{-1} + \cdots + d_{n_d}\,q^{-n_d}$$

$$F(q) = 1 + f_1\,q^{-1} + \cdots + f_{n_f}\,q^{-n_f}$$

$n_b = \{0,1,2,3,4\}$ $n_c = \{0,1,2,3,4\}$ $n_d = \{1,2,3,4\}$ $n_f = \{1,2,3,4\}$ $n_k = \{1,2,3,4\}$

[ ]:
```
from functions import bj # diversos erros

nb_range = range(0, 4 + 1)
nc_range = range(0, 4 + 1)
nd_range = range(1, 4 + 1)
nf_range = range(1, 4 + 1)
nk_range = range(1, 4 + 1)

models_bj = bj(u_i, y_i, u_v, y_v, nb_range, nc_range, nd_range, nf_range,␣
 ↪nk_range)
```

6.1 Display Best

[ ]:
```
from functions import display_models

columns=[
  'nb', 'nc', 'nd', 'nf', 'nk',
  'Jv', 'Ji',
  'AICv', 'AICCv',
  'AICi', 'AICCi',
  'B', 'C', 'D', 'F',
]

for criterion in ['Jv', 'AICCv', 'AICCi']:
  display_models(models_bj.sort_values(by=[criterion]), precision=3, qty=6,␣
 ↪columns=columns)
```

```
      nb  nc  nd  nf  nk     Jv     Ji    AICv   AICCv     AICi    AICCi  \
1308   4   0   2   4   1  5.34   5.01  89.008  98.437   279.83  281.614
980    3   0   2   2   1  5.396  5.168  83.425   88.07  278.794  279.748
976    3   0   2   1   1  5.398  5.167  81.439  84.939  276.769  277.505
```

```
664    2   0   2   3   1  5.487  5.214  84.096  88.741  280.206   281.16
668    2   0   2   4   1  5.502  5.204  86.206  92.206  281.897  283.097
660    2   0   2   2   1  5.521  5.214  82.344  85.844  278.228  278.965

                                                B     C                     D  \
1308  [0.    2.024 2.89  0.407 2.452 2.621]  [1.]  [ 1.    -1.445  0.552]
980    [ 0.     2.053  2.523 -0.667  0.171]  [1.]  [ 1.    -1.441  0.552]
976    [ 0.     2.061  2.872 -0.187  0.118]  [1.]  [ 1.    -1.44   0.551]
664           [ 0.     2.063  2.592 -0.655]  [1.]  [ 1.    -1.44   0.551]
668              [0.    2.049 4.507 2.254]  [1.]  [ 1.    -1.442  0.553]
660              [0.    2.078 3.219 0.275]  [1.]  [ 1.    -1.44   0.551]

                                          F
1308  [ 1.    -0.796  0.071  0.851 -0.77 ]
980                [ 1.    -1.002  0.141]
976                       [ 1.    -0.834]
664          [ 1.    -0.97   0.072  0.034]
668   [ 1.    -0.033 -0.6   -0.131  0.064]
660                [ 1.    -0.672 -0.139]

     nb  nc  nd  nf  nk     Jv     Ji    AICv   AICCv     AICi    AICCi  \
656   2   0   2   1   1  5.524  5.214  80.363  82.909  276.229  276.778
336   1   0   2   1   1  5.941  5.183  81.276   83.04  273.256  273.646
352   1   0   3   1   1  5.683  5.193  81.496  84.041  275.571   276.12
976   3   0   2   1   1  5.398  5.167  81.439  84.939  276.769  277.505
660   2   0   2   2   1  5.521  5.214  82.344  85.844  278.228  278.965
368   1   0   4   1   1  5.677  5.083  83.457  86.957  274.137  274.874

                                          B     C  \
656         [ 0.     2.083  2.874 -0.198]  [1.]
336                  [0.    2.052 2.863]  [1.]
352                  [0.    2.039 2.825]  [1.]
976  [ 0.     2.061  2.872 -0.187  0.118]  [1.]
660            [0.    2.078 3.219 0.275]  [1.]
368                  [0.    2.057 2.86 ]  [1.]

                                       D                     F
656               [ 1.    -1.44   0.551]         [ 1.    -0.838]
336               [ 1.    -1.439  0.55 ]         [ 1.    -0.832]
352         [ 1.    -1.403  0.452  0.069]         [ 1.    -0.835]
976               [ 1.    -1.44   0.551]         [ 1.    -0.834]
660               [ 1.    -1.44   0.551]  [ 1.    -0.672 -0.139]
368  [ 1.    -1.41   0.413  0.198 -0.094]         [ 1.    -0.832]

      nb  nc  nd  nf  nk     Jv     Ji     AICv    AICCv     AICi    AICCi  \
1336   4   0   4   3   1   9.15  4.635  112.552  124.107  269.396  271.519
1016   3   0   4   3   1  9.249  4.756  110.981   120.41  271.499  273.283
984    3   0   2   3   1  7.272    4.9   97.359  103.359   272.29   273.49
336    1   0   2   1   1  5.941  5.183   81.276    83.04  273.256  273.646
```

```
1000   3   0   3   3   1  8.463  4.853  105.426  113.012  272.735  274.212
368    1   0   4   1   1  5.677  5.083   83.457   86.957  274.137  274.874

                                                B     C  \
1336  [0.    2.153 2.096 0.911 3.028 0.013]  [1.]
1016        [0.    2.153 2.096 0.899 3.034]  [1.]
984         [0.    2.172 2.101 0.918 3.037]  [1.]
336                   [0.    2.052 2.863]  [1.]
1000        [0.    2.153 2.066 0.909 2.997]  [1.]
368                   [0.    2.057 2.86 ]  [1.]

                                              D                                F
1336  [ 1.    -1.407  0.341  0.302 -0.131]  [ 1.    -1.158  1.262 -0.823]
1016  [ 1.    -1.407  0.341  0.302 -0.131]  [ 1.    -1.159  1.262 -0.824]
984                [ 1.    -1.456  0.563]  [ 1.    -1.158  1.262 -0.824]
336                [ 1.    -1.439  0.55 ]                [ 1.    -0.832]
1000         [ 1.    -1.39   0.39   0.121]  [ 1.    -1.16   1.262 -0.826]
368   [ 1.    -1.41   0.413  0.198 -0.094]                [ 1.    -0.832]
```

7 Results

[ ]:
```
from functions import models_frame

models = pd.concat([models_frame(), models_arx, models_armax, models_oe,␣
 ↪models_bj], ignore_index=True)

print('Successful models:', len(models.loc[models.B.notnull()]))
print('Failed models:    ', len(models.loc[models.B.isnull()]))
```

```
Successful models: 1120
Failed models:     960
```

7.1 Display Models with Lowest Cost

[ ]:
```
from functions import display_models
from control import frequency_response, mag2db

columns=[
  'model',
  'na', 'nb', 'nc', 'nd', 'nf', 'nk',
  'Jv', 'Ji',
  'AICv', 'AICCv',
  'AICi', 'AICCi',
]

qty = 6
for criterion in ['Jv', 'AICCv', 'AICCi']:
```

```
display_models(models.sort_values(by=[criterion]), precision=3, qty=qty,␣
↪columns=columns)
```

```
      model na  nb nc nd nf  nk     Jv     Ji    AICv   AICCv     AICi  \
56      ARX  3   4  -  -  -   1  5.133  5.296  81.427  86.072  282.699
36      ARX  2   4  -  -  -   1  5.179  5.297  79.783  83.283  280.747
144   ARMAX  1   4  1  -  -   1  5.199  5.922  79.935  83.435  298.594
224   ARMAX  2   4  1  -  -   1   5.25  5.299  82.332  86.977  282.814
336   ARMAX  4   1  1  -  -   1  5.277  5.337  80.531  84.031  281.932
1788     BJ  -   4  0  2  4   1   5.34   5.01  89.008  98.437   279.83

        AICCi
56    283.653
36    281.484
144    299.33
224   283.768
336   282.669
1788  281.614
```

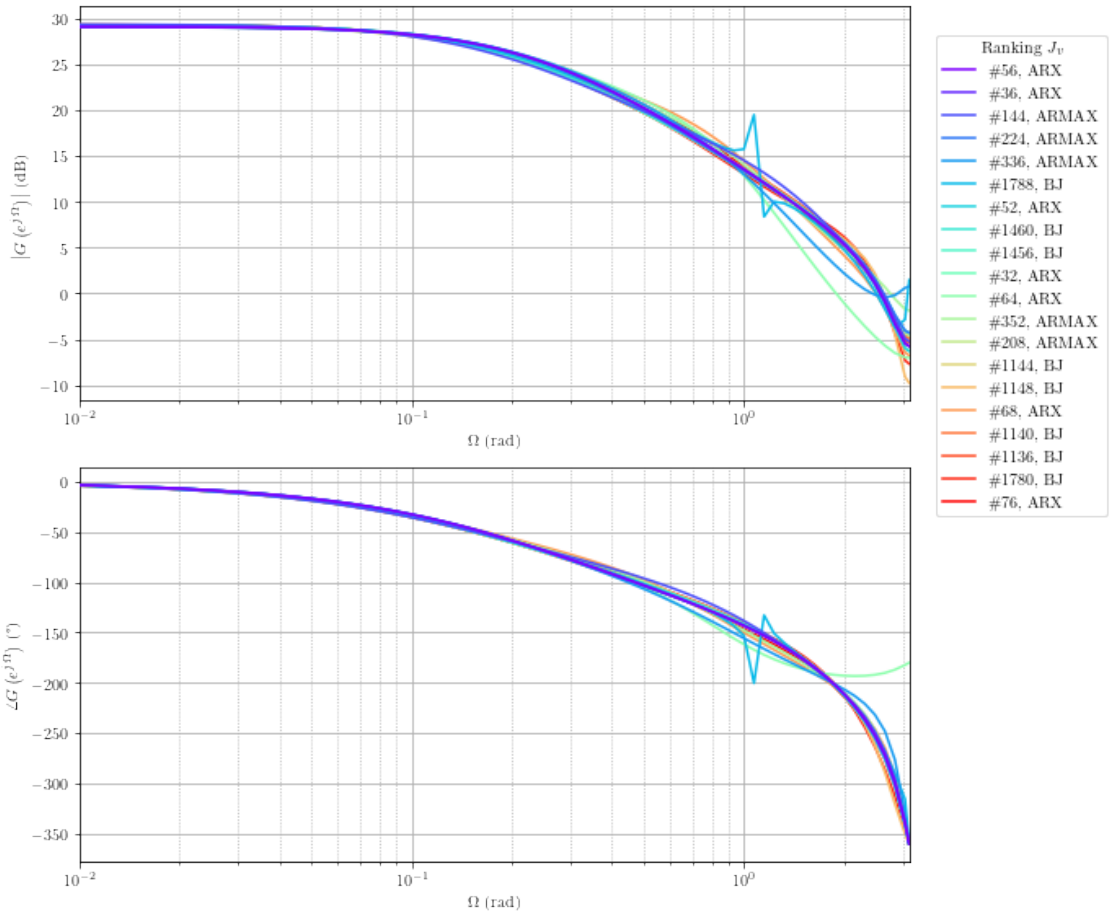
```
   model na  nb nc nd nf  nk     Jv     Ji    AICv   AICCv     AICi    AICCi
24   ARX  2   1  -  -  -   1  5.801   5.75   78.32  79.463  287.883  288.142
44   ARX  3   1  -  -  -   1  5.556   5.74  78.594  80.358  289.581  289.971
28   ARX  2   2  -  -  -   1  5.589  5.276  78.832  80.596  276.114  276.503
20   ARX  2   0  -  -  -   1  6.384   5.91  80.153   80.82  290.269  290.423
40   ARX  3   0  -  -  -   1  6.074  5.875   80.16  81.303  291.314  291.572
32   ARX  2   3  -  -  -   1   5.41  5.264  79.531  82.077   277.74  278.289
```

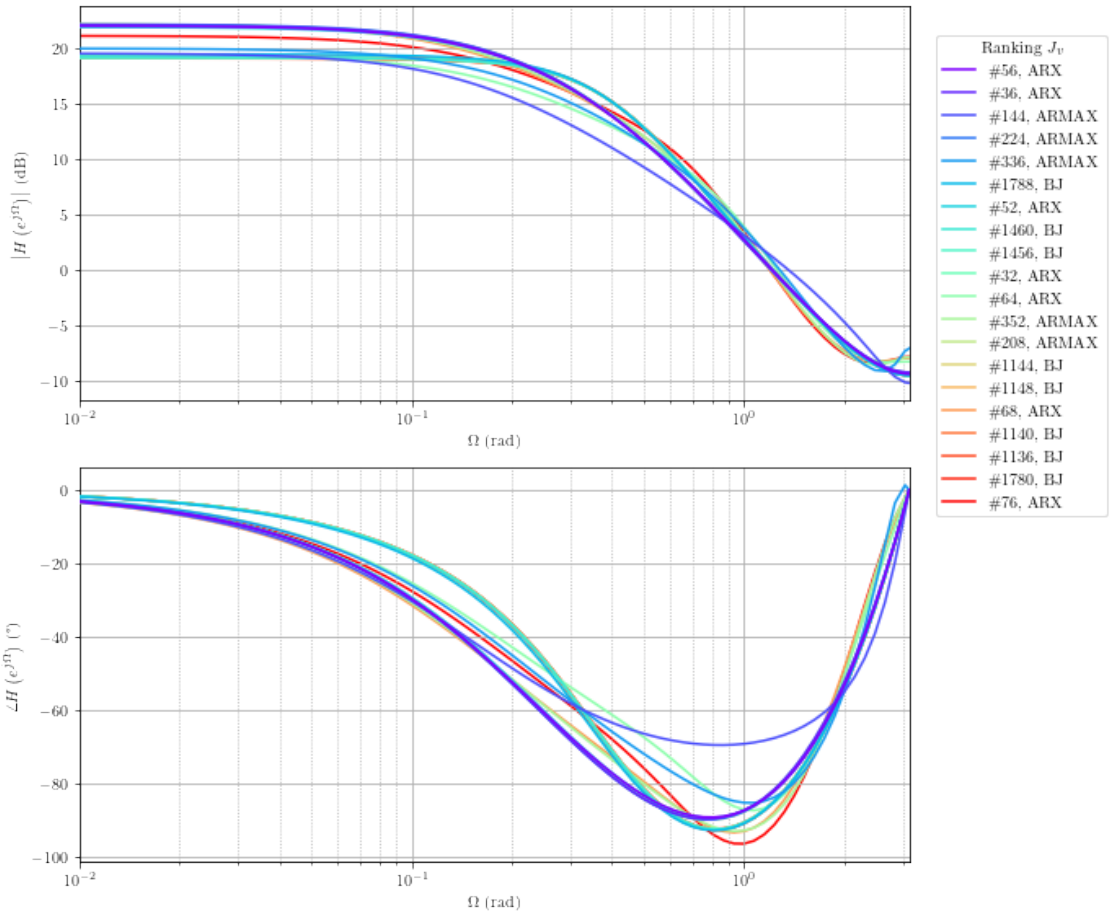
```
      model na  nb nc nd nf  nk        Jv     Ji     AICv    AICCv     AICi  \
204   ARMAX  2   2  4  -  -   1  5137.121  4.239   359.77   365.77  249.074
312   ARMAX  3   4  3  -  -   1   7611.76  4.183  379.498  388.927  250.981
376   ARMAX  4   3  3  -  -   1  6086.247  4.214  370.551   379.98  252.143
316   ARMAX  3   4  4  -  -   1  6614.672  4.221  375.882  387.437  254.401
100   ARMAX  1   1  2  -  -   1    12.835  5.109  112.086  113.851   270.95
1816     BJ  -   4  0  4  3   1      9.15  4.635  112.552  124.107  269.396

        AICCi
204   250.274
312   252.765
376   253.926
316   256.523
100   271.339
1816  271.519
```

