Machine Intelligence 2

Exercise 07

Kurtosis, negentropy, and the independent components of image patches

Group Members:
Xugang Zhou
Fangzhou Yang

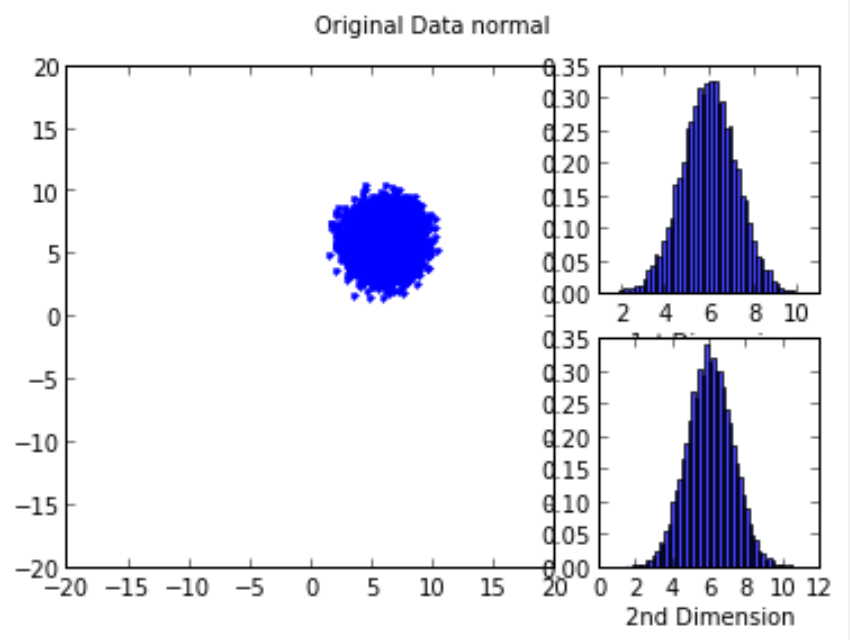
Tutor:
Timm Lochmann

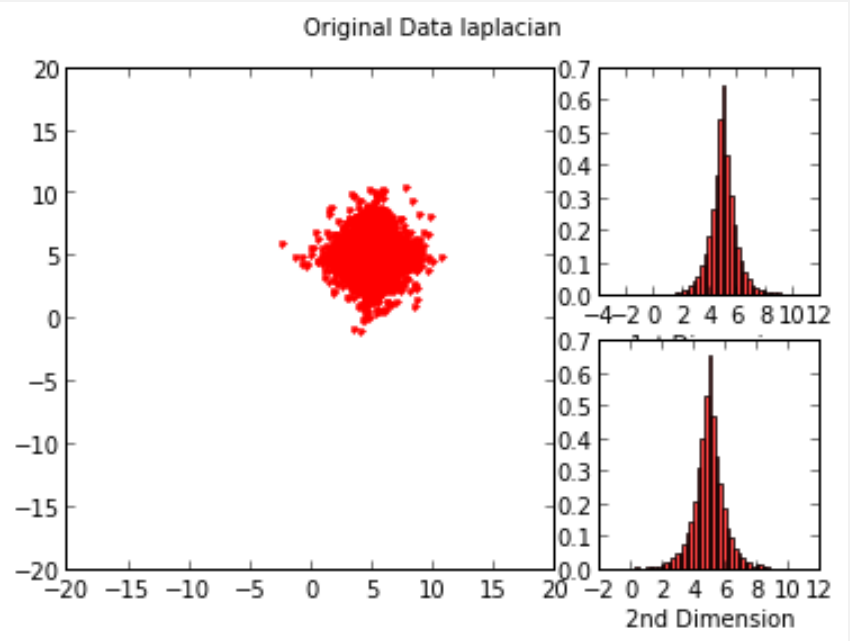
June 13, 2013

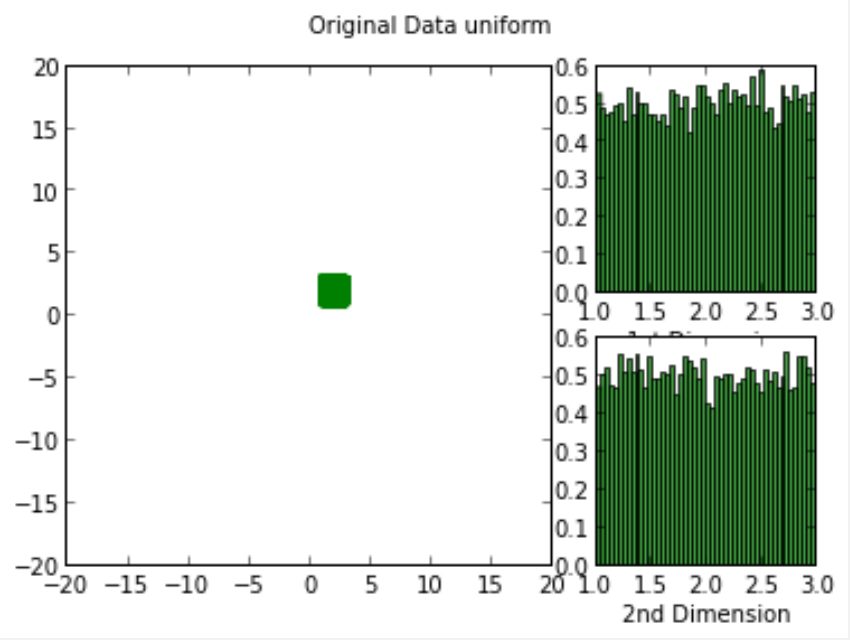
1 7.1 Kurtosis of Toy Data

```
from numpy import *
from scipy import linalg, io
from scipy.linalg import sqrtm, inv
from scipy.io import loadmat
import matplotlib
import matplotlib.pyplot as plt
from PrincipalComponentsTool import get_PC
from math import *

def plotSaH(X, Y, Name, c, ran):
    fig = plt.figure()
    plt.suptitle(Name)
    ax = plt.subplot2grid((2, 3), (0, 0), rowspan = 2, colspan = 2)
