# gn_attemp

August 25, 2022

[1]:
```
import numpy as np
import scipy
```

This is a numerical attempt to solve for the system of equations derived in the constrain_struct.pdf

## 0.1 Using Newton-Gauss

2 dimensional case:

[27]:
```
# Number of observations
n = 100
# Dimension
d = 2
```

[28]:
```
## lambda
lam = 1
```

Gamma is the sample covariance eigenvalue matrix $\Lambda$, and $D$ is the eigenvalue matrix for M

[29]:
```
Gamma = np.array([102, 103])
D = np.array([1/102, 1/103])
```

Initialize alpha value based on guessed D values

[30]:
```
alpha_initial = n*(1-lam)/d*np.prod(D**(1/d))*sum((lam*D + (1-lam)*np.
 ↪prod(D**(1/d)))**(-1) - Gamma)
## X[0] is alpha and X[i] is D[i] (i is from 1 to d)
X = np.append(alpha_initial, D)
```

[31]:
```
X[1:]**(1/d)
```

[31]:
```
array([0.09901475, 0.09853293])
```

[32]:
```
def test_2d(X):
    return np.array([
        ((1-lam)*sum((lam*X[1:] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma)␣
 ↪- X[0]*d/n*np.prod(X[1:]**(-1/d)))**2,
        (lam * ((lam*X[1] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[0]) -␣
 ↪2*X[0]/n/X[1])**2,
```

```
        (lam * ((lam*X[2] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[1]) -␣
↪2*X[0]/n/X[2])**2,
    ])
```

[33]:
```
scipy.optimize.least_squares(test_2d, X)
sol = scipy.optimize.least_squares(test_2d, X)
solD = sol['x']
```

Result:

[46]:
```
def result_summary(Gamma, solD):
    print("* * * * * | lam = " + str(lam) + " | * * * * *")
    print('alpha: ' + str(solD[0]))
    print('Gamma: ' +str(Gamma))
    print('D: ' + str(solD[1:]))
    print('inverse D: ' + str(1/solD[1:]))
    print('obj: ' +str(test_2d(solD)))
    print("s: " + str(np.prod(solD[1:])**(1/d)))
    print("* * * * * * * * * * * * * * * * * " + "\n")
```

[47]:
```
## obj is the objective function Gauss-Newton tries to minimize
result_summary(Gamma, solD)
```

```
* * * * * | lam = 1 | * * * * *
alpha: 0.0
Gamma: [99 61]
D: [0.00980392 0.00970874]
inverse D: [102. 103.]
obj: [   0.    9. 1764.]
s: 0.009756213637973549
* * * * * * * * * * * * * * * * * * * *
```

More tests:

[48]:
```
def _2d_(n, lam, Gamma, D, f):
    d = 2
    alpha_initial = n*(1-lam)/d*np.prod(D**(1/d))*sum((lam*D + (1-lam)*np.
↪prod(D**(1/d)))**(-1) - Gamma)
    X = np.append(alpha_initial, D)
    sol = f(test_2d, X)
    if f == scipy.optimize.least_squares:
        solD = sol['x']
    else:
        solD = sol
    result_summary(Gamma, solD)
```