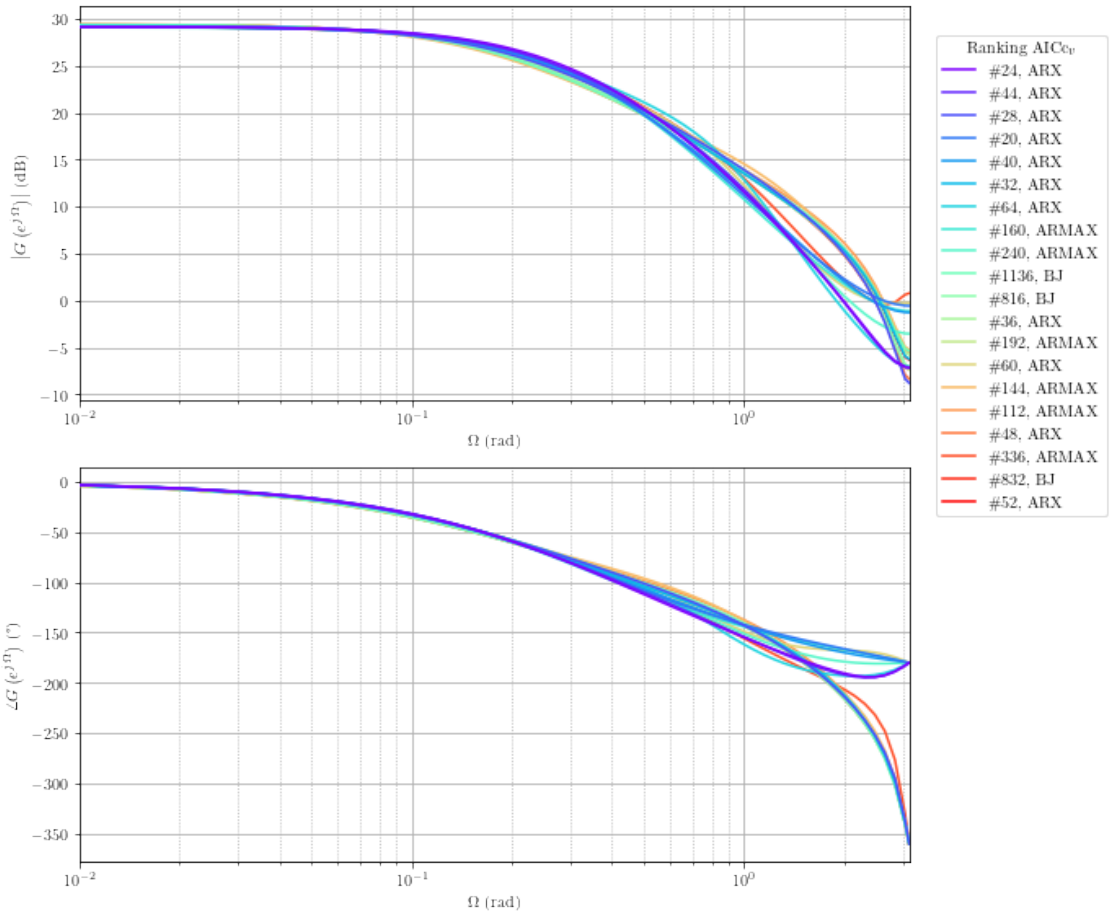
7.2 Frequency Response

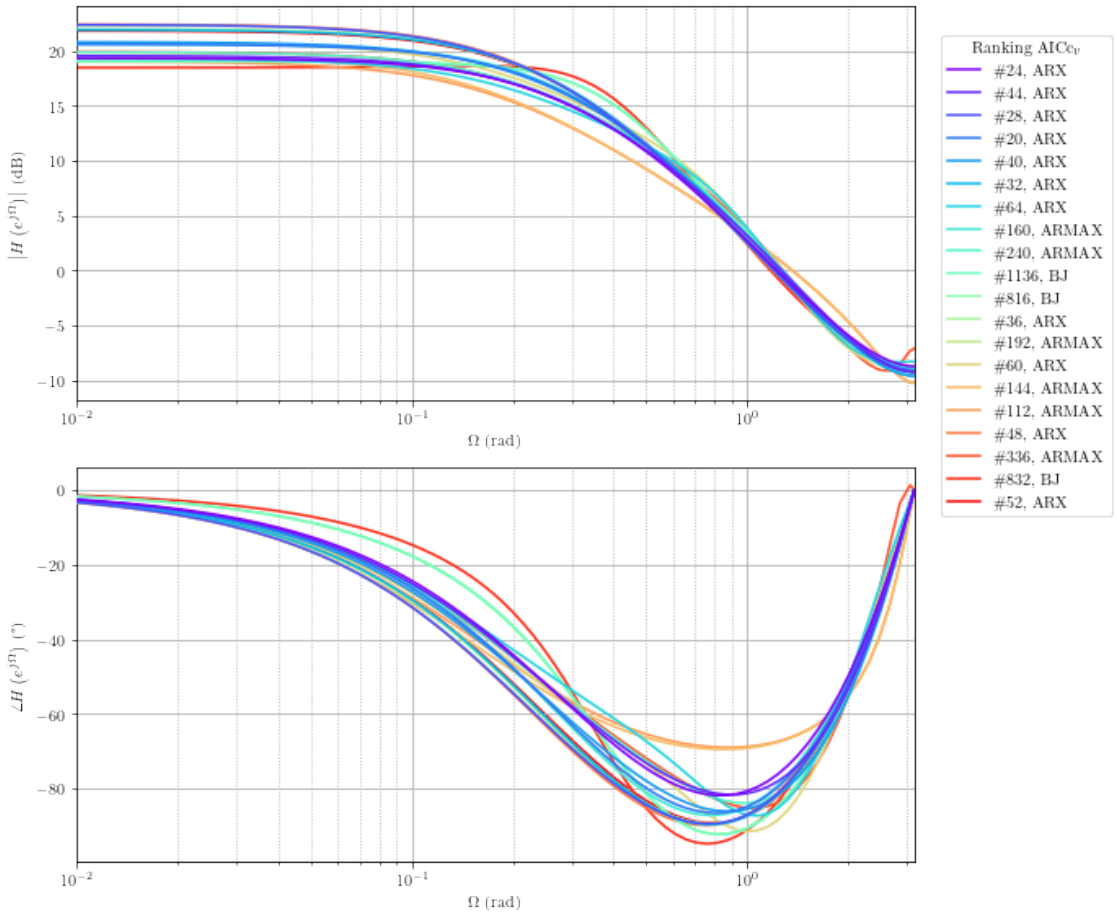
[ ]:
```
logspace = np.logspace(-2, 1, 100)
logspace = np.append(logspace[logspace < np.pi], np.pi)
```

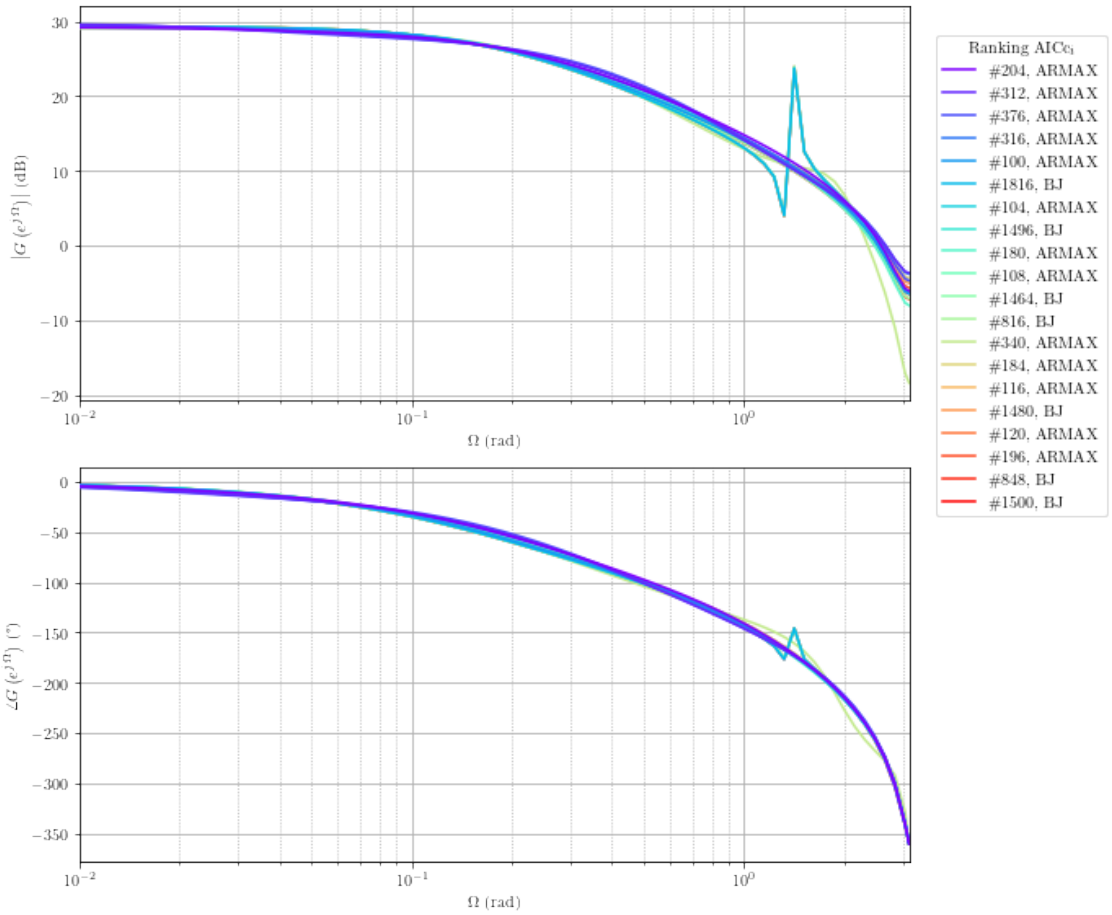
```
costs = {
  'Jv': '$J_v$',
  'Ji': '$J_i$',
  'AICv': 'AIC${{}}_v$',
  'AICi': 'AIC${{}}_i$',
  'AICCv': 'AICc${{}}_v$',
  'AICCi': 'AICc${{}}_i$',
}

qty = 20
for criterion in ['Jv', 'AICCv', 'AICCi']:
  for tf in ['G', 'H']:
    fig, axs = plt.subplots(2, 1, figsize=(8,8))
    colors = iter(plt.cm.rainbow(np.linspace(0, 1, qty)))
    for i, (index, model) in enumerate(models.sort_values(by=[criterion]).
 ↪iterrows()):
      if i >= qty:
        break
      color = next(colors)
      mag, phase, omega = frequency_response(model[tf], omega=logspace)
      axs[0].plot(omega, mag2db(mag), c=color, zorder=qty-i)
      axs[1].plot(omega, 180/np.pi*np.unwrap(phase), c=color, zorder=qty-i,␣
 ↪label=f"\\#{index}, {model.model}")

    for ax in axs:
      ax.set_xscale('log')
      ax.set_xlim(omega[0], omega[-1])
      ax.grid(which='major')
      ax.grid(which='minor', linestyle=':')
      ax.set_xlabel('$\\Omega$ (rad)')
    # axs[0].set_title(f'${tf}(q)$')
    axs[0].set_ylabel(f'$\\left| {tf}\\left(e^{{j \\, \\Omega}}\\right)␣
 ↪\\right|$ (dB)')
    axs[1].set_ylabel(f'$\\angle{{{tf}\\left(e^{{j \\, \\Omega}}\\right)}}$␣
 ↪(°)')
    leg = fig.legend(title=f'Ranking {costs[criterion]}', bbox_to_anchor=(1.2,␣
 ↪0.96))
    plt.tight_layout()
    plt.savefig(figPath + f'bode_{tf}_{criterion}.' + figExt, format=figExt,␣
 ↪bbox_extra_artists=(leg,), bbox_inches='tight')
    plt.show()
```

