cov-matrix-processing

July 1, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.offsetbox import AnchoredText
%matplotlib inline
```

[2]:
```
n = 125
cov_nercome = np.loadtxt(f"../output/
 ↪Patchy_V6C_BOSS_DR12_NGC_z1_cov_nercome_{n}.matrix")
cov_sample = np.loadtxt(f"../output/Patchy_V6C_BOSS_DR12_NGC_z1_cov_sample_{n}.
 ↪matrix")

# 'Real' covariance matrix before processing.
# The term 'real' here indicates that this is the matrix that we use to compare
# our estimates to.
cov_real_preproc = np.loadtxt("../data/
 ↪C_2048_BOSS_DR12_NGC_z1_V6C_1_1_1_1_1_10_200_200_prerecon.matrix")
```

[3]:
```
print(cov_nercome)
print(cov_sample)
```

```
[[ 2.42278140e+08  2.20522500e+07  5.40836349e+06 … -5.85612570e+04
  -1.31632017e+05 -5.26394269e+04]
 [ 2.20522500e+07  5.58438961e+07  4.76770885e+06 …  4.78730143e+04
   3.72423079e+04 -4.66811795e+03]
 [ 5.40836349e+06  4.76770885e+06  2.33652105e+07 …  3.75540702e+04
   2.25893312e+04  3.41081543e+04]
 …
 [-5.85612570e+04  4.78730143e+04  3.75540702e+04 …  1.78046420e+04
   3.54869663e+03  1.01461533e+03]
 [-1.31632017e+05  3.72423079e+04  2.25893312e+04 …  3.54869663e+03
   1.63399268e+04  1.84398032e+03]
 [-5.26394269e+04 -4.66811795e+03  3.41081543e+04 …  1.01461533e+03
   1.84398032e+03  1.71032703e+04]]
[[ 2.35627951e+08  2.28750394e+07  6.33742351e+06 … -5.80164085e+04
  -1.29910277e+05 -5.25361357e+04]
 [ 2.28750394e+07  4.73703604e+07  4.55518478e+06 …  3.86984558e+04
   2.97950701e+04 -1.38467341e+04]
 [ 6.33742351e+06  4.55518478e+06  1.86805365e+07 …  2.21667946e+04
```

