KRZYSZTOF SOKÓŁ-SZOŁTYSEK

PROGRAM 4

GRUPA PONIEDZIAŁKOWA

WYNIKI

A. N = 192

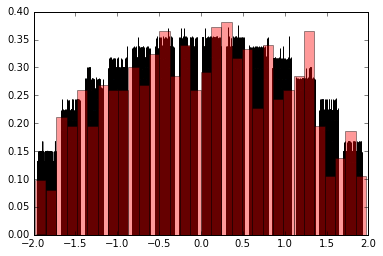
C. Dla mojego N i próby 600 macierzy otrzymałem wyniki :

mean t: 192.060681577

standard deviation t: 1.4544023297

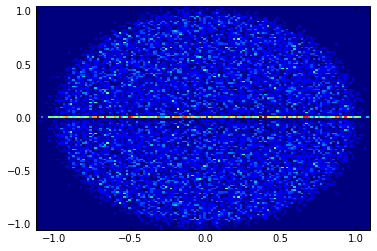
Jak zauważyłem przeciętna wartość widma bardzo nieznacznie odbiega od zadanego dowolnego N(niska standardowa dewiacja).

D.



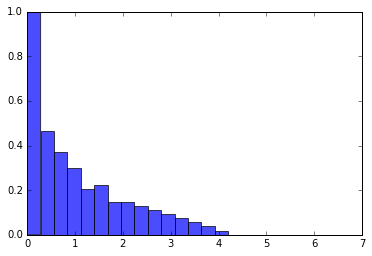
Dla 600 macierzy \* 192 wartości własne = 115200 wartości własnych (bins = sqrt, czarny kolor) uzyskałem po unormowaniu wykres który przypomina rozklad półkolisty Wignera(100 losowych próbek z rozkładu, kolor czerwony, R = 2) - rozkład wartości własnych dużych losowych macierzy jest ograniczony właśnie w ten sposób (Winger semicircle law).

E.



Wartości własne losowej macierzy ułożyły się w koło, większe zagęszczenie na osi rzeczywistej - zgodnie z Circular Law.

F.



Po przeskalowaniu widzimy, że zgodnie z oczekiwaniami wartości osobliwe dużych prostokątnych macierzy losowych przybierają postać rozkładu Marchenko-Pastur.

G.

Wyniki z kilku pomiarów :

mean t: 191.99160921

standard deviation t: 1.69212032869

Kappa:

mean K:

2261.04404122

std K:

4962.3326014

log(Kappa)

mean logarithm:

6.85604239216

logarithmic average:

949.601470807

>>> runfile('/home/dm/metodynumeryczne/4/IV.py', wdir='/home/dm/metodynumeryczne/4')

mean t: 191.899343097

standard deviation t: 1.35008456548

Kappa:

mean K:

1946.50150317

std K:

3545.32783364

log(Kappa)

mean logarithm:

6.85718717909

logarithmic average:

950.689184633

>>> runfile('/home/dm/metodynumeryczne/4/IV.py', wdir='/home/dm/metodynumeryczne/4')

mean t: 191.926652046

standard deviation t: 1.40654264565

Kappa:

mean K:

2610.58548579

std K:

7151.7701397

log(Kappa)

mean logarithm:

6.8065711523

logarithmic average:

903.766610019

>>> runfile('/home/dm/metodynumeryczne/4/IV.py', wdir='/home/dm/metodynumeryczne/4')

mean t: 192.128003788

standard deviation t: 1.40436826455

Kappa:

mean K:

2221.22474874

std K:

4703.18785279

log(Kappa)

mean logarithm:

6.89017502977

logarithmic average:

982.573381765

>>> runfile('/home/dm/metodynumeryczne/4/IV.py', wdir='/home/dm/metodynumeryczne/4')

mean t: 192.098304731

standard deviation t: 1.46677333943

Kappa:

mean K:

1039.48679441

std K:

1506.0901049

log(Kappa)

mean logarithm:

6.50582850173

logarithmic average:

669.029732092

Zauważalna jest duża standardowa dewiacja średniej arytmetycznej i stosunkowo duże odchyły niej samej. Dla danych o znacznym “rozrzucie” rozsądniejsza okazuje sie średnia logarytmiczna, która daje bardziej powtarzalne rezultaty.