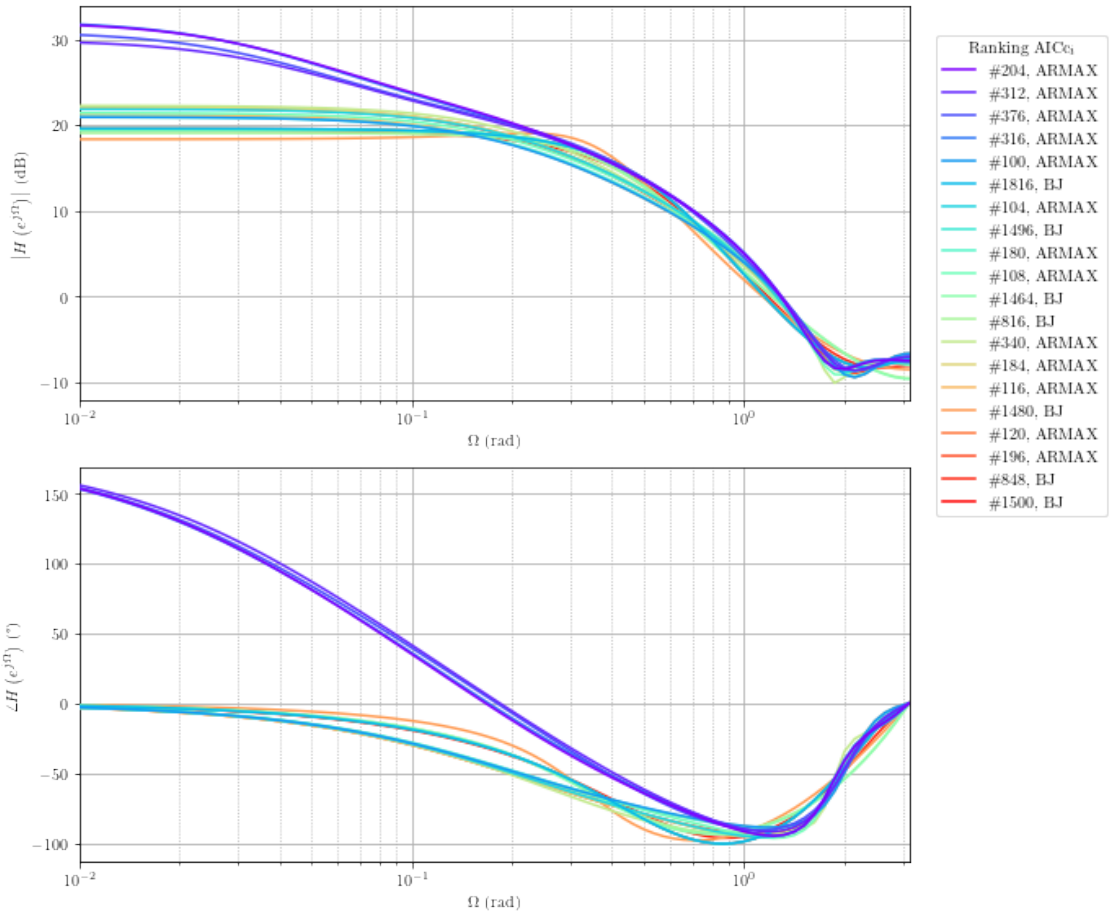

Ranking J_v
#56, ARX
#36, ARX
#144, ARMAX
#224, ARMAX
#336, ARMAX
#1788, BJ
#52, ARX
#1460, BJ
#1456, BJ
#32, ARX
#64, ARX
#352, ARMAX
#208, ARMAX
#1144, BJ
#1148, BJ
#68, ARX
#1140, BJ
#1136, BJ
#1780, BJ
#76, ARX
$|G(e^{j\Omega})|$ (dB)
$\angle G(e^{j\Omega})$ (°)
Ω (rad)
Ω (rad)


Ranking J_v
#56, ARX
#36, ARX
#144, ARMAX
#224, ARMAX
#336, ARMAX
#1788, BJ
#52, ARX
#1460, BJ
#1456, BJ
#32, ARX
#64, ARX
#352, ARMAX
#208, ARMAX
#1144, BJ
#1148, BJ
#68, ARX
#1140, BJ
#1136, BJ
#1780, BJ
#76, ARX
$|H(e^{j\Omega})|$ (dB)
$\angle H(e^{j\Omega})$ (°)
Ω (rad)
Ω (rad)


Ranking AICc$_v$
#24, ARX
#44, ARX
#28, ARX
#20, ARX
#40, ARX
#32, ARX
#64, ARX
#160, ARMAX
#240, ARMAX
#1136, BJ
#816, BJ
#36, ARX
#192, ARMAX
#60, ARX
#144, ARMAX
#112, ARMAX
#48, ARX
#336, ARMAX
#832, BJ
#52, ARX
$|G(e^{j\Omega})|$ (dB)
$\angle G(e^{j\Omega})$ (°)
Ω (rad)
Ω (rad)


Ranking AICc_v
#24, ARX
#44, ARX
#28, ARX
#20, ARX
#40, ARX
#32, ARX
#64, ARX
#160, ARMAX
#240, ARMAX
#1136, BJ
#816, BJ
#36, ARX
#192, ARMAX
#60, ARX
#144, ARMAX
#112, ARMAX
#48, ARX
#336, ARMAX
#832, BJ
#52, ARX
|H (e^jΩ)| (dB)
∠H (e^jΩ) (°)
Ω (rad)
Ω (rad)

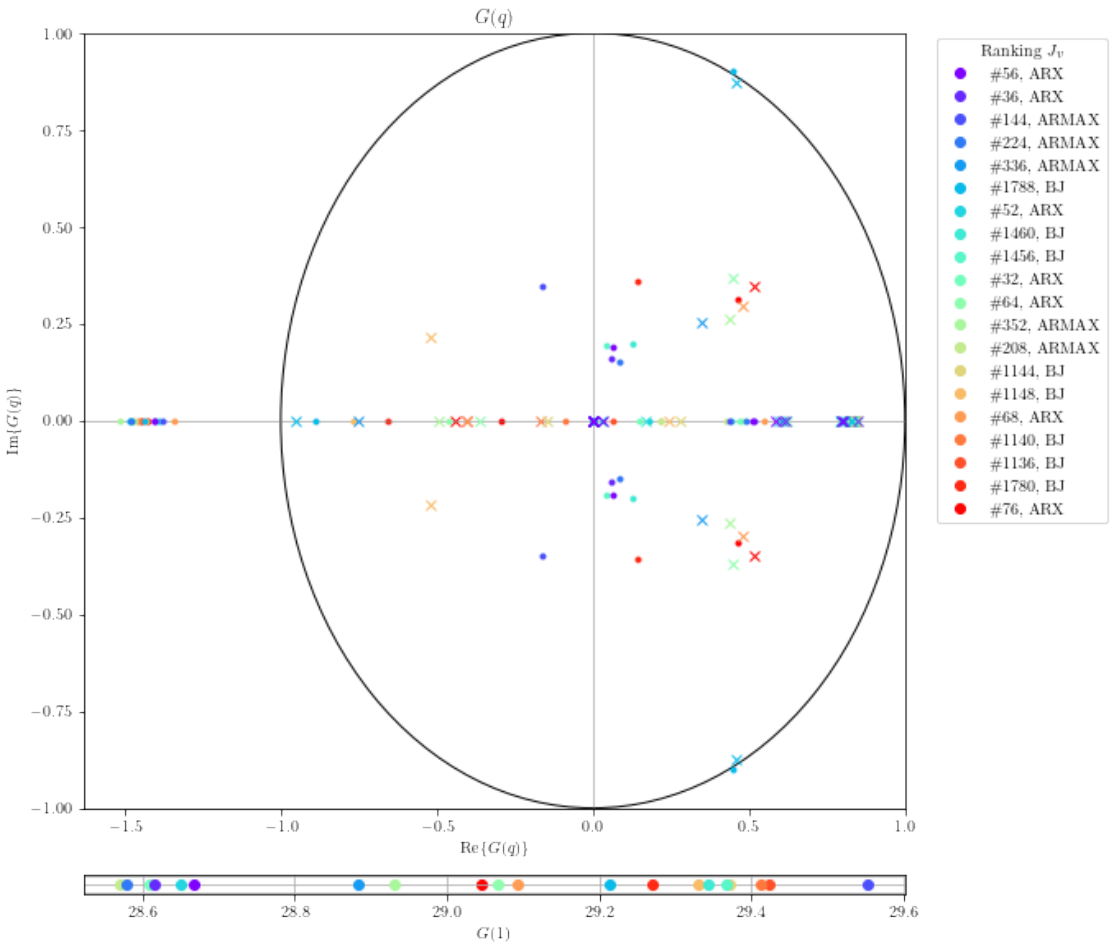

Ranking AICc
#204, ARMAX
#312, ARMAX
#376, ARMAX
#316, ARMAX
#100, ARMAX
#1816, BJ
#104, ARMAX
#1496, BJ
#180, ARMAX
#108, ARMAX
#1464, BJ
#816, BJ
#340, ARMAX
#184, ARMAX
#116, ARMAX
#1480, BJ
#120, ARMAX
#196, ARMAX
#848, BJ
#1500, BJ
|G (e^jΩ)| (dB)
∠G (e^jΩ) (°)
Ω (rad)
Ω (rad)