    ax.axis([-ran, ran, -ran, ran])
    ax.plot(X, Y, c + '.')
    ax = plt.subplot2grid((2, 3), (0, 2))
    ax.hist(X, 50, normed = 1, facecolor = c, alpha = 0.75)
    ax.set_xlabel("1st Dimension")
    ax = plt.subplot2grid((2, 3), (1, 2))
    ax.hist(Y, 50, normed = 1, facecolor = c, alpha = 0.75)
    ax.set_xlabel("2nd Dimension")
    #fig.savefig(Name+".png")
```

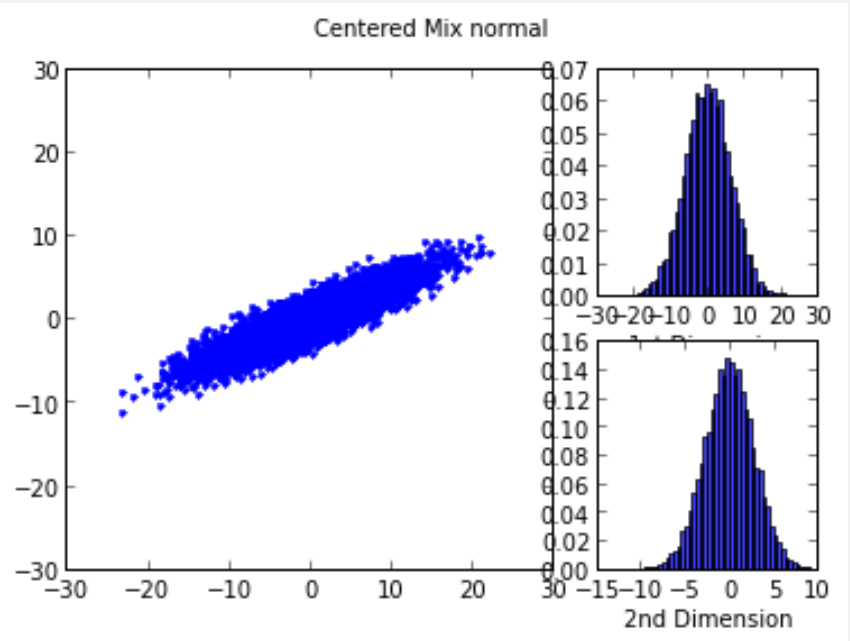
```
matData = loadmat("datafilesICA/distrib.mat")

dic = ["normal", "laplacian", "uniform"]
color = ['b', 'r', 'g']

data = [0 for i in range(3)]
A = [[4, 3],[1, 2]]
for i in range(3):
    data[i] = matData[dic[i]]
    plotSaH(data[i][0], data[i][1], "Original Data " + dic[i], color[i], 20)
```