```
  1.51871282e+04  2.67959571e+04]
 …
 [-5.80164085e+04  3.86984558e+04  2.21667946e+04 …  4.49990943e+03
  1.55269947e+03  3.56928303e+02]
 [-1.29910277e+05  2.97950701e+04  1.51871282e+04 …  1.55269947e+03
  3.40139860e+03  7.27016169e+02]
 [-5.25361357e+04 -1.38467341e+04  2.67959571e+04 …  3.56928303e+02
  7.27016169e+02  3.95752797e+03]]
```

[4]:
```
indices = np.concatenate((np.arange(40), np.arange(40)+80, np.arange(40)+160))
cov_real = (cov_real_preproc[indices, :])[:, indices]
```

[5]:
```
print(cov_real)
```

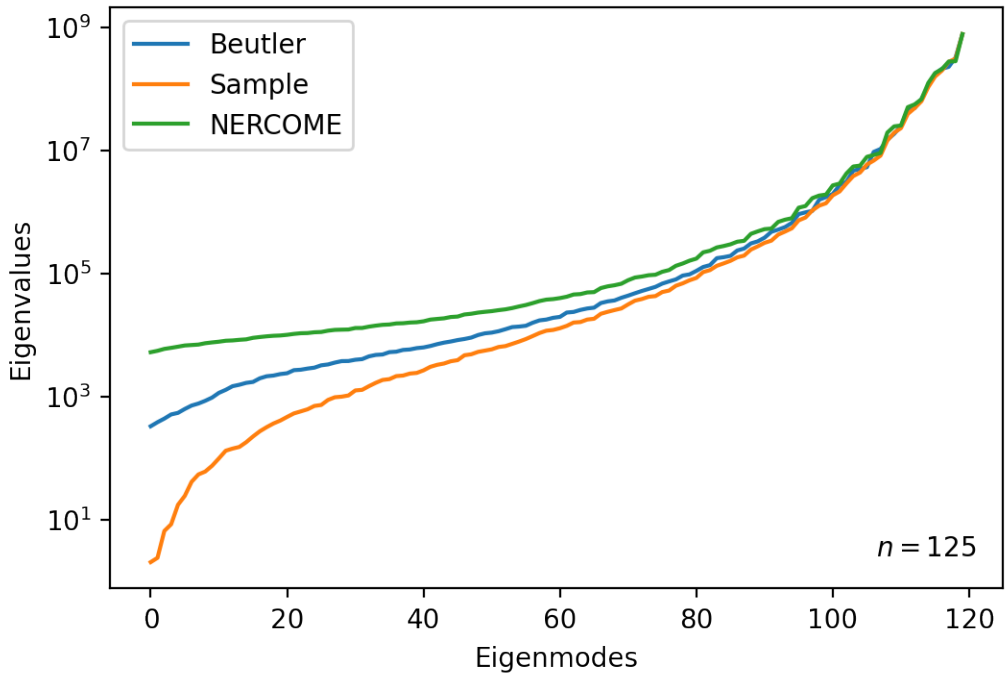
```
[[ 2.39531896e+08  1.91051563e+07  1.44038639e+05 …  9.68083967e+03
  -1.21684291e+04  2.21567909e+04]
 [ 1.91051563e+07  5.15256710e+07  4.78376836e+06 …  9.13608077e+03
  9.26311893e+03 -1.41639754e+04]
 [ 1.44038639e+05  4.78376836e+06  1.67772970e+07 … -5.49127765e+02
  -2.28200815e+03  6.76773412e+03]
 …
 [ 9.68083967e+03  9.13608077e+03 -5.49127765e+02 …  4.22979902e+03
  1.28931823e+03  3.78440184e+02]
 [-1.21684291e+04  9.26311893e+03 -2.28200815e+03 …  1.28931823e+03
  3.88691287e+03  1.00320951e+03]
 [ 2.21567909e+04 -1.41639754e+04  6.76773412e+03 …  3.78440184e+02
  1.00320951e+03  3.62984716e+03]]
```

[6]:
```
evals_nercome = np.linalg.eigvalsh(cov_nercome)
evals_sample = np.linalg.eigvalsh(cov_sample)
evals_real = np.linalg.eigvalsh(cov_real)
```

[7]:
```
plt.figure(dpi=200)
plt.plot(evals_real, label="Beutler")
plt.plot(evals_sample, label="Sample")
plt.plot(evals_nercome, label="NERCOME")
plt.yscale("log")
plt.legend()
plt.xlabel("Eigenmodes")
plt.ylabel("Eigenvalues")

parameters = AnchoredText(fr"$n = {n}$", frameon=False, loc="lower right",␣
 ↪pad=0.5)
plt.setp(parameters.patch, facecolor='white', alpha=0.5)
plt.gca().add_artist(parameters)
```

[7]:
```
<matplotlib.offsetbox.AnchoredText at 0x7fa9ca788910>
```



[8]:
```
nercome_rel_errors = (cov_nercome-cov_real)/np.abs(cov_real)
print(nercome_rel_errors)
```

```
[[ 1.14650440e-02  1.54256457e-01  3.65480048e+01 … -7.04919191e+00
  -9.81750288e+00 -3.37576945e+00]
 [ 1.54256457e-01  8.38072557e-02 -3.35708390e-03 …  4.23999465e+00
   3.02049334e+00  6.70423181e-01]
 [ 3.65480048e+01 -3.35708390e-03  3.92668350e-01 …  6.93885838e+01
   1.08988828e+01  4.03981889e+00]
 …
 [-7.04919191e+00  4.23999465e+00  6.93885838e+01 …  3.20933523e+00
   1.75238226e+00  1.68104545e+00]
 [-9.81750288e+00  3.02049334e+00  1.08988828e+01 …  1.75238226e+00
   3.20383151e+00  8.38080974e-01]
 [-3.37576945e+00  6.70423181e-01  4.03981889e+00 …  1.68104545e+00
   8.38080974e-01  3.71184309e+00]]
```

[9]:
```
nercome_pos_count = 0
nercome_neg_count = 0
for row in nercome_rel_errors:
    for error in row:
        if error >= 0:
            nercome_pos_count += 1
        elif error < 0:
            nercome_neg_count += 1
```