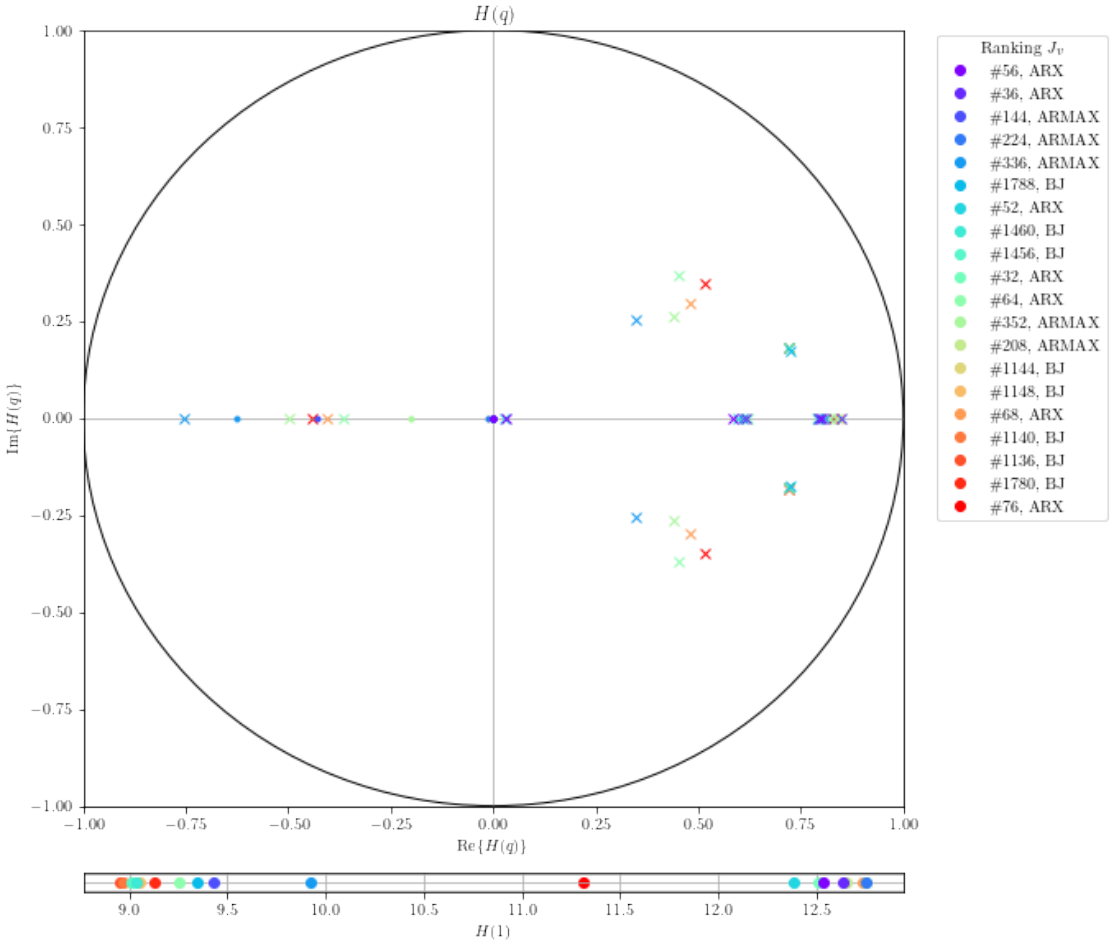


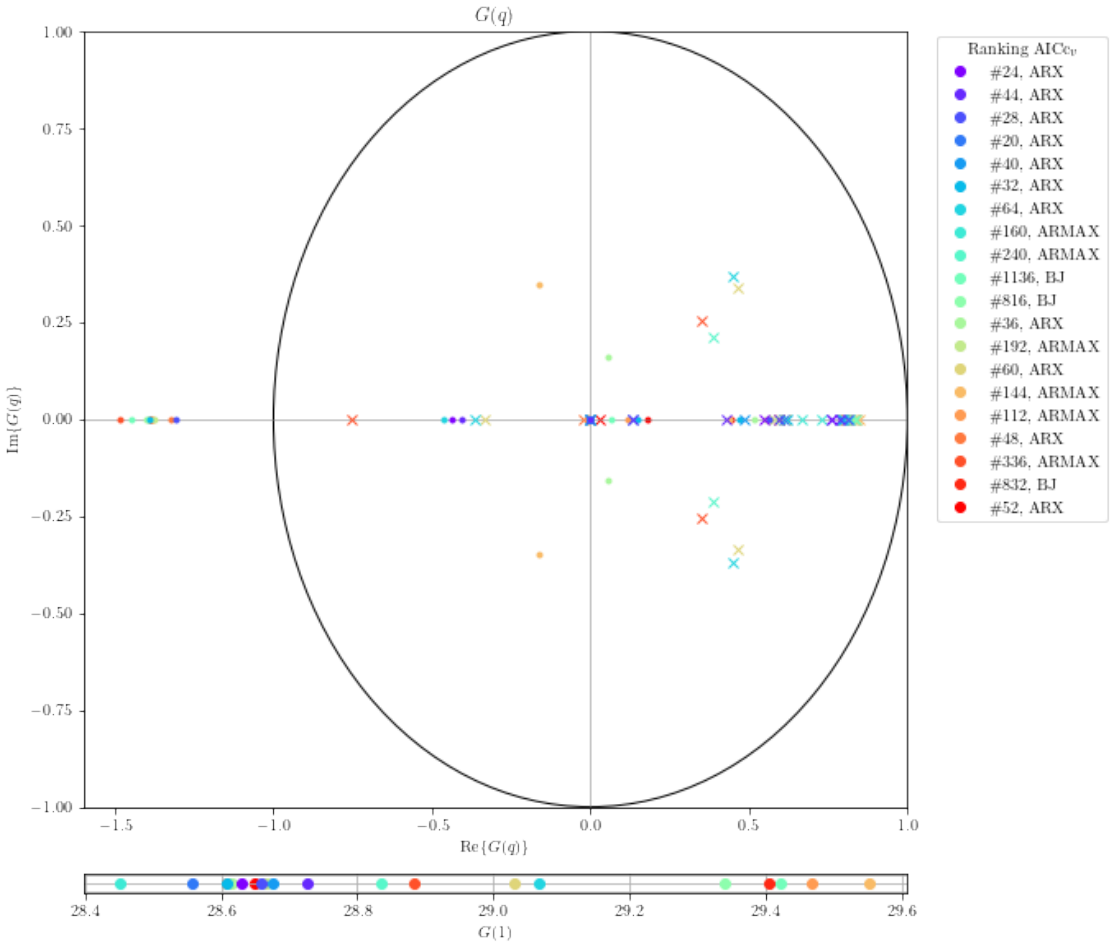
7.3 Pole Zero Map

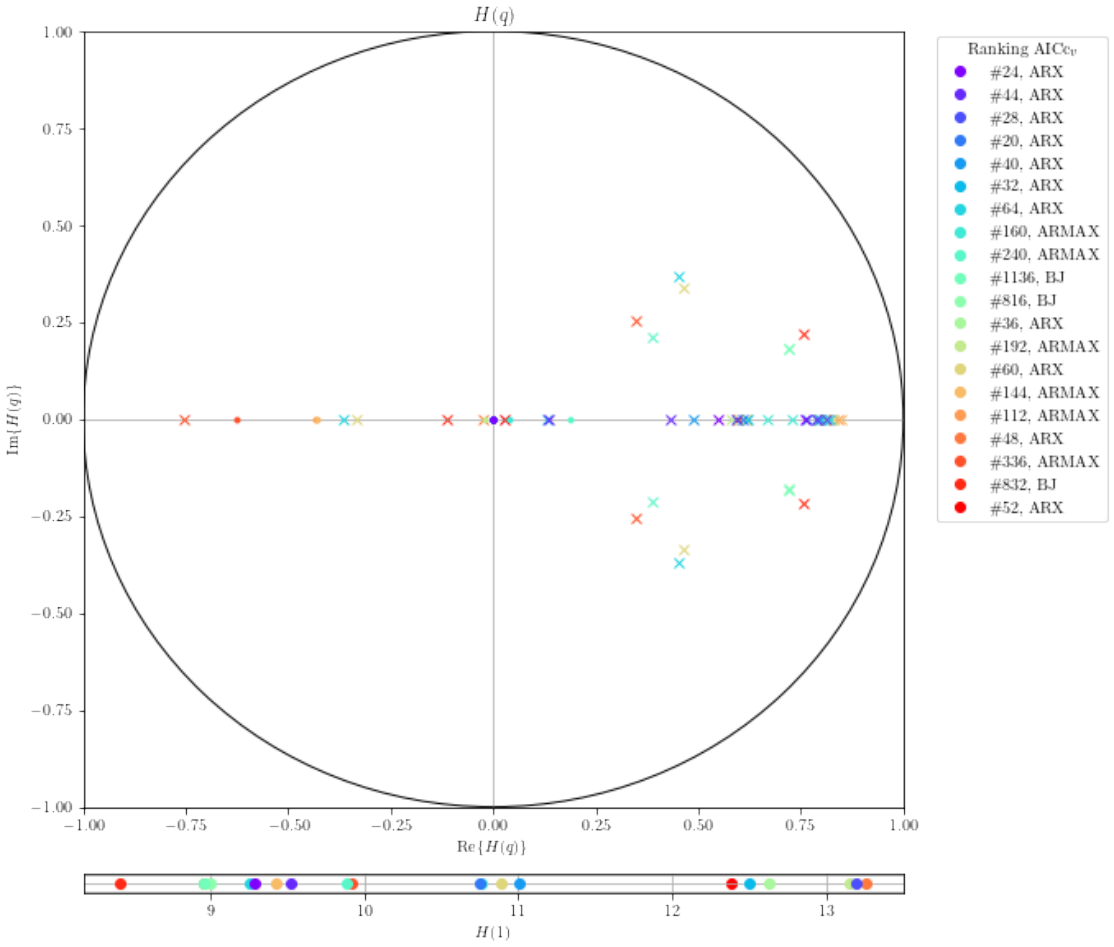
[ ]:
```
qty = 20
for criterion in ['Jv', 'AICCv', 'AICCi']:
  for tf in ['G', 'H']:
    fig, ax = plt.subplots(2, 1, figsize=(8,8.2), height_ratios=[8, 0.2])
    colors = iter(plt.cm.rainbow(np.linspace(0, 1, qty)))
    for i, (index, model) in enumerate(models.sort_values(by=[criterion]).
 ↪iterrows()):
      if i >= qty:
        break
      color = next(colors)
      for pole in model[f'p{tf}']:
        ax[0].plot(pole.real, pole.imag, 'x', c=color, zorder=qty-i)
      for zero in model[f'z{tf}']:
        ax[0].plot(zero.real, zero.imag, '.', c=color, zorder=qty-i)
      ax[1].plot(model[f'k{tf}'], 0, 'o', c=color, zorder=qty-i,␣
 ↪label=f"\\#{index}, {model.model}")
```

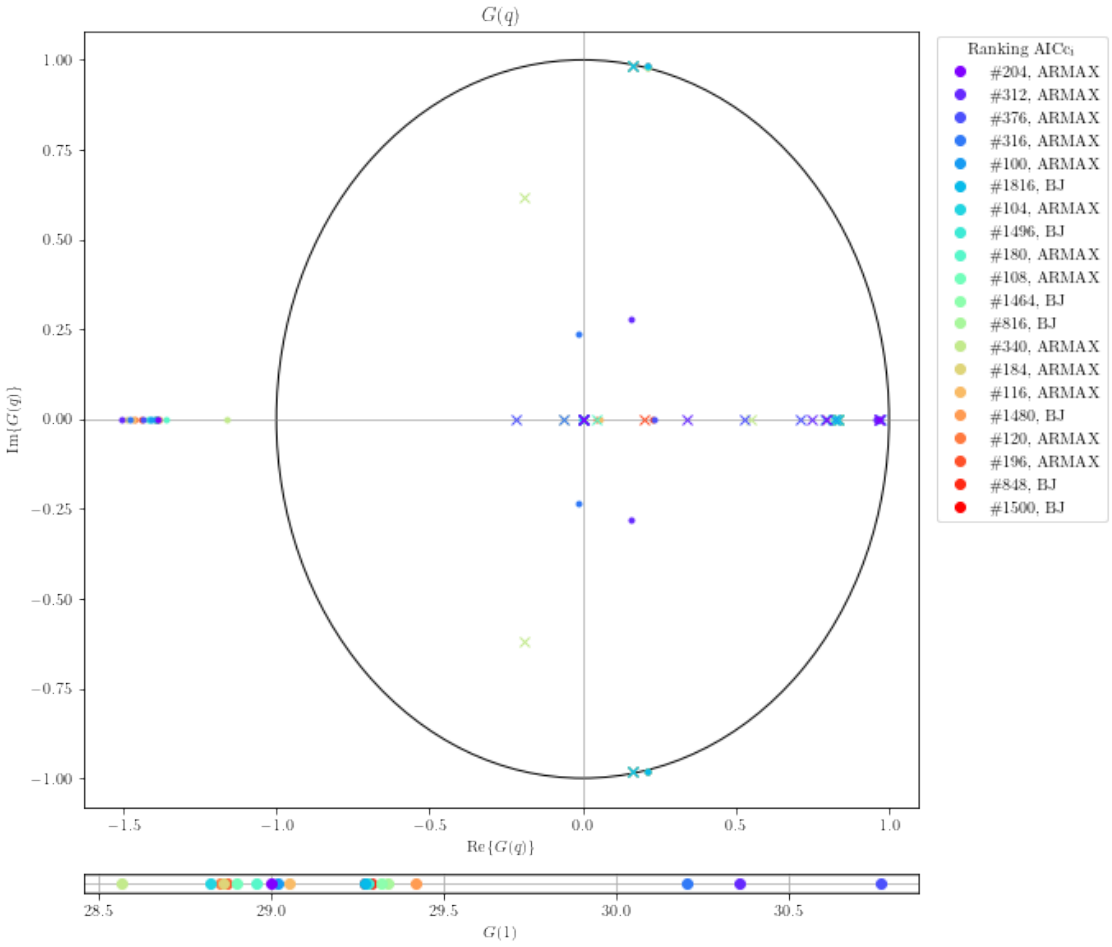
```
ax[0].set_title(f'${tf}(q)$')
ax[1].set_xlabel(f'${tf}(1)$')
ax[0].add_artist(plt.Circle((0, 0), 1, fill=False))
ax[0].set_xlabel(f'$\\mathrm{{Re}}\\{{{tf}(q)\\}}$')
ax[0].set_ylabel(f'$\\mathrm{{Im}}\\{{{tf}(q)\\}}$')
xlim = ax[0].get_xlim()
ylim = ax[0].get_ylim()
ax[0].set_xlim(min(xlim[0], -1), max(xlim[1], 1))
ax[0].set_ylim(min(ylim[0], -1), max(ylim[1], 1))
ax[0].axhline(0, color='gray', linewidth=0.5)
ax[0].axvline(0, color='gray', linewidth=0.5)
ax[1].tick_params(axis='y',left=False, labelleft=False)
ax[1].grid()
fig.legend(title=f'Ranking {costs[criterion]}', bbox_to_anchor=(1.19, 0.96))
plt.tight_layout()
plt.savefig(figPath + f'zpk_{tf}_{criterion}.' + figExt, format=figExt)
plt.show()
```

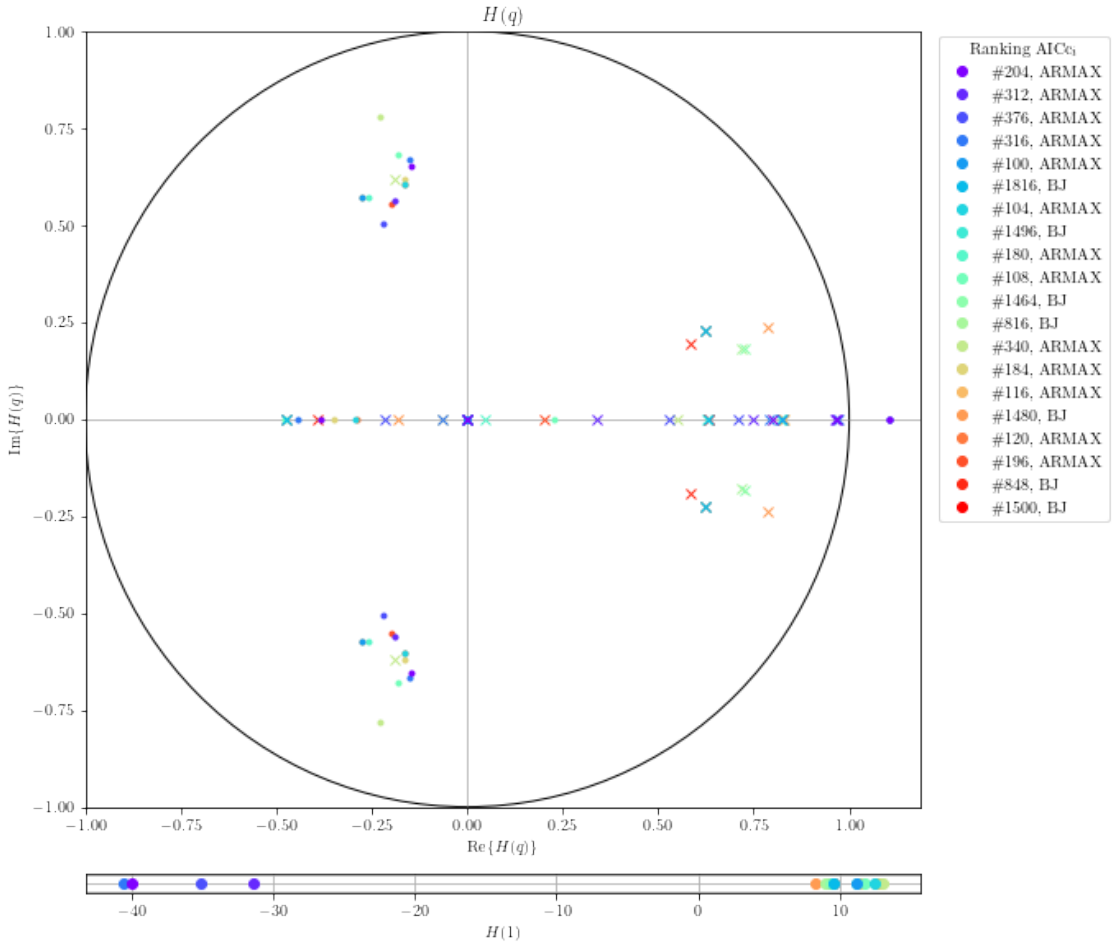



$H(q)$
Ranking J_v
#56, ARX
#36, ARX
#144, ARMAX
#224, ARMAX
#336, ARMAX
#1788, BJ
#52, ARX
#1460, BJ
#1456, BJ
#32, ARX
#64, ARX
#352, ARMAX
#208, ARMAX
#1144, BJ
#1148, BJ
#68, ARX
#1140, BJ
#1136, BJ
#1780, BJ
#76, ARX
$\mathrm{Im}\{H(q)\}$
$\mathrm{Re}\{H(q)\}$
$H(1)$


$G(q)$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
$\mathrm{Im}\{G(q)\}$
−1.5
−1.0
−0.5
0.0
0.5
1.0
$\mathrm{Re}\{G(q)\}$
28.4
28.6
28.8
29.0
29.2
29.4
29.6
$G(1)$
Ranking $\mathrm{AICc_v}$
#24, ARX
#44, ARX
#28, ARX
#20, ARX
#40, ARX
#32, ARX
#64, ARX
#160, ARMAX
#240, ARMAX
#1136, BJ
#816, BJ
#36, ARX
#192, ARMAX
#60, ARX
#144, ARMAX
#112, ARMAX
#48, ARX
#336, ARMAX
#832, BJ
#52, ARX


$H(q)$
Ranking AICc$_v$
#24, ARX
#44, ARX
#28, ARX
#20, ARX
#40, ARX
#32, ARX
#64, ARX
#160, ARMAX
#240, ARMAX
#1136, BJ
#816, BJ
#36, ARX
#192, ARMAX
#60, ARX
#144, ARMAX
#112, ARMAX
#48, ARX
#336, ARMAX
#832, BJ
#52, ARX
$\mathrm{Im}\{H(q)\}$
$\mathrm{Re}\{H(q)\}$
$H(1)$