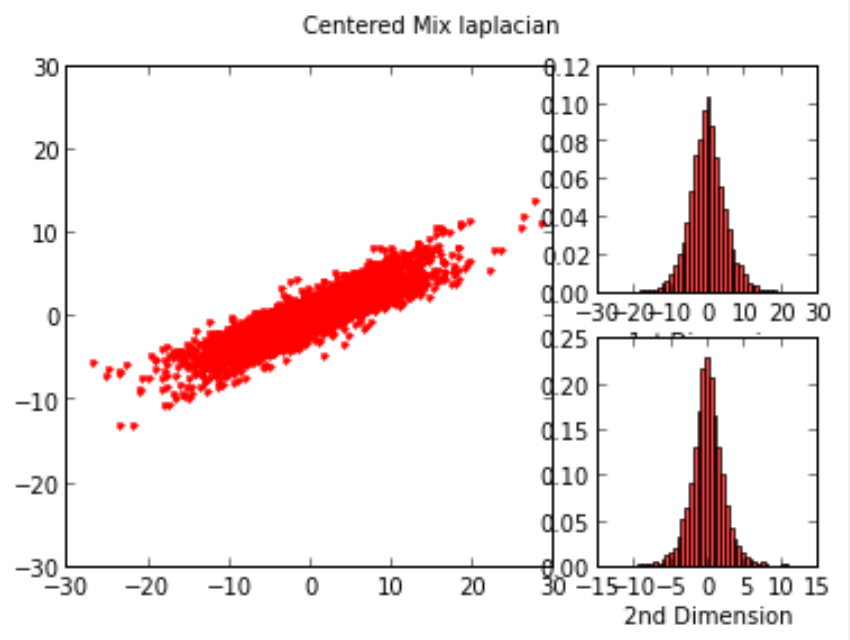

Original Data normal
2nd Dimension

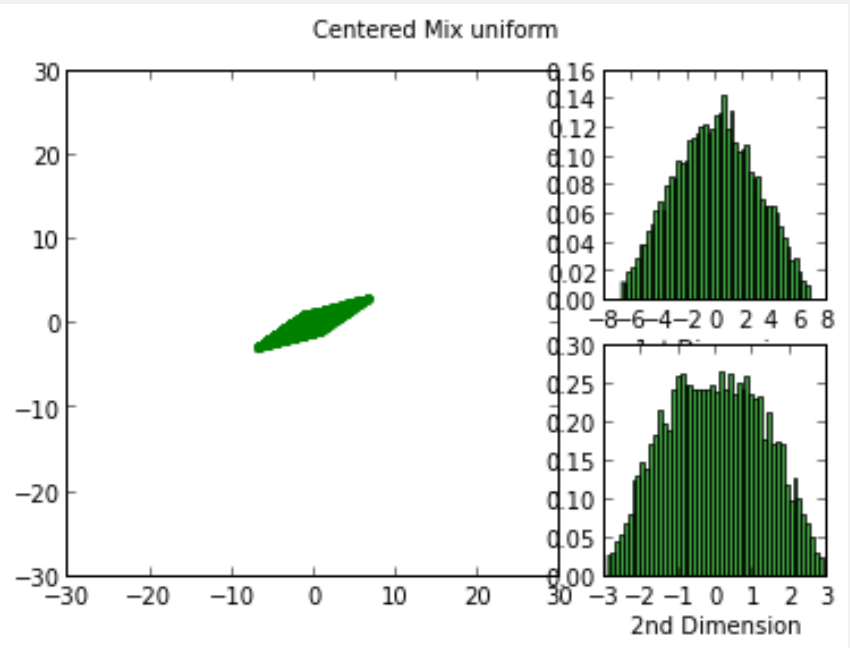

Original Data laplacian
2nd Dimension




```
mixed = [0 for i in range(3)]
for i in range(3):
    mixed[i] = [[sum(A[j][k] * data[i][k][l] for k in range(2)) for l in range
        (10000)] for j in range(2)]
    m0 = sum(mixed[i][0]) / 10000
    m1 = sum(mixed[i][1]) / 10000
    mixed[i][0] = [mixed[i][0][j] - m0 for j in range(10000)]
    mixed[i][1] = [mixed[i][1][j] - m1 for j in range(10000)]
    plotSaH(mixed[i][0], mixed[i][1], "Centered Mix " + dic[i], color[i], 30)
```