```
print(f"NERCOME overestimated {nercome_pos_count} elements")
print(f"NERCOME underestimated {nercome_neg_count} elements")

index_max_nercome = np.unravel_index(np.abs(nercome_rel_errors).argmax(),␣
 ↪nercome_rel_errors.shape)

print(f"Maximum relative error is {nercome_rel_errors[index_max_nercome]}")
print(f"Maximum relative error index is {index_max_nercome}")
print(f"NERCOME: {cov_nercome[index_max_nercome]}, real:␣
 ↪{cov_real[index_max_nercome]}")
```

```
NERCOME overestimated 8251 elements
NERCOME underestimated 6149 elements
Maximum relative error is -43295.78785237451
Maximum relative error index is (80, 94)
NERCOME: -576517.5911522222, real: -13.315481811674502
```

[10]:
```
sample_rel_errors = (cov_sample-cov_real)/np.abs(cov_real)
print(sample_rel_errors)
```

```
[[-1.62982280e-02  1.97322812e-01  4.29980798e+01 … -6.99291079e+00
  -9.67601045e+00 -3.37110762e+00]
 [ 1.97322812e-01 -8.06454441e-02 -4.77831615e-02 …  3.23578302e+00
   2.21652678e+00  2.23977594e-02]
 [ 4.29980798e+01 -4.77831615e-02  1.13441365e-01 …  4.13672806e+01
   7.65515948e+00  2.95936906e+00]
 …
 [-6.99291079e+00  3.23578302e+00  4.13672806e+01 …  6.38589230e-02
   2.04279465e-01 -5.68435444e-02]
 [-9.67601045e+00  2.21652678e+00  7.65515948e+00 …  2.04279465e-01
  -1.24909995e-01 -2.75309735e-01]
 [-3.37110762e+00  2.23977594e-02  2.95936906e+00 … -5.68435444e-02
  -2.75309735e-01  9.02739963e-02]]
```

[11]:
```
sample_pos_count = 0
sample_neg_count = 0
for row in sample_rel_errors:
    for error in row:
        if error >= 0:
            sample_pos_count += 1
        elif error < 0:
            sample_neg_count += 1

print(f"Sample overestimated {sample_pos_count} elements")
print(f"Sample underestimated {sample_neg_count} elements")
```

```
index_max_sample = np.unravel_index(np.abs(sample_rel_errors).argmax(),␣
 ↪sample_rel_errors.shape)

print(f"Maximum relative error is {sample_rel_errors[index_max_sample]}")
print(f"Maximum relative error index is {index_max_sample}")
print(f"Sample: {cov_sample[index_max_sample]}, real:␣
 ↪{cov_real[index_max_sample]}")
```

```
Sample overestimated 7614 elements
Sample underestimated 6786 elements
Maximum relative error is -41554.26075549169
Maximum relative error index is (94, 80)
Sample: -553328.3187691408, real: -13.315481811674502
```

[12]:
```
MSE_NERCOME = np.trace((cov_nercome-cov_real)@(cov_nercome-cov_real).T)
MSE_sample = np.trace((cov_sample-cov_real)@(cov_sample-cov_real).T)
print(f"MSE NERCOME: {MSE_NERCOME}")
print(f"MSE sample: {MSE_sample}")
```

```
MSE NERCOME: 1.5059661732944432e+16
MSE sample: 1.4913453744881512e+16
```

[ ]:
```

```