cov-matrix-processing

July 19, 2022

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

[2]:
```
n = 500
cov_nercome = np.loadtxt(f"../output/data/
 ↪Patchy_V6C_BOSS_DR12_NGC_z1_cov_nercome_{n}_avg10000.matrix")
cov_sample = np.loadtxt(f"../output/data/
 ↪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)
```

```
[[ 2.46604449e+08  1.92959769e+07 -1.66300069e+05 …  9.46477087e+03
   5.28779873e+03 -1.02158640e+04]
 [ 1.92959769e+07  5.70610403e+07  4.69247881e+06 …  8.49211770e+03
   4.00700285e+04 -8.47914566e+03]
 [-1.66300069e+05  4.69247881e+06  1.75513111e+07 …  1.53365239e+03
   1.11978330e+04  1.65337234e+04]
 …
 [ 9.46477087e+03  8.49211770e+03  1.53365239e+03 …  6.33454458e+03
   1.61119987e+03  1.16636765e+02]
 [ 5.28779873e+03  4.00700285e+04  1.11978330e+04 …  1.61119987e+03
   5.72624845e+03  1.24653029e+03]
 [-1.02158640e+04 -8.47914566e+03  1.65337234e+04 …  1.16636765e+02
   1.24653029e+03  5.36600900e+03]]
[[ 2.03256210e+08  2.26312165e+07 -8.10168108e+05 … -2.56605953e+04
   3.12592410e+04  6.58264930e+04]
 [ 2.26312165e+07  4.70756413e+07  4.54771244e+06 …  3.46957861e+02
   7.10159306e+03 -2.86043497e+04]
 [-8.10168108e+05  4.54771244e+06  1.61021470e+07 …  1.47426460e+04
```

```
  1.28943477e+04  1.12660773e+04]
 …
 [-2.56605953e+04  3.46957861e+02  1.47426460e+04 …  3.93458356e+03
   9.11664424e+02  3.91148844e+02]
 [ 3.12592410e+04  7.10159306e+03  1.28943477e+04 …  9.11664424e+02
   3.65193234e+03  9.30709554e+02]
 [ 6.58264930e+04 -2.86043497e+04  1.12660773e+04 …  3.91148844e+02
   9.30709554e+02  3.51887142e+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]:
```
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}")
```

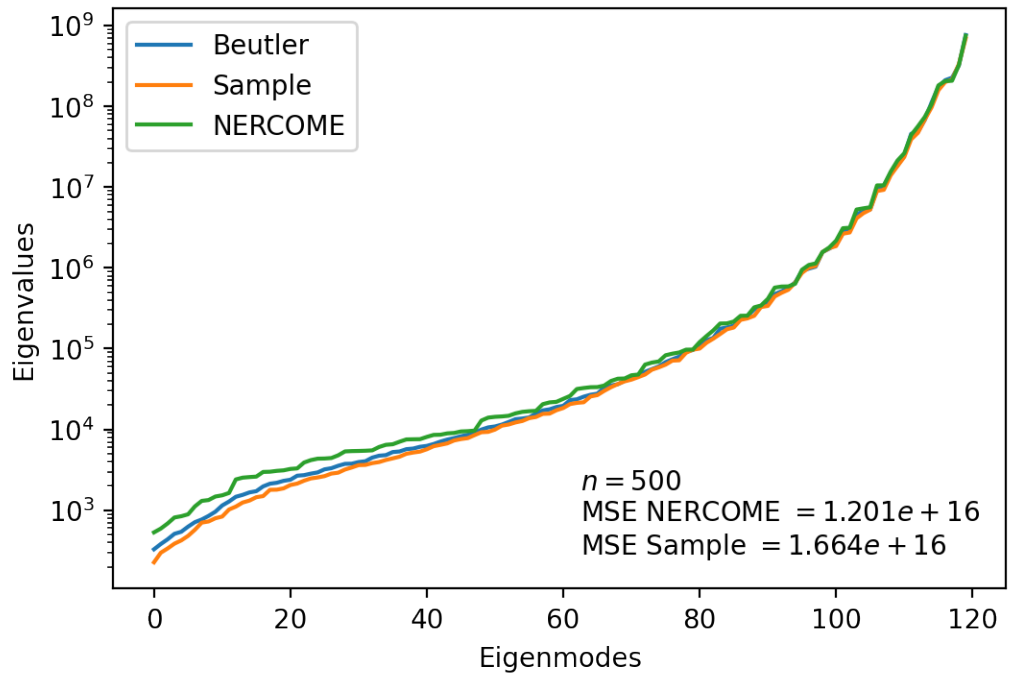
```
MSE NERCOME: 1.2010178469695016e+16
MSE sample: 1.664069018003966e+16
```

[7]:
```
evals_nercome = np.linalg.eigvalsh(cov_nercome)
evals_sample = np.linalg.eigvalsh(cov_sample)
evals_real = np.linalg.eigvalsh(cov_real)
```

[8]:
```
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"$n = {n}$" "\n"
    fr"MSE NERCOME $= {np.format_float_scientific(MSE_NERCOME, precision=3)}$"␣
 ↪"\n"
    fr"MSE Sample $={np.format_float_scientific(MSE_sample, precision=3)}$",
    frameon=False, loc="lower right", pad=0.5)
plt.setp(parameters.patch, facecolor='white', alpha=0.5)
plt.gca().add_artist(parameters)
```

[8]: <matplotlib.offsetbox.AnchoredText at 0x7ff0f708e640>



[9]:
```
nercome_rel_errors = (cov_nercome-cov_real)/np.abs(cov_real)
print(nercome_rel_errors)
```

```
[[ 0.02952656  0.00998791 -2.1545518  … -0.02231922  1.43455065
  -1.46107146]
 [ 0.00998791  0.10742935 -0.01908319 … -0.0704857   3.32575991
   0.40135835]
 [-2.1545518  -0.01908319  0.04613461 …  3.79288808  5.90700835
   1.44302201]
 …
 [-0.02231922 -0.0704857   3.79288808 …  0.49759942  0.2496526
  -0.69179603]
 [ 1.43455065  3.32575991  5.90700835 …  0.2496526   0.47321246
```

```
  0.24254234]
 [-1.46107146  0.40135835  1.44302201 … -0.69179603  0.24254234
  0.47830164]]
```

[10]:
```
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 7436 elements
NERCOME underestimated 6964 elements
Maximum relative error is 32973.08051800205
Maximum relative error index is (80, 94)
NERCOME: 439039.1384305235, real: -13.315481811674502
```

[11]:
```
sample_rel_errors = (cov_sample-cov_real)/np.abs(cov_real)
print(sample_rel_errors)
```

```
[[-0.15144407  0.18456066 -6.62465819 … -3.65065802  3.56888056
   1.97093985]
 [ 0.18456066 -0.08636529 -0.04934518 … -0.96202334 -0.23334752
  -1.01951422]
 [-6.62465819 -0.04934518 -0.04024188 … 27.84738773  6.65043895
   0.66467492]
 …
 [-3.65065802 -0.96202334 27.84738773 … -0.06979421 -0.29290969
   0.03358169]
 [ 3.56888056 -0.23334752  6.65043895 … -0.29290969 -0.06045428
  -0.07226802]
 [ 1.97093985 -1.01951422  0.66467492 …  0.03358169 -0.07226802
  -0.03057312]]
```

[12]:
```
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 7434 elements
Sample underestimated 6966 elements
Maximum relative error is 24707.78037823806
Maximum relative error index is (80, 94)
Sample: 328982.6847514654, real: -13.315481811674502
```

[ ]:
```

```