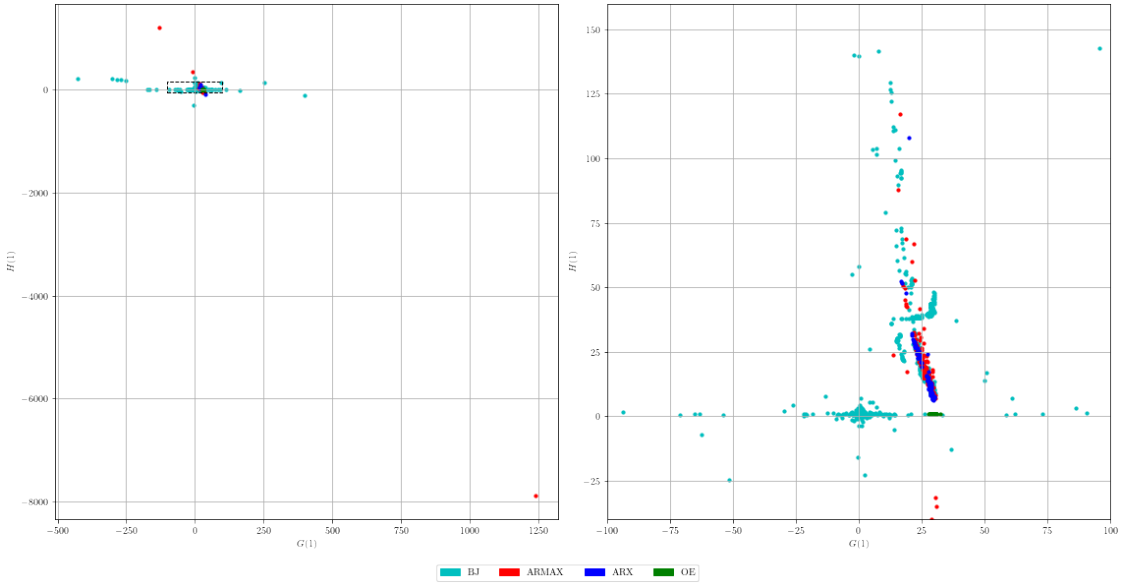

$G(q)$
Ranking AICc$_1$
#204, ARMAX
#312, ARMAX
#376, ARMAX
#316, ARMAX
#100, ARMAX
#1816, BJ
#104, ARMAX
#1496, BJ
#180, ARMAX
#108, ARMAX
#1464, BJ
#816, BJ
#340, ARMAX
#184, ARMAX
#116, ARMAX
#1480, BJ
#120, ARMAX
#196, ARMAX
#848, BJ
#1500, BJ
$\mathrm{Im}\{G(q)\}$
$\mathrm{Re}\{G(q)\}$
$G(1)$



7.4 Static Gains Scatter

[ ]:
```
import matplotlib.patches as mpatches

fig, ax = plt.subplots(1, 2, figsize=(16,8))

for i in range(0, 1+1):
  ax[i].scatter(models.loc[models.model == 'BJ'   ].kG, models.loc[models.model␣
 ↪== 'BJ'   ].kH, s=10, color='c')
  ax[i].scatter(models.loc[models.model == 'ARMAX'].kG, models.loc[models.model␣
 ↪== 'ARMAX'].kH, s=10, color='r')
  ax[i].scatter(models.loc[models.model == 'ARX'  ].kG, models.loc[models.model␣
 ↪== 'ARX'  ].kH, s=10, color='b')
  ax[i].scatter(models.loc[models.model == 'OE'   ].kG, models.loc[models.model␣
 ↪== 'OE'   ].kH, s=10, color='g')
  ax[i].set_xlabel('$G(1)$')
  ax[i].set_ylabel('$H(1)$')
```

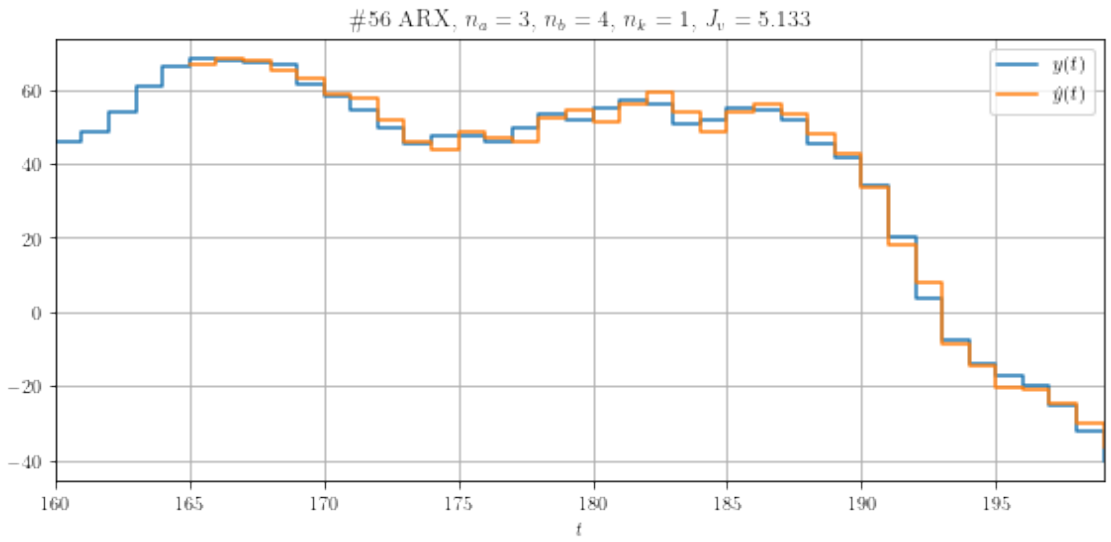
```
  ax[i].grid()

ax[0].add_patch(plt.Rectangle((-100,-40), 200, 200, fill=False, linestyle='--'))

ax[1].set_xlim((-100, 100))
ax[1].set_ylim((-40, 160))

fig.legend(ncols=4, bbox_to_anchor=(0.625, 0), handles=[
  mpatches.Patch(label='BJ',    color='c'),
  mpatches.Patch(label='ARMAX', color='r'),
  mpatches.Patch(label='ARX',   color='b'),
  mpatches.Patch(label='OE',    color='g'),
])
plt.tight_layout()
plt.show()
```

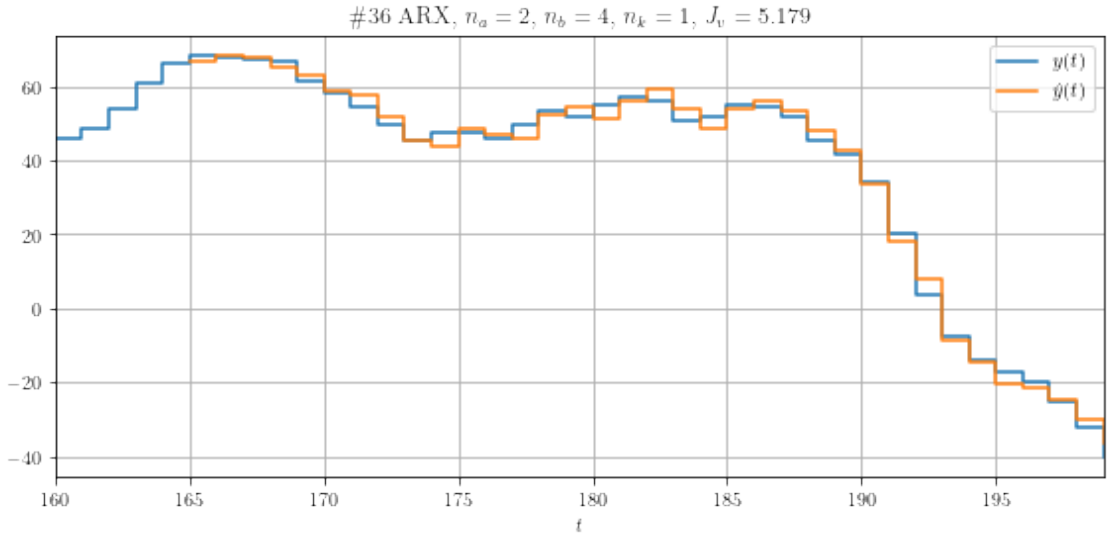


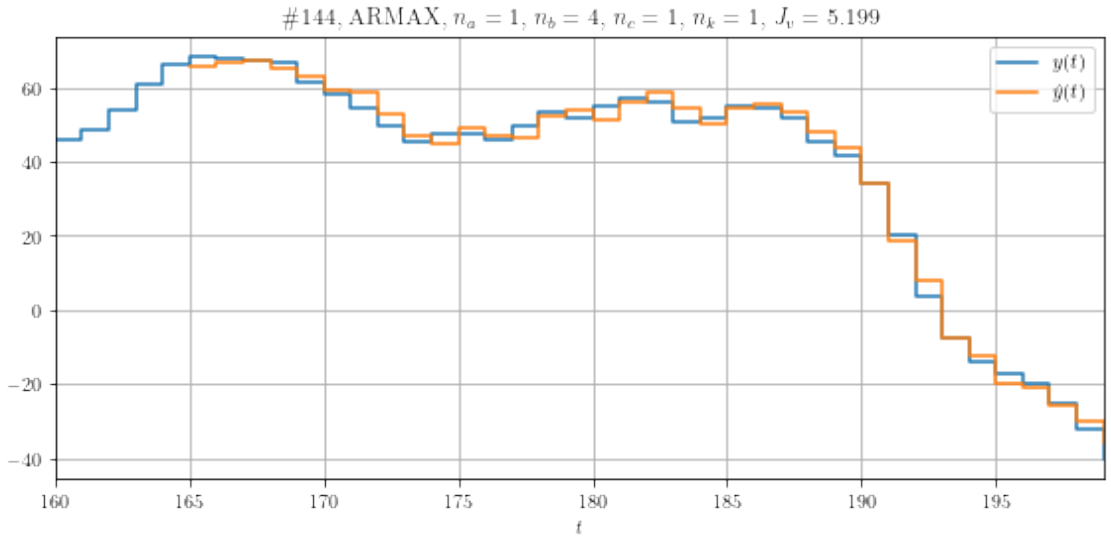
7.5 Predictions

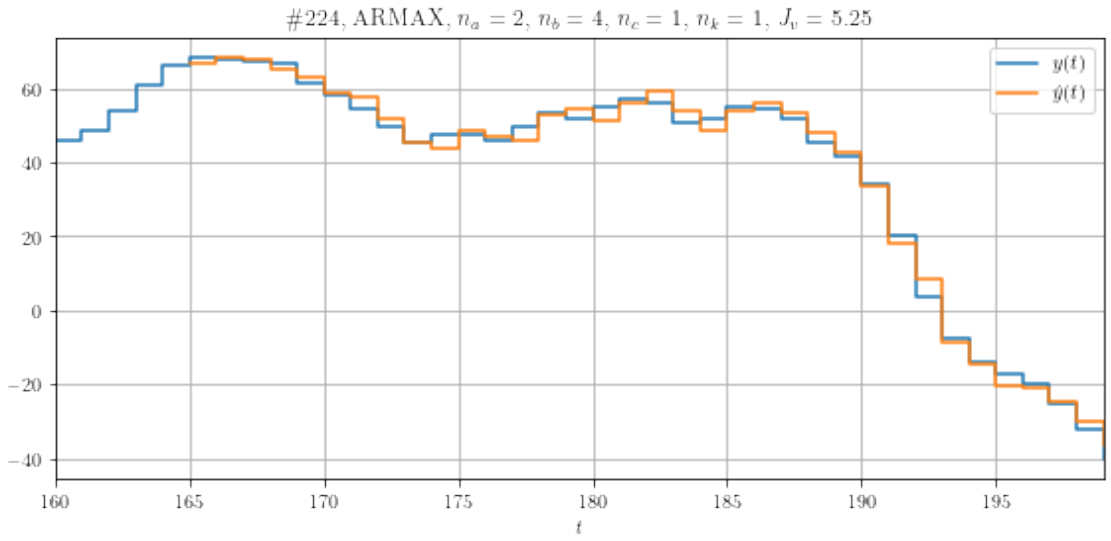
[ ]:
```
qty = 5

for i, (index, model) in enumerate(models.sort_values(by=['Jv']).iterrows()):
  if i >= qty:
    break

  if np.isnan(model.yp).any():
    continue
```

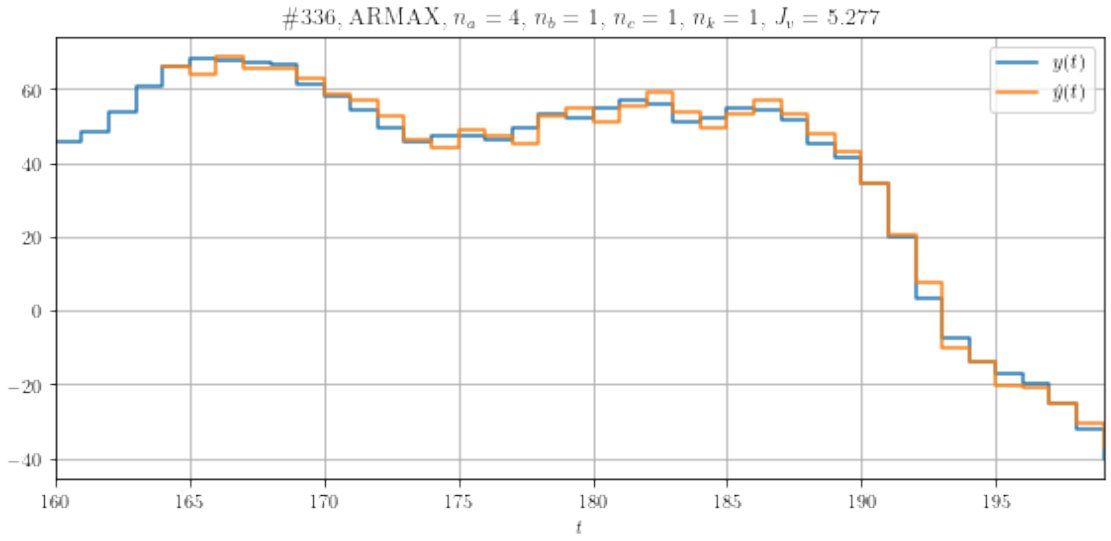
```
if model.model == 'ARX':
  title = f'\\#{index} {model.model}, $n_a={model.na}$, $n_b={model.nb}$,␣
↪$n_k={model.nk}$, $J_v={model.Jv:.4g}$'
elif model.model == 'ARMAX':
  title = f'\\#{index}, {model.model}, $n_a={model.na}$, $n_b={model.nb}$,␣
↪$n_c={model.nc}$, $n_k={model.nk}$, $J_v={model.Jv:.4g}$'
elif model.model == 'OE':
  title = f'\\#{index}, {model.model}, $n_b={model.nb}$, $n_f={model.nf}$,␣
↪$n_k={model.nk}$, $J_v={model.Jv:.4g}$'
elif model.model == 'BJ':
  title = f'\\#{index}, {model.model}, $n_b={model.nb}$, $n_c={model.nc}$,␣
↪$n_d={model.nd}$, $n_f={model.nf}$, $n_k={model.nk}$, $J_v={model.Jv:.4g}$'
else:
  assert(False)

plt.figure(figsize=(8,4))
plt.title(title)
plt.plot(t_v, y_v, label='$y(t)$', drawstyle='steps-post')
plt.plot(t_v[int(model.delay):], model.yp, label='$\\hat{y}(t)$',␣
↪drawstyle='steps-post')
plt.xlim(t_v[0], t_v[-1])
plt.xlabel('$t$')
plt.grid()
plt.legend()
plt.tight_layout()
plt.show()
```