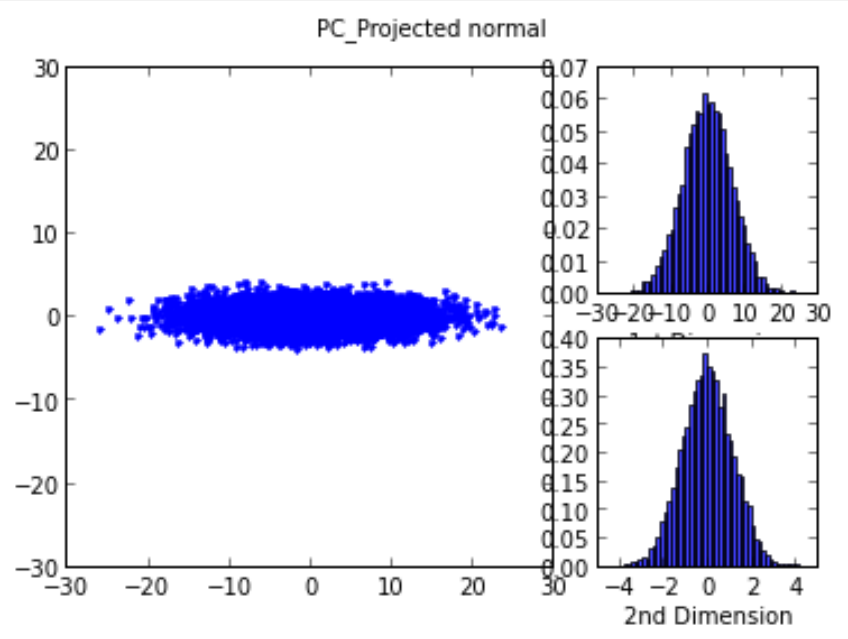


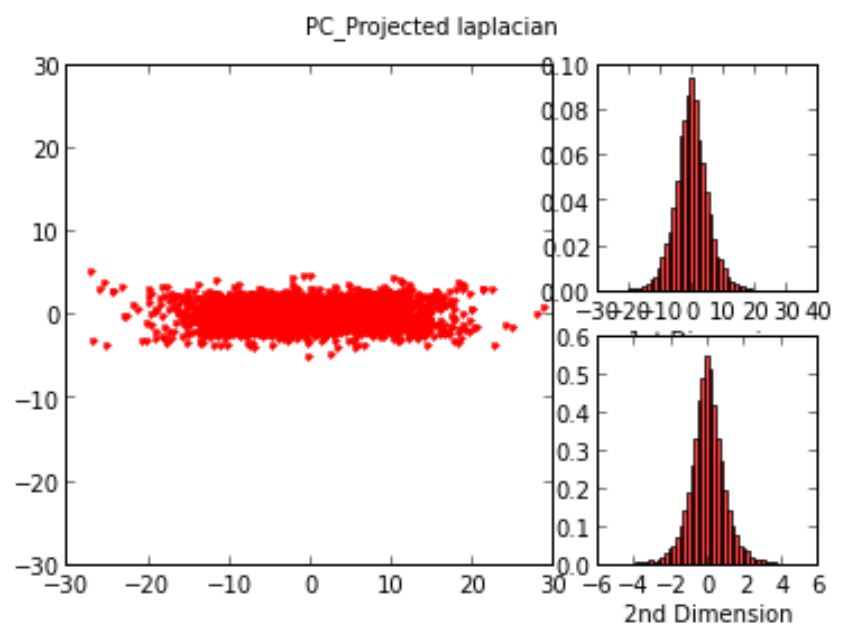


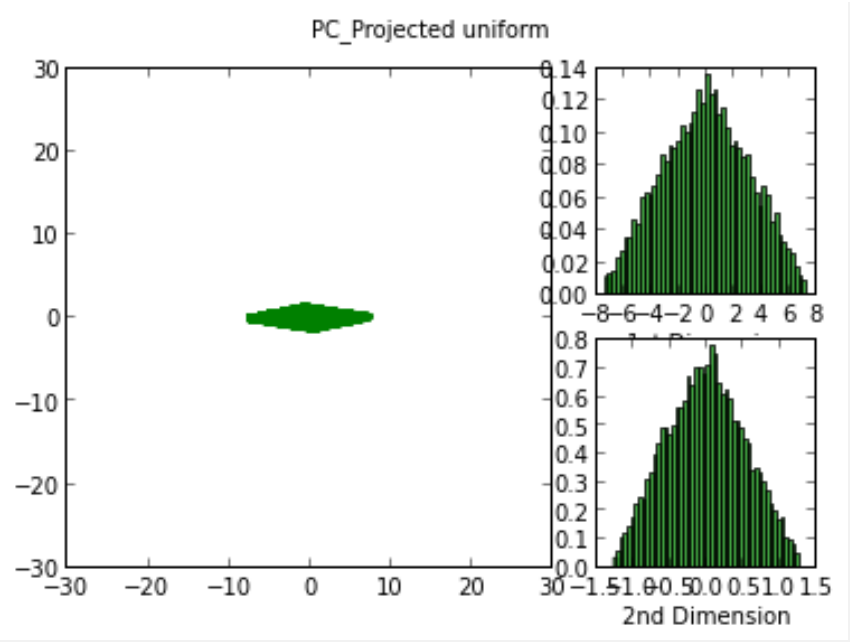



```
projected = [[0 for i in range(2)] for j in range(3)]
whiten = [0 for j in range(3)]
for i in range(3):
    evals, evecs = get_PC(mixed[i], 2, 2)
    projected[i][0] = [mixed[i][0][j] * evecs[0, 0] + mixed[i][1][j] * evecs
        [1, 0] for j in range(10000)]
    projected[i][1] = [mixed[i][0][j] * evecs[0, 1] + mixed[i][1][j] * evecs
        [1, 1] for j in range(10000)]
    plotSaH(projected[i][0], projected[i][1], "PC_Projected " + dic[i], color[
        i], 30)
    whiten[i] = dot(dot(array(mixed[i]).T, array(evecs)), inv(sqrtm(diag(evals
```