[49]:
```
Gamma = np.random.randint(low=50, high=100, size=2)
D = 1/Gamma
for lam in [0, 0.2, 0.4, 0.6, 0.8, 1]:
    _2d_(n, lam, Gamma, D, scipy.optimize.least_squares)
```

```
* * * * * | lam = 0 | * * * * *
alpha: -1.0298009496113934e-05
Gamma: [79 67]
D: [0.01261541 0.01487488]
inverse D: [79.26816081 67.22742758]
obj: [3.56992064e-09 2.66541066e-10 1.91716528e-10]
s: 0.01369863715368149
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.2 | * * * * *
alpha: 1.1418272439497504e-05
Gamma: [79 67]
D: [0.00903319 0.0203688 ]
inverse D: [110.70286061  49.09470588]
obj: [9.88517060e-09 4.44438871e-10 1.86442930e-10]
s: 0.013564483045046547
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.4 | * * * * *
alpha: -2.1225402507914126e-05
Gamma: [79 67]
D: [0.01113227 0.0168    ]
inverse D: [89.82895444 59.52381712]
obj: [7.21554745e-09 3.91579571e-09 2.36721238e-09]
s: 0.013675601612981921
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.6 | * * * * *
alpha: -5.068814330360337e-05
Gamma: [79 67]
D: [0.01195438 0.01573289]
inverse D: [83.65136735 63.56109677]
obj: [2.89221143e-09 6.68314942e-09 1.55539062e-09]
s: 0.013714114990541496
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.8 | * * * * *
alpha: -4.482244511371996e-05
Gamma: [79 67]
D: [0.01238944 0.01522337]
inverse D: [80.71387426 65.68847253]
obj: [7.83124546e-11 4.32827214e-09 8.21902475e-10]
s: 0.013733502832414451
```

```
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 1 | * * * * *
alpha: 0.0
Gamma: [79 67]
D: [0.01265823 0.01492537]
inverse D: [79. 67.]
obj: [0. 0. 0.]
s: 0.013745136370813411
* * * * * * * * * * * * * * * * * * *
```

using fsolve:

[16]:
```
Gamma = np.random.randint(low=50, high=100, size=2)
D = 1/Gamma
for lam in [0, 0.2, 0.4, 0.6, 0.8, 1]:
    _2d_(n, lam, Gamma, D, scipy.optimize.fsolve)
```

```
* * * * * | lam = 0 | * * * * *
alpha: 1.029443642017731e-11
Gamma: [57 66]
D: [0.01749683 0.0151109 ]
inverse D: [57.15320792 66.17738772]
obj: [1.04243595e-21 1.38467027e-22 1.85645481e-22]
s: 0.016303866268150616
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.2 | * * * * *
alpha: 1.3169710833861707e-09
Gamma: [57 66]
D: [0.0231869  0.01122517]
inverse D: [43.12780794 89.08548206]
obj: [9.62577818e-18 7.08046091e-20 6.57778309e-18]
s: 0.016303866268150616
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.4 | * * * * *
alpha: 9.26111418661642e-10
Gamma: [57 66]
D: [0.01950195 0.01352109]
inverse D: [51.27691475 73.95852283]
obj: [7.73596282e-19 5.88354040e-19 4.23572197e-20]
s: 0.016303866268150616
* * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.6 | * * * * *
alpha: 7.535609288529538e-08
Gamma: [57 66]
```

```
D: [0.01838997 0.01440273]
inverse D: [54.37747101 69.43129072]
obj: [1.04226991e-15 1.07638628e-15 4.00752747e-15]
s: 0.016303866268150616
* * * * * * * * * * * * * * * * * * * *

* * * * * | lam = 0.8 | * * * * *
alpha: 2.2318749354243708e-09
Gamma: [57 66]
D: [0.0178566  0.01486617]
inverse D: [56.00171606 67.26684367]
obj: [2.12256391e-18 1.25550850e-17 1.53238895e-17]
s: 0.016303866268150616
* * * * * * * * * * * * * * * * * * * *

* * * * * | lam = 1 | * * * * *
alpha: 0.0
Gamma: [57 66]
D: [0.01754386 0.01515152]
inverse D: [57. 66.]
obj: [0. 0. 0.]
s: 0.016303866268150616
* * * * * * * * * * * * * * * * * * * *
```

using Broyden: (Not converging)