#36 ARX, $n_a = 2$, $n_b = 4$, $n_k = 1$, $J_v = 5.179$
$y(t)$
$\hat{y}(t)$
60
40
20
0
−20
−40
160
165
170
175
180
185
190
195
t

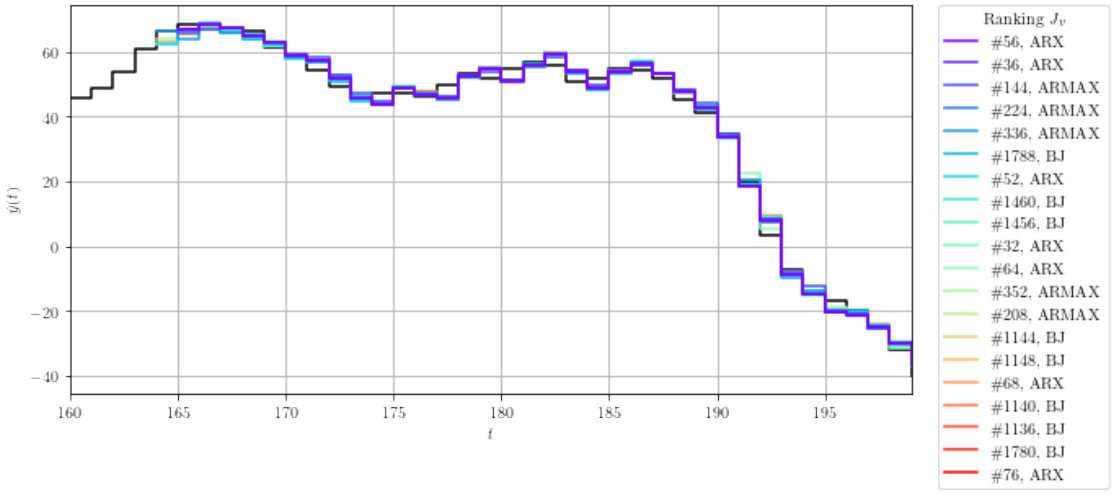

#144, ARMAX, $n_a = 1$, $n_b = 4$, $n_c = 1$, $n_k = 1$, $J_v = 5.199$
$y(t)$
$\hat{y}(t)$
60
40
20
0
−20
−40
160
165
170
175
180
185
190
195
t

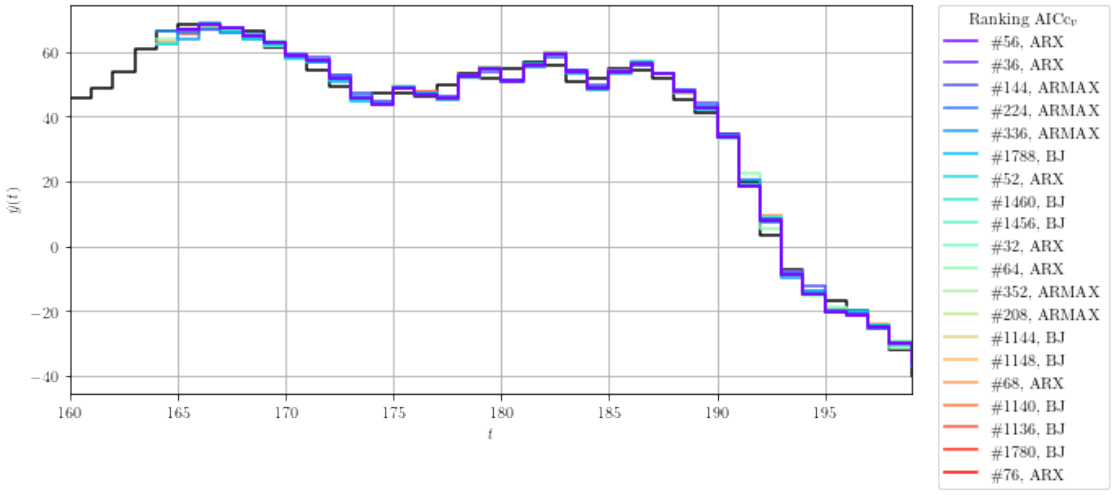


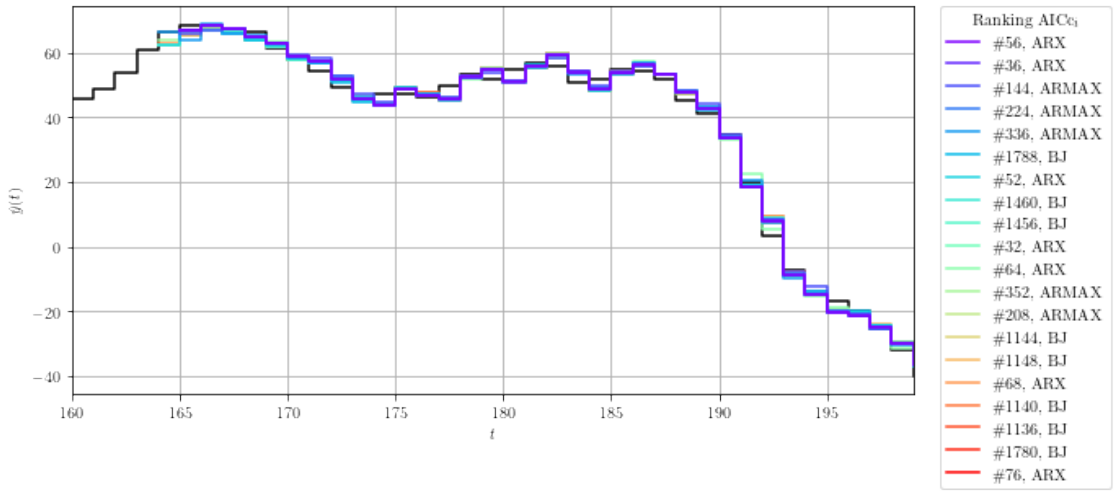


[ ]:
```
qty = 20
for criterion in ['Jv', 'AICCv', 'AICCi']:
  fig = plt.figure(figsize=(8,4))
  plt.plot(t_v, y_v, drawstyle='steps-post', color='k')
  colors = iter(plt.cm.rainbow(np.linspace(0, 1, qty)))
  for i, (index, model) in enumerate(models.sort_values(by=['Jv']).iterrows()):
    if i >= qty:
      break
    color = next(colors)
```

```
   plt.plot(t_v[int(model.delay):], model.yp, drawstyle='steps-post',␣
↪zorder=qty-i, label=f"\\#{index}, {model.model}", color=color)

plt.xlabel('$t$')
plt.ylabel('$\\hat{y}(t)$')
plt.xlim(t_v[0], t_v[-1])
plt.grid()
leg = fig.legend(title=f'Ranking {costs[criterion]}', bbox_to_anchor=(1.2, 0.
↪98))
plt.tight_layout()
plt.savefig(figPath + f'prediction_{criterion}.' + figExt, format=figExt,␣
↪bbox_extra_artists=(leg,), bbox_inches='tight')
plt.show()
```

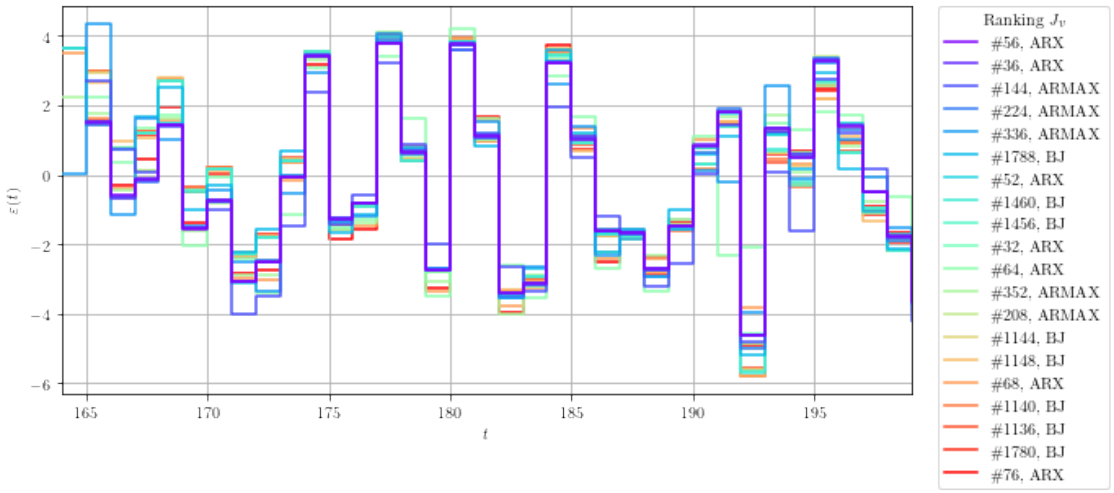


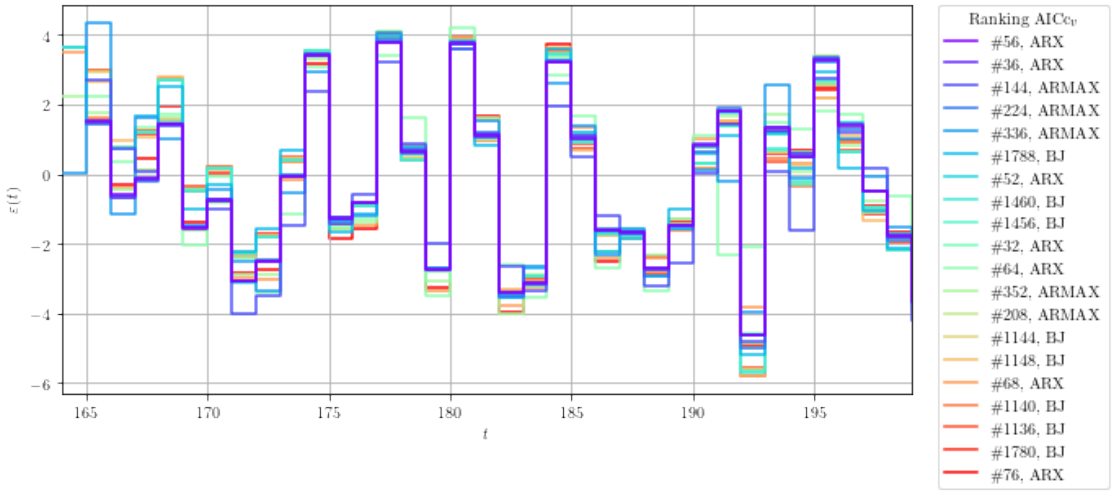


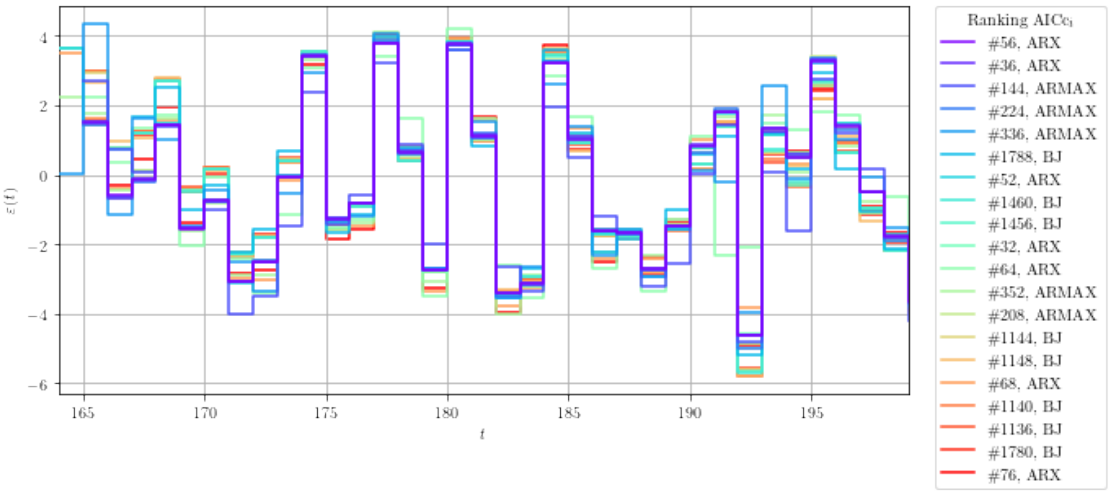


7.6 Residues

[ ]:
```
qty = 20
for criterion in ['Jv', 'AICCv', 'AICCi']:
  fig = plt.figure(figsize=(8,4))
  colors = iter(plt.cm.rainbow(np.linspace(0, 1, qty)))
  min_t = 1E1000
  for i, (index, model) in enumerate(models.sort_values(by=['Jv']).iterrows()):
    if i >= qty:
      break
    color = next(colors)

    min_t = min(min_t, t_v[int(model.delay)])

    plt.plot(t_v[int(model.delay):], model.ev, drawstyle='steps-post',␣
 ↪zorder=qty-i, label=f"\\#{index}, {model.model}", color=color)

  plt.xlabel('$t$')
  plt.ylabel('$\\varepsilon(t)$')
  plt.xlim(min_t, t_v[-1])
  plt.grid()
  leg = fig.legend(title=f'Ranking {costs[criterion]}', bbox_to_anchor=(1.2, 0.
 ↪98))
  plt.tight_layout()
  plt.savefig(figPath + f'residue_{criterion}.' + figExt, format=figExt,␣
 ↪bbox_extra_artists=(leg,), bbox_inches='tight')
  plt.show()
```