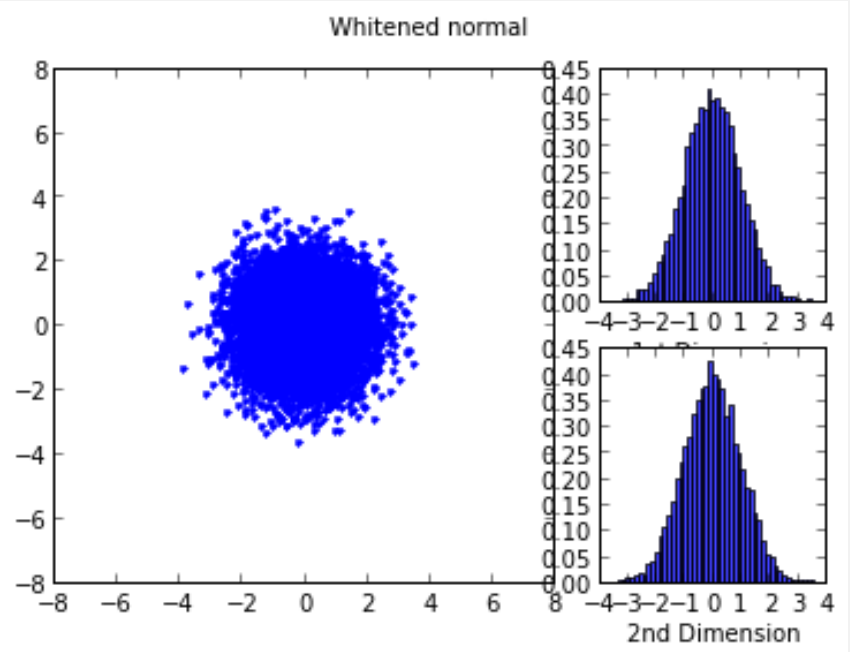
```
real  T
```

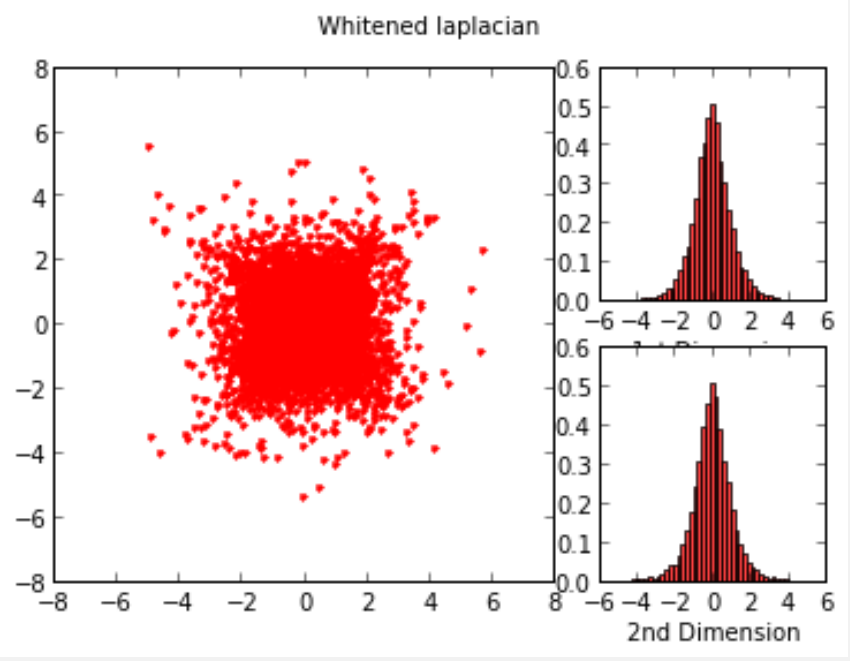

PC_Projected normal
2nd Dimension

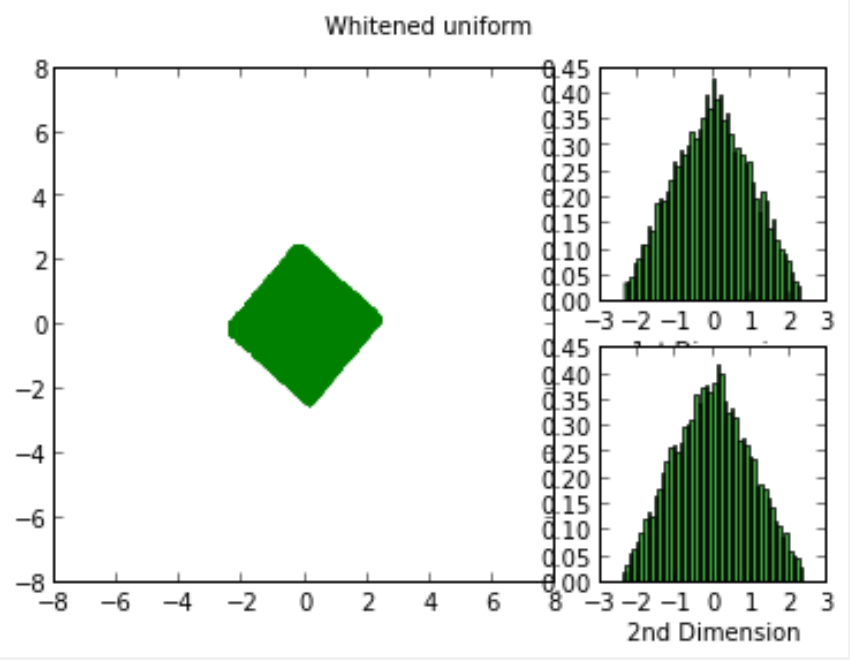

PC_Projected laplacian
2nd Dimension




```
for i in range(3):
    plotSaH(whiten[i][0], whiten[i][1], "Whitened " + dic[i], color[i], 8)
```