[17]:
```
Gamma = np.random.randint(low=50, high=100, size=2)
D = 1/Gamma
for lam in [0, 0.2, 0.4, 0.6, 0.8, 1]:
    _2d_(n, lam, Gamma, D, scipy.optimize.broyden1)
```

```
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\2894031937.py:3:
RuntimeWarning: invalid value encountered in sqrt
  ((1-lam)*sum((lam*X[1:] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma) -
X[0]*d/n*np.prod(X[1:]**(-1/d)))**2,
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\2894031937.py:3:
RuntimeWarning: invalid value encountered in power
  ((1-lam)*sum((lam*X[1:] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma) -
X[0]*d/n*np.prod(X[1:]**(-1/d)))**2,
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\2894031937.py:4:
RuntimeWarning: invalid value encountered in sqrt
  (lam * ((lam*X[1] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[0]) -
2*X[0]/n/X[1])**2,
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\2894031937.py:5:
RuntimeWarning: invalid value encountered in sqrt
  (lam * ((lam*X[2] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[1]) -
2*X[0]/n/X[2])**2,
C:\Users\Ziming Huang\AppData\Local\Programs\Python\Python310\lib\site-
```

```
packages\scipy\optimize\_nonlin.py:911: RuntimeWarning: invalid value
encountered in divide
  d = v / vdot(df, v)
```

```
---------------------------------------------------------------------------
NoConvergence                             Traceback (most recent call last)
Input In [17], in <cell line: 3>()
      2 D = 1/Gamma
      3 for lam in [0, 0.2, 0.4, 0.6, 0.8, 1]:
----> 4     _2d_(n, lam, Gamma, D, scipy.optimize.broyden1)

Input In [13], in _2d_(n, lam, Gamma, D, f)
      3 alpha_initial = n*(1-lam)/d*np.prod(D**(1/d))*sum((lam*D + (1-lam)*np.
 ↪prod(D**(1/d)))**(-1) - Gamma)
      4 X = np.append(alpha_initial, D)
----> 5 sol = f(test_2d, X)
      6 if f == scipy.optimize.least_squares:
      7     solD = sol['x']

File <string>:6, in broyden1(F, xin, iter, alpha, reduction_method, max_rank,␣
 ↪verbose, maxiter, f_tol, f_rtol, x_tol, x_rtol, tol_norm, line_search,␣
 ↪callback, **kw)

File␣
 ↪~\AppData\Local\Programs\Python\Python310\lib\site-packages\scipy\optimize\_nonlin.
 ↪py:241, in nonlin_solve(F, x0, jacobian, iter, verbose, maxiter, f_tol,␣
 ↪f_rtol, x_tol, x_rtol, tol_norm, line_search, callback, full_output,␣
 ↪raise_exception)
    239 else:
    240     if raise_exception:
--> 241         raise NoConvergence(_array_like(x, x0))
    242     else:
    243         status = 2

NoConvergence: [-0.53941057  0.01538467  0.01250009]
```

**Special case where all elements of $\Lambda$ are the same**

[18]:
```
Gamma = np.array([100, 100])
D = 1/Gamma
for lam in [0.5]:
    _2d_(n, lam, Gamma, D, scipy.optimize.fsolve)
```

```
* * * * * | lam = 0.5 | * * * * *
alpha: -5.329070513233791e-15
Gamma: [100 100]
D: [0.01 0.01]
inverse D: [100. 100.]
obj: [1.26217746e-29 1.26217744e-29 1.26217744e-29]
```

```
s: 0.01
* * * * * * * * * * * * * * * * * * *
```

**Compare fsolve and Gauss-Newton**

[19]:
```
lam = 0.5
Gamma = np.random.randint(low=50, high=100, size=2)
D = 1/Gamma
print('*fsolve*'),_2d_(n, lam, Gamma, D, scipy.optimize.fsolve);
print('*gauss-newton*'),_2d_(n, lam, Gamma, D, scipy.optimize.least_squares);
```

```
*fsolve*
* * * * * | lam = 0.5 | * * * * *
alpha: 7.149385317393383e-09
Gamma: [73 76]
D: [0.01397444 0.01289297]
inverse D: [71.55921057 77.56164383]
obj: [1.43949051e-16 5.75020341e-17 3.91249930e-17]
s: 0.013425540338526368
* * * * * * * * * * * * * * * * * * *

*gauss-newton*
* * * * * | lam = 0.5 | * * * * *
alpha: -9.35878730642716e-06
Gamma: [73 76]
D: [0.01397438 0.01289305]
inverse D: [71.55951326 77.56118685]
obj: [2.29846111e-09 5.97538892e-09 1.23822840e-08]
s: 0.013425540338526368
* * * * * * * * * * * * * * * * * * *
```

fsolve seems to be better than Gauss-Newton but the difference is trivial.