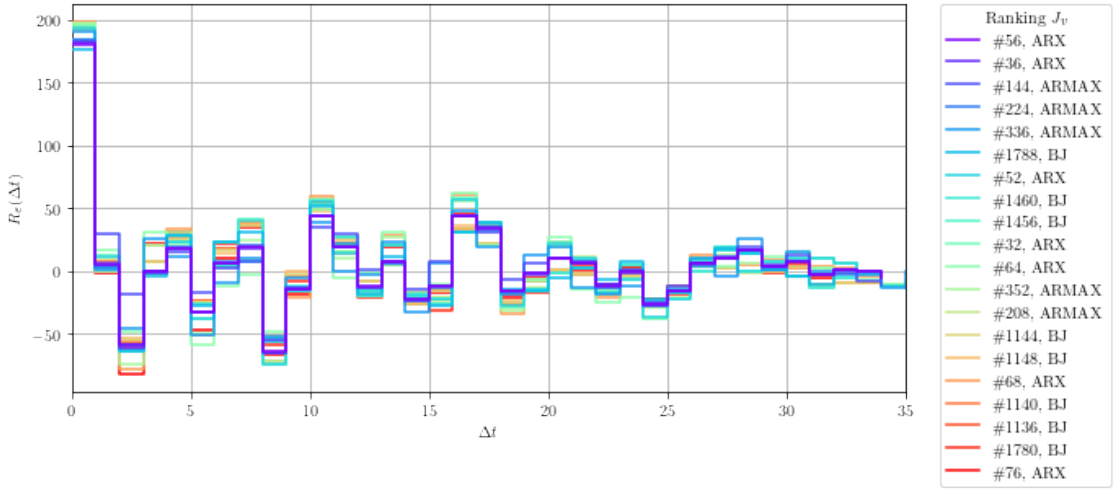

Ranking J_v
#56, ARX
#36, ARX
#144, ARMAX
#224, ARMAX
#336, ARMAX
#1788, BJ
#52, ARX
#1460, BJ
#1456, BJ
#32, ARX
#64, ARX
#352, ARMAX
#208, ARMAX
#1144, BJ
#1148, BJ
#68, ARX
#1140, BJ
#1136, BJ
#1780, BJ
#76, ARX
$\varepsilon(t)$
t

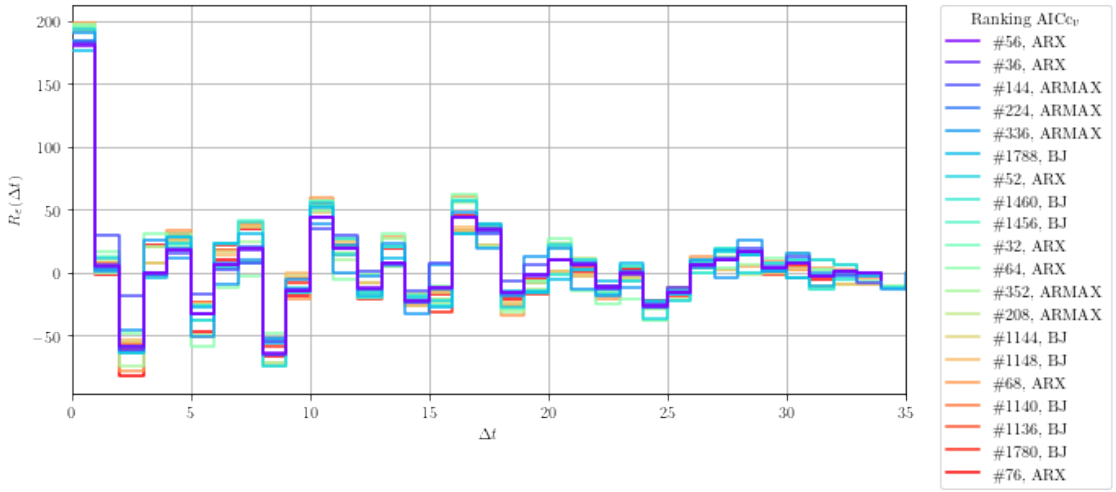

Ranking $AICc_v$
#56, ARX
#36, ARX
#144, ARMAX
#224, ARMAX
#336, ARMAX
#1788, BJ
#52, ARX
#1460, BJ
#1456, BJ
#32, ARX
#64, ARX
#352, ARMAX
#208, ARMAX
#1144, BJ
#1148, BJ
#68, ARX
#1140, BJ
#1136, BJ
#1780, BJ
#76, ARX
$\varepsilon(t)$
t

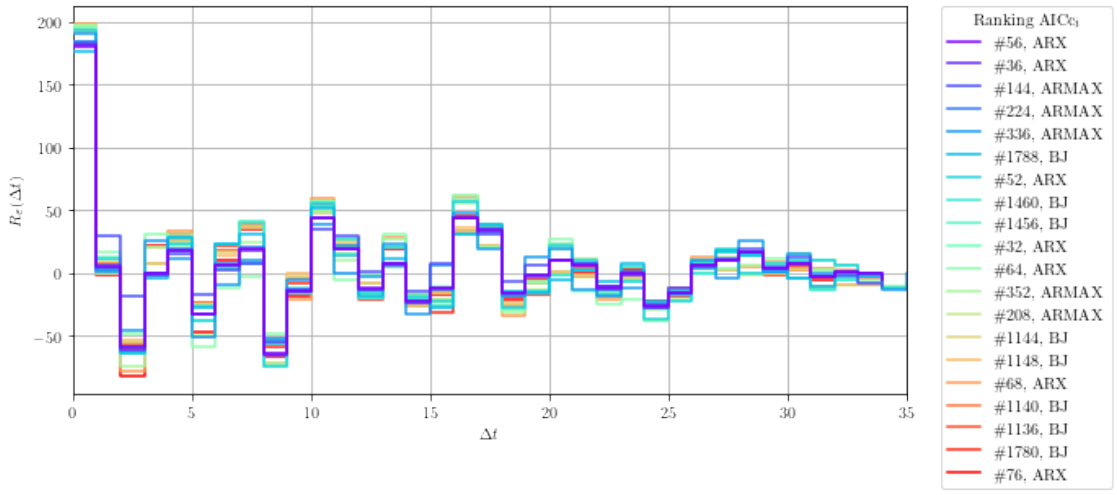


7.7 Residues Autocorrelation

[ ]:
```
qty = 20
for criterion in ['Jv', 'AICCv', 'AICCi']:
  fig = plt.figure(figsize=(8,4))
  colors = iter(plt.cm.rainbow(np.linspace(0, 1, qty)))
  max_lag = 0
  for i, (index, model) in enumerate(models.sort_values(by=['Jv']).iterrows()):
    if i >= qty:
      break
    color = next(colors)

    autocorr = (np.correlate(model.ev, model.ev, 'full')[len(model.ev)-1:])
    max_lag = max(max_lag, len(autocorr)-1)

    plt.plot(autocorr, drawstyle='steps-post', zorder=qty-i,␣
 ↪label=f"\\#{index}, {model.model}", color=color)

  plt.xlabel('$\\Delta t$')
  plt.ylabel('$R_{\\varepsilon}(\\Delta t)$')
  plt.xlim(0, max_lag)
  plt.grid()
  leg = fig.legend(title=f'Ranking {costs[criterion]}', bbox_to_anchor=(1.2, 0.
 ↪98))
  plt.tight_layout()
  plt.savefig(figPath + f'residue_autocorrelation_{criterion}.' + figExt,␣
 ↪format=figExt, bbox_extra_artists=(leg,), bbox_inches='tight')
  plt.show()
```


200
150
100
50
0
−50
0
5
10
15
20
25
30
35
Δt
$R_\varepsilon(\Delta t)$
Ranking J_v
#56, ARX
#36, ARX
#144, ARMAX
#224, ARMAX
#336, ARMAX
#1788, BJ
#52, ARX
#1460, BJ
#1456, BJ
#32, ARX
#64, ARX
#352, ARMAX
#208, ARMAX
#1144, BJ
#1148, BJ
#68, ARX
#1140, BJ
#1136, BJ
#1780, BJ
#76, ARX


200
150
100
50
0
−50
0
5
10
15
20
25
30
35
Δt
$R_\varepsilon(\Delta t)$
Ranking $AICc_v$
#56, ARX
#36, ARX
#144, ARMAX
#224, ARMAX
#336, ARMAX
#1788, BJ
#52, ARX
#1460, BJ
#1456, BJ
#32, ARX
#64, ARX
#352, ARMAX
#208, ARMAX
#1144, BJ
#1148, BJ
#68, ARX
#1140, BJ
#1136, BJ
#1780, BJ
#76, ARX



8 Model in Class

[ ]:
```
from control import TransferFunction

model = models.loc[(models.model == 'ARX') & (models.na == 2) & (models.nb ==␣
 ↪2) & (models.nk == 1)]
assert(len(model) == 1)
idx = model.index[0]
model = model.iloc[0]

G0 = TransferFunction.minreal(TransferFunction([2, 2, -1.5], [1, -1.4, 0.48,␣
 ↪0], dt=True))
H0 = TransferFunction.minreal(TransferFunction([1, 0, 0, 0], [1, -1.4, 0.48,␣
 ↪0], dt=True))

print('A = ')
display(model.A)
print('B = ')
display(model.B)

print('G_0 =')
display(G0)
print('G =')
display(model.G)

print('H_0 =')
display(H0)
print('H =')
display(model.H)
```

```
print(f'J_v    = {model.Jv:7.3f}')
print(f'J_i    = {model.Ji:7.3f}')
print(f'AIC_v  = {model.AICv:7.3f}')
print(f'AICC_v = {model.AICCv:7.3f}')
print(f'AIC_i  = {model.AICi:7.3f}')
print(f'AICC_i = {model.AICCi:7.3f}')
```

```
A =
array([ 1.        , -1.40683412,  0.48261202])
B =
array([ 0.        ,  2.16246556,  1.61084338, -1.6016382 ])
G_0 =
```

$$\frac{2z^2 + 2z - 1.5}{z^3 - 1.4z^2 + 0.48z}$$

```
G =
```

$$\frac{2.162z^2 + 1.611z - 1.602}{z^3 - 1.407z^2 + 0.4826z}$$

```
H_0 =
```

$$\frac{z^2}{z^2 - 1.4z + 0.48}$$

```
H =
```

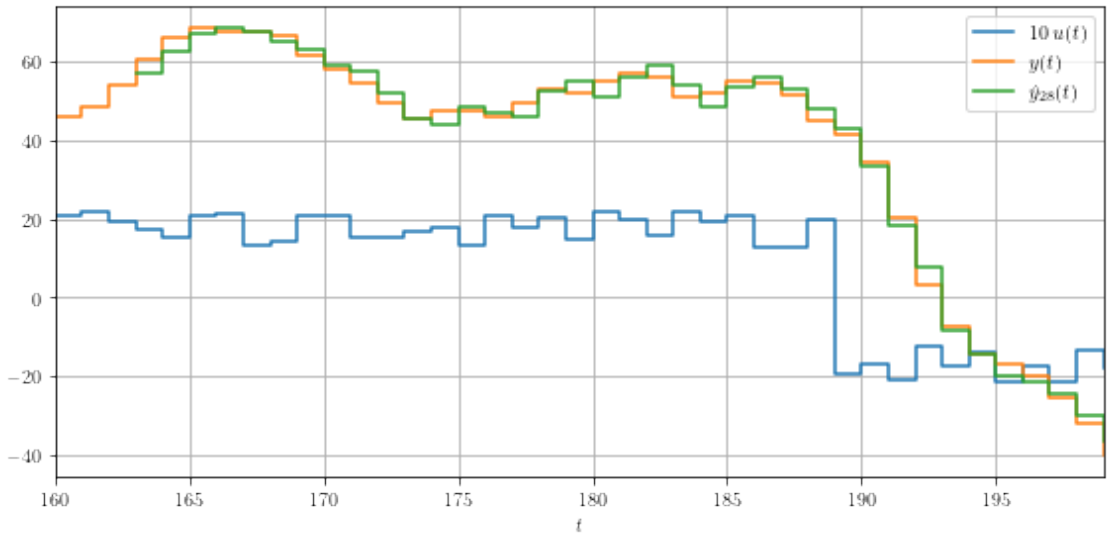
$$\frac{z^2}{z^2 - 1.407z + 0.4826}$$

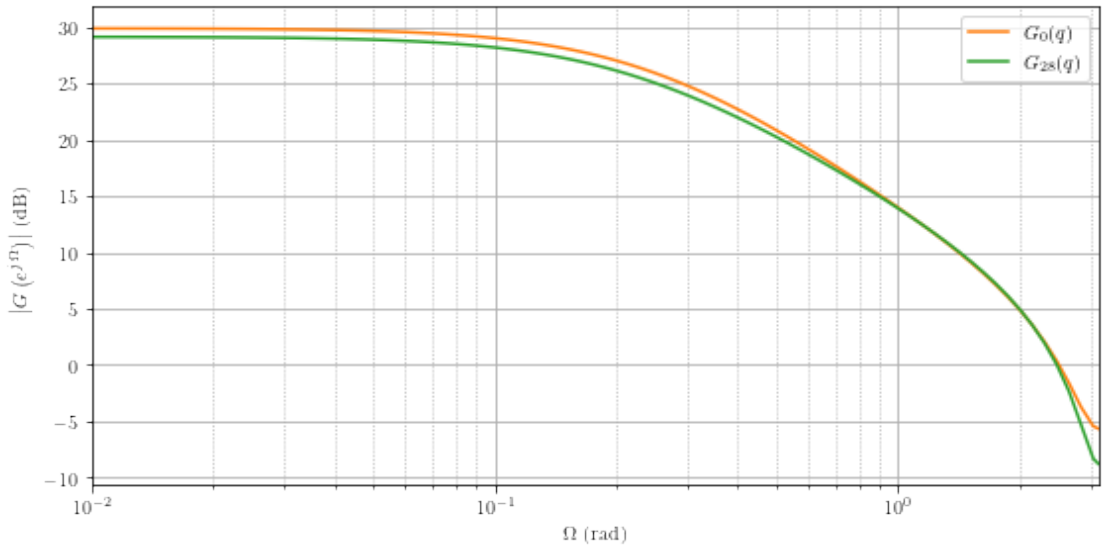
```
J_v    =   5.589
J_i    =   5.276
AIC_v  =  78.832
AICC_v =  80.596
AIC_i  = 276.114
AICC_i = 276.503
```