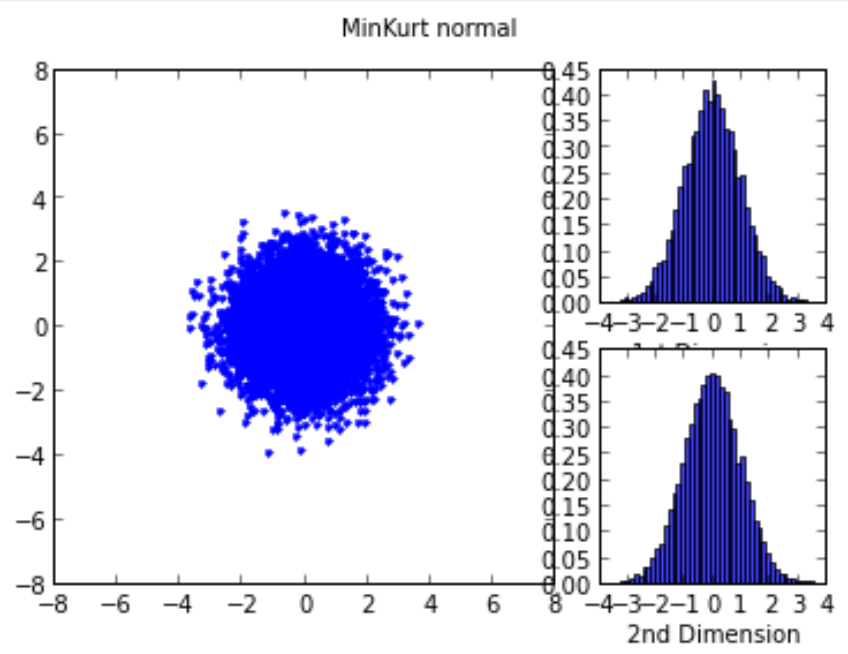


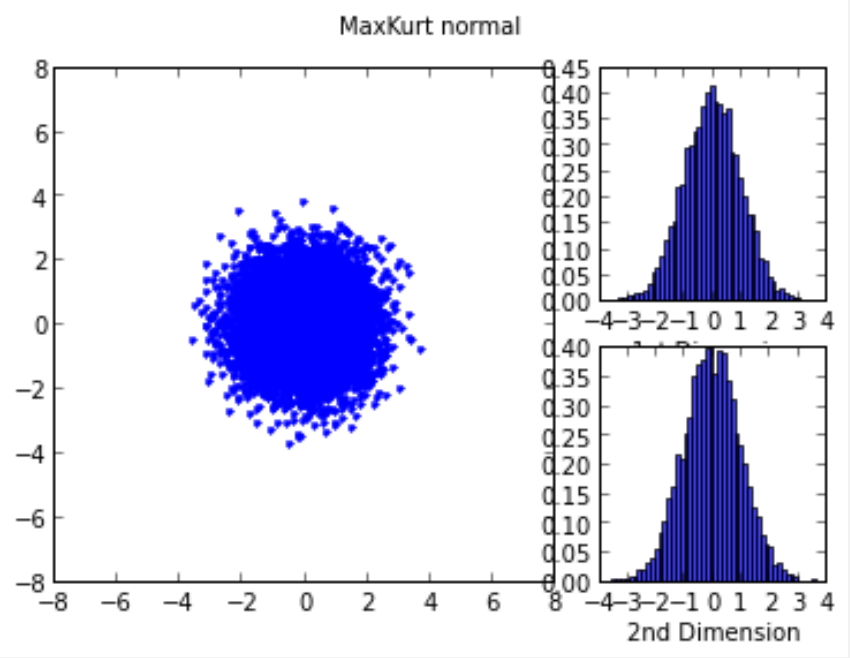


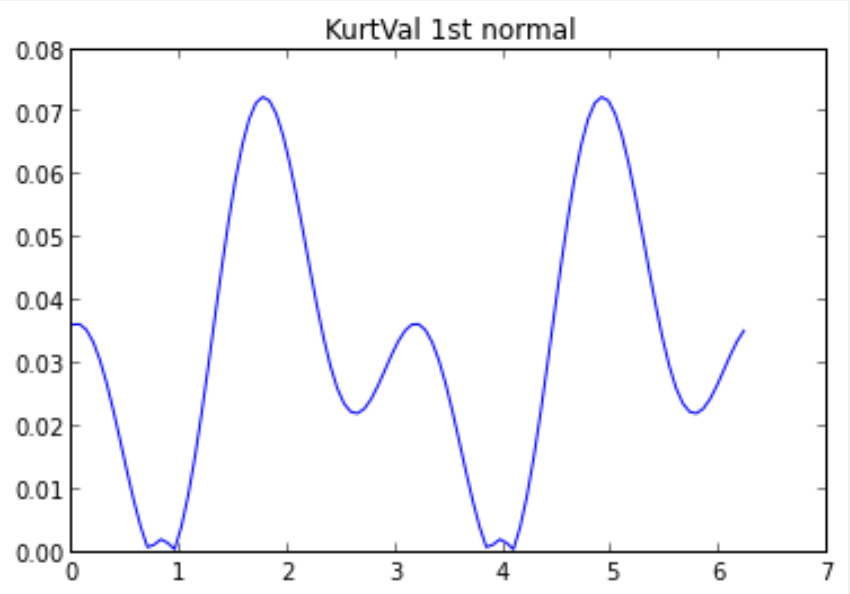


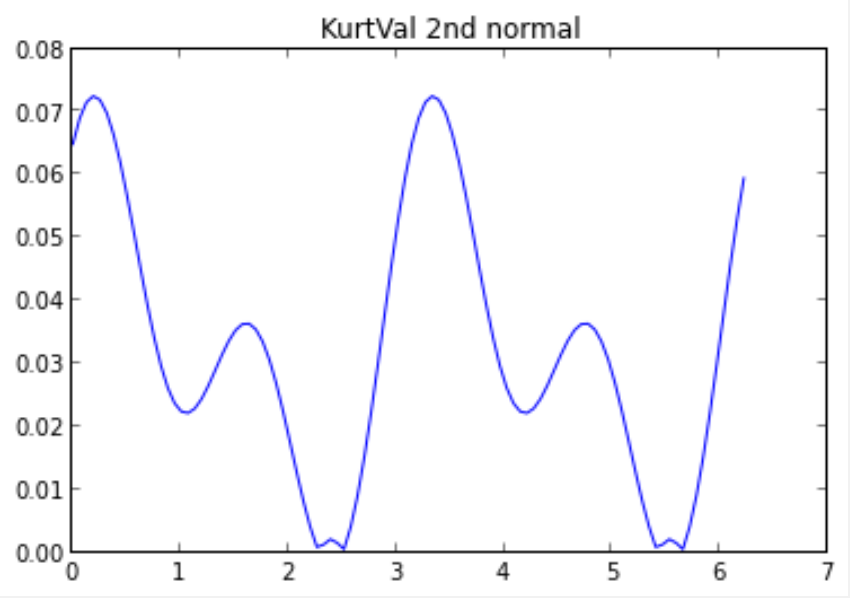

```
theta = [i * pi / 50 for i in range(100)]
kurtVal1 = [0 for i in range(100)]
kurtVal2 = [0 for i in range(100)]
maxData = matrix([[0 for i in range(10000)] for j in range(2)])
minData = matrix([[0 for i in range(10000)] for j in range(2)])
for i in range(3):
    minKurt = 10000
    maxKurt = -10000
    for k in range(100):
        R = array([[cos(theta[k]), -sin(theta[k])],[sin(theta[k]), cos(theta[k
```