### 0.1.1 Dimension = 3

[20]:
```
def test_3d(X):
    return np.array([
        ((1-lam)*sum((lam*X[1:] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma)
 ↪- X[0]*d/n*np.prod(X[1:]**(-1/d)))**2,
        (lam * ((lam*X[1] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[0]) -
 ↪2*X[0]/n/X[1])**2,
        (lam * ((lam*X[2] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[1]) -
 ↪2*X[0]/n/X[2])**2,
        (lam * ((lam*X[3] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[2]) -
 ↪2*X[0]/n/X[3])**2
    ])
```

[21]:
```
def _3d_(n, lam, Gamma, D, f):
    alpha_initial = n*(1-lam)/d*np.prod(D**(1/d))*sum((lam*D + (1-lam)*np.
 ↪prod(D**(1/d)))**(-1) - Gamma)
    X = np.append(alpha_initial, D)
    sol = f(test_3d, X)
    if f == scipy.optimize.least_squares:
        solD = sol['x']
    else:
        solD = sol
    result_summary(Gamma, solD)
```

Case: $\lambda = 1$

[22]:
```
n = 100
lam = 1
Gamma = np.random.randint(low=50, high=100, size=3)
D = 1/Gamma
_3d_(n, lam, Gamma, D, scipy.optimize.least_squares)
```

```
* * * * * | lam = 1 | * * * * *
alpha: 0.0
Gamma: [96 83 95]
D: [0.01041667 0.01204819 0.01052632]
inverse D: [96. 83. 95.]
obj: [0. 0. 0.]
s: 0.0011493797321856895
* * * * * * * * * * * * * * * * * * *
```

Case: $\lambda = 0.9$

[23]:
```
n = 100
lam = 0.9
Gamma = np.random.randint(low=50, high=100, size=3)
D = 1/Gamma
_3d_(n, lam, Gamma, D, scipy.optimize.least_squares)
```

```
* * * * * | lam = 0.9 | * * * * *
alpha: 1.1600278635673114e-05
Gamma: [60 64 66]
D: [0.01826557 0.01710816 0.01658207]
inverse D: [54.74781712 58.45162741 60.30610026]
obj: [6.27435886e-09 2.98935397e-09 3.00293163e-09]
s: 0.0019863803942616505
* * * * * * * * * * * * * * * * * * *
```

Case: $\lambda = 0.2$

[24]:
```
n = 100
lam = 0.2
Gamma = np.random.randint(low=50, high=100, size=3)
D = 1/Gamma
_3d_(n, lam, Gamma, D, scipy.optimize.least_squares)
```

```
* * * * * | lam = 0.2 | * * * * *
alpha: 32.600415968757105
Gamma: [80 66 66]
D: [0.00813219 0.00628029 0.00628029]
inverse D: [122.96817627 159.22832572 159.22832572]
obj: [1.22708161e-08 4.21065943e-10 4.46130979e-10]
s: 0.0016939908920452953
* * * * * * * * * * * * * * * * * * * *


C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\1453620363.py:3:
RuntimeWarning: invalid value encountered in sqrt
  ((1-lam)*sum((lam*X[1:] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma) -
X[0]*d/n*np.prod(X[1:]**(-1/d)))**2,
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\1453620363.py:3:
RuntimeWarning: invalid value encountered in power
  ((1-lam)*sum((lam*X[1:] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma) -
X[0]*d/n*np.prod(X[1:]**(-1/d)))**2,
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\1453620363.py:4:
RuntimeWarning: invalid value encountered in sqrt
  (lam * ((lam*X[1] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[0]) -
2*X[0]/n/X[1])**2,
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\1453620363.py:5:
RuntimeWarning: invalid value encountered in sqrt
  (lam * ((lam*X[2] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[1]) -
2*X[0]/n/X[2])**2,
C:\Users\Ziming Huang\AppData\Local\Temp\ipykernel_14016\1453620363.py:6:
RuntimeWarning: invalid value encountered in sqrt
  (lam * ((lam*X[3] + (1-lam)*np.prod(X[1:]**(1/d)))**(-1) - Gamma[2]) -
2*X[0]/n/X[3])**2
```

Numerical instability arises when dimension increases.