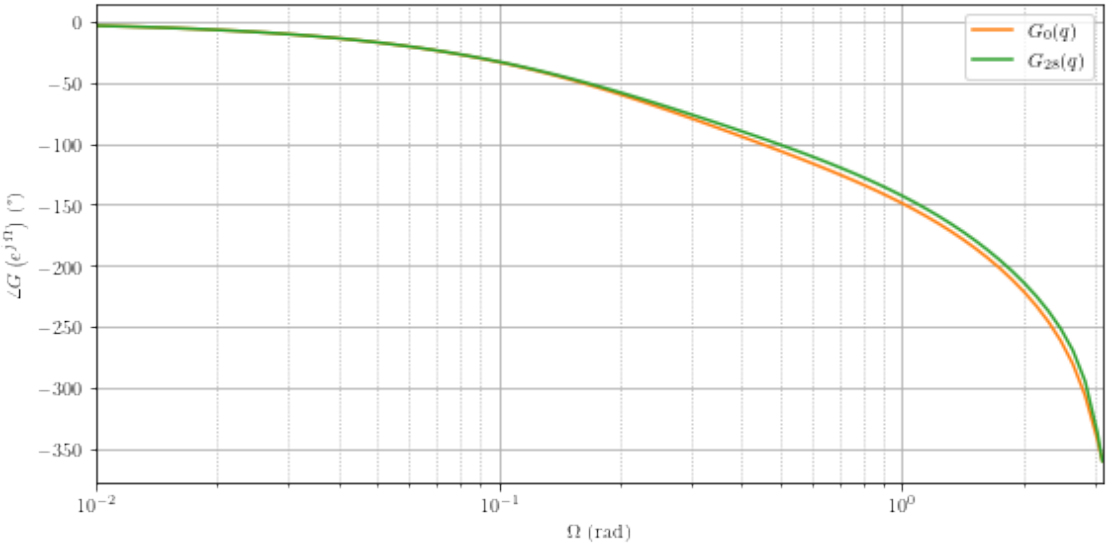
8.1 Compare to Original System

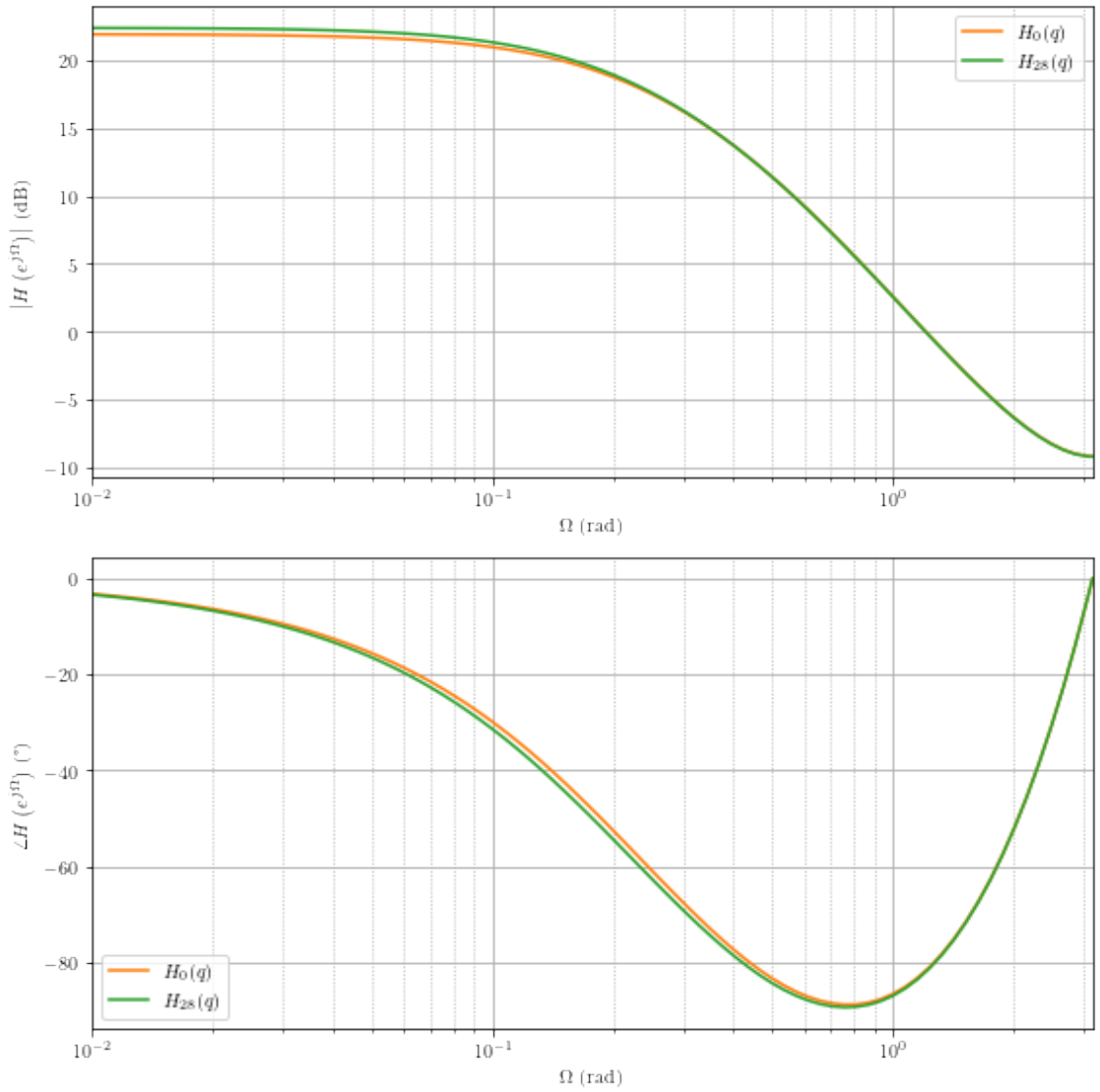
[ ]:
```
from control import frequency_response, mag2db

plt.figure(figsize=(8,4))
plt.plot(t_v, 10*u_v, label='$10\,u(t)$', drawstyle='steps-post')
plt.plot(t_v, y_v, label='$y(t)$', drawstyle='steps-post', color='tab:orange')
```

```
plt.plot(t_v[int(model.delay):], model.yp, label=f'$\\hat{{y}}_{{{idx}}}(t)$',␣
 ↪drawstyle='steps-post', color='tab:green')
plt.xlim(t_v[0], t_v[-1])
plt.xlabel('$t$')
plt.grid()
plt.legend()
plt.tight_layout()
plt.savefig(figPath + f'y_p_{idx}.' + figExt, format=figExt)
plt.show()

logspace = np.logspace(-2, 1, 100)
logspace = np.append(logspace[logspace < np.pi], np.pi)

tf0 = {
  'G': G0,
  'H': H0,
}

for tf in ['G', 'H']:
  mag,  phase,  omega  = frequency_response(model[tf], omega=logspace)
  mag0, phase0, omega0 = frequency_response(tf0[tf],   omega=logspace)

  fig, axs = plt.subplots(2, 1, figsize=(8,8))
  axs[0].plot(omega0, mag2db(mag0), label = f'${tf}_0(q)$', color='tab:orange')
  axs[1].plot(omega0, 180/np.pi*np.unwrap(phase0), label = f'${tf}_0(q)$',␣
 ↪color='tab:orange')
  axs[0].plot(omega,  mag2db(mag), label = f'${tf}_{{{idx}}}(q)$', color='tab:
 ↪green')
  axs[1].plot(omega,  180/np.pi*np.unwrap(phase), label =␣
 ↪f'${tf}_{{{idx}}}(q)$', color='tab:green')
  for ax in axs:
    ax.set_xscale('log')
    ax.set_xlim(omega[0], omega[-1])
    ax.grid(which='major')
    ax.grid(which='minor', linestyle=':')
    ax.legend()
    ax.set_xlabel('$\\Omega$ (rad)')
    axs[0].set_ylabel(f'$\\left| {tf}\\left(e^{{j \\, \\Omega}}\\right)␣
 ↪\\right|$ (dB)')
    axs[1].set_ylabel(f'$\\angle{{{tf}\\left(e^{{j \\, \\Omega}}\\right)}}$␣
 ↪(°)')
  plt.tight_layout()
  plt.savefig(figPath + f'bode_{tf}_{idx}.' + figExt, format=figExt)
  plt.show()
```


$10\,u(t)$
$y(t)$
$\hat{y}_{2s}(t)$
60
40
20
0
−20
−40
160
165
170
175
180
185
190
195
t


$G_0(q)$
$G_{2s}(q)$
$|G(e^{j\Omega})|$ (dB)
Ω (rad)


$G_0(q)$
$G_{2s}(q)$
$\angle G(e^{j\Omega})$ (°)
Ω (rad)



9 Extra code for LaTeX

[ ]:
```
columns=[
  'model',
  'na', 'nb', 'nc', 'nd', 'nf', 'nk',
  'AICCv', 'AICCi',
  'Jv', 'Ji',
]

qty = 50
```

```
display_models(models.sort_values(by=['AICCv']), precision=3, qty=qty,␣
 ↪columns=columns)
# display_models(models.loc[(models.model == 'BJ')].sort_values(by=['AICCv']),␣
 ↪precision=3, qty=qty, columns=columns)
```

	model	na	nb	nc	nd	nf	nk	AICCv	AICCi	Jv	Ji
24	ARX	2	1	-	-	-	1	79.463	288.142	5.801	5.75
44	ARX	3	1	-	-	-	1	80.358	289.971	5.556	5.74
28	ARX	2	2	-	-	-	1	80.596	276.503	5.589	5.276
20	ARX	2	0	-	-	-	1	80.82	290.423	6.384	5.91
40	ARX	3	0	-	-	-	1	81.303	291.572	6.074	5.875
32	ARX	2	3	-	-	-	1	82.077	278.289	5.41	5.264
64	ARX	4	1	-	-	-	1	82.145	288.495	5.419	5.611
160	ARMAX	2	0	1	-	-	1	82.406	292.387	6.244	5.905
240	ARMAX	3	0	1	-	-	1	82.44	291.698	5.853	5.802
1136	BJ	-	2	0	2	1	1	82.909	276.778	5.524	5.214
816	BJ	-	1	0	2	1	1	83.04	273.646	5.941	5.183
36	ARX	2	4	-	-	-	1	83.283	281.484	5.179	5.297
192	ARMAX	2	2	1	-	-	1	83.302	278.685	5.579	5.277
60	ARX	4	0	-	-	-	1	83.403	291.217	5.995	5.784
144	ARMAX	1	4	1	-	-	1	83.435	299.33	5.199	5.922
112	ARMAX	1	2	1	-	-	1	83.574	294.45	6.021	5.902
48	ARX	3	2	-	-	-	1	83.698	278.625	5.634	5.275
336	ARMAX	4	1	1	-	-	1	84.031	282.669	5.277	5.337
832	BJ	-	1	0	3	1	1	84.041	276.12	5.683	5.193
52	ARX	3	3	-	-	-	1	84.774	280.442	5.376	5.263
320	ARMAX	4	0	1	-	-	1	84.879	291.151	5.803	5.705
1456	BJ	-	3	0	2	1	1	84.939	277.505	5.398	5.167
256	ARMAX	3	1	1	-	-	1	84.955	284.692	5.814	5.479
176	ARMAX	2	1	1	-	-	1	84.983	287.503	6.237	5.652
208	ARMAX	2	3	1	-	-	1	85.515	280.548	5.476	5.266
68	ARX	4	2	-	-	-	1	85.708	277.49	5.503	5.167
1140	BJ	-	2	0	2	2	1	85.844	278.965	5.521	5.214
56	ARX	3	4	-	-	-	1	86.072	283.653	5.133	5.296
848	BJ	-	1	0	4	1	1	86.957	274.874	5.677	5.083
224	ARMAX	2	4	1	-	-	1	86.977	283.768	5.25	5.299
128	ARMAX	1	3	1	-	-	1	87.074	296.459	6.13	5.897
820	BJ	-	1	0	2	2	1	87.227	275.785	6.154	5.182
1152	BJ	-	2	0	3	1	1	87.266	276.65	5.721	5.14
500	BJ	-	0	0	2	2	1	87.934	293.775	6.714	5.878
21	ARX	2	0	-	-	-	2	88.031	316.766	7.645	6.968
272	ARMAX	3	2	1	-	-	1	88.036	280.369	5.832	5.26
1460	BJ	-	3	0	2	2	1	88.07	279.748	5.396	5.168
504	BJ	-	0	0	2	3	1	88.345	286.727	6.328	5.549
836	BJ	-	1	0	3	2	1	88.464	278.264	5.895	5.192
352	ARMAX	4	2	1	-	-	1	88.64	279.09	5.473	5.147
1144	BJ	-	2	0	2	3	1	88.741	281.16	5.487	5.214
800	BJ	-	1	0	1	1	1	89.184	326.372	7.397	7.302

1776	BJ	-	4	0	2	1	1	89.226	280.782	5.554	5.201
252	ARMAX	3	0	4	-	-	1	89.323	288.898	5.568	5.472
72	ARX	4	3	-	-	-	1	89.631	278.693	5.611	5.134
132	ARMAX	1	3	2	-	-	1	89.667	276.054	6.075	5.12
41	ARX	3	0	-	-	-	2	89.688	318.363	7.49	6.946
488	BJ	-	0	0	1	3	1	89.763	331.5	7.028	7.441
824	BJ	-	1	0	2	3	1	89.767	277.946	6.09	5.181
508	BJ	-	0	0	2	4	1	90.071	287.249	6.137	5.491