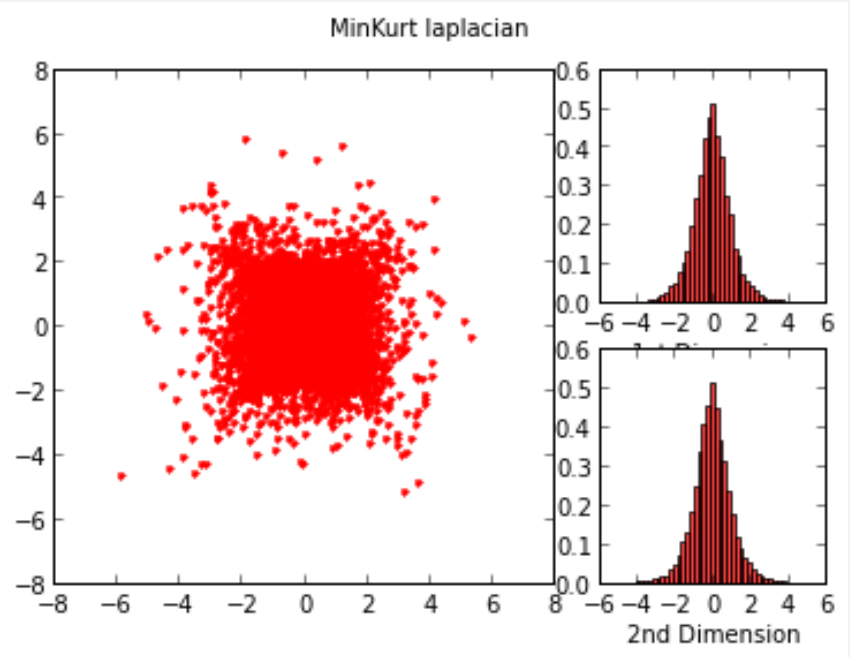
```
    tempData   dot R  whiten i
    kurtVal1 k
    kurtVal2 k
    for j in range
        kurtVal1 k     tempData    j
        kurtVal2 k     tempData    j
    kurtVal1 k    abs kurtVal1 k
    kurtVal2 k    abs kurtVal2 k
    if kurtVal1 k    maxKurt
        maxKurt   kurtVal1 k
        maxData   tempData
    if kurtVal1 k    minKurt
        minKurt   kurtVal1 k
        minData   tempData
plotSaH minData        minData        "MinKurt "   dic i   color i
plotSaH maxData        maxData        "MaxKurt "   dic i   color i
fig   plt figure
ax   fig add_subplot
ax plot theta  kurtVal1  color i    '-'
ax set_title "KurtVal 1st "   dic i
fig savefig "KurtVal 1st "   dic i    ".png"
fig   plt figure
ax   fig add_subplot
ax plot theta  kurtVal2  color i    '-'
ax set_title "KurtVal 2nd "   dic i
fig savefig "KurtVal 2nd "   dic i    ".png"
```

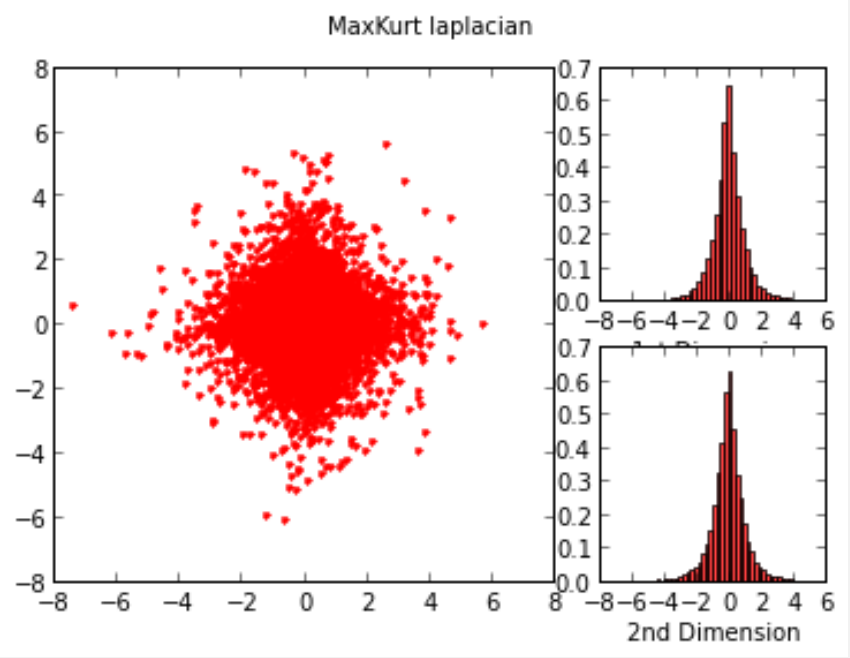