LISTING

# -\*- coding: utf-8 -\*-

"""

Spyder Editor

This is a temporary script file.

"""

#import Gnuplot

import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as ss

import pylab as pl

#A.

n = 192

Matricescount = 100#600 for D. ; 1 for F.

#B.

#print(G)

G = [np.matrix(np.sqrt(1/n) \* np.random.randn(n, n)) for i in range(Matricescount)] # np.sqrt(1/n) \* np.mat

#C.

#print(GxGt)

#print(t)

#Z=np.matrix([[1,2,3],[4,5,6],[1,1,1]])

#print(Z)

#print(np.trace(Z))

GxGt = [g \* np.transpose(g) for g in G]

t = [np.trace(gxgt) for gxgt in GxGt]

#print(t)

print("mean t:", end= ' ')

print(np.mean(t))

print("standard deviation t:", end= ' ')

print(np.std(t))

#D.

#print(H)

#print("Eigenvalues:")

#print(EV)

#EV2=[np.linalg.eigvals(h) for h in H]

#print("ev2")

#print(EV2)

#r= cauchy.pdf(0.5)

#plt.hist(r, normed=True, bins='auto',color='r')

#np.histogram

#plt.subplots

#np.linspace

#zet=np.linspace(0,2\*np.pi,400)

#plt.hist(zet,bins='auto')

#plt.show

#print("ev done")

#s = s[(s>-3) & (s<3)] # truncate distribution so it plots well

#plt.plot(s)

#plt.show()

#hist = Gnuplot.Gnuplot()

#Z=np.matrix([[1,2,3],[4,5,6],[1,1,1]])

#print(Z\*np.transpose(Z))

H=[((g + np.transpose(g)) / np.sqrt(2)) for g in G]

EV=[np.linalg.eigvalsh(h) for h in H]

plt.hist(EV, normed=True, bins='sqrt')

r = ss.semicircular.rvs(size=100, scale=2)

plt.hist(r, bins='sqrt',normed=True,color='r',alpha=0.4)

plt.show()

#E.

#plt.hist(cEV, bins=350)

#plt.show()

#print(G)

#print(cEV)

#print(Real)

#print(Imag.ravel())

cEV = [np.linalg.eigvals(g) for g in G]

Real = np.real(cEV)

Imag = np.imag(cEV)

plt.hist2d(Real.ravel(),Imag.ravel(), bins=np.sqrt(n \* Matricescount), normed = True)

plt.show()

#F.

#print(rEV)

#print(revx4)

#print(m\_px)

#m\_py = m\_py[(m\_py>0) & (m\_py<4)]

W = [gxgt / np.trace(gxgt) for gxgt in GxGt]

rEV = [np.linalg.eigvalsh(w) for w in W]

x=[rev \* n for rev in rEV]

plt.hist(x, alpha = 0.7, bins='sqrt', normed = True, range = [0,7])

m\_px = np.asarray(x)

m\_py = (1/2\*np.pi) \* np.sqrt((4-m\_px)/m\_px)

pl.ylim([0,1])

pl.plot(m\_px, m\_py)

#G.

SV = [np.linalg.svd(g, compute\_uv = False) for g in G]

Sigmamax = [np.amax(sv) for sv in SV]

Sigmamin = [np.amin(sv) for sv in SV]

#print(Sigmamax)

#print(Sigmamin)

K = [np.amax(sv) / np.amin(sv) for sv in SV]

#print(K)

print("Kappa:")

#print(K)

print("mean K:")

print(np.mean(K))

print("std K:")

print(np.std(K))

print("log(Kappa)")

#print(np.log(K))

print("mean logarithm:")

print(np.mean(np.log(K)))

print("logarithmic average:")

print(np.exp(np.mean(np.log(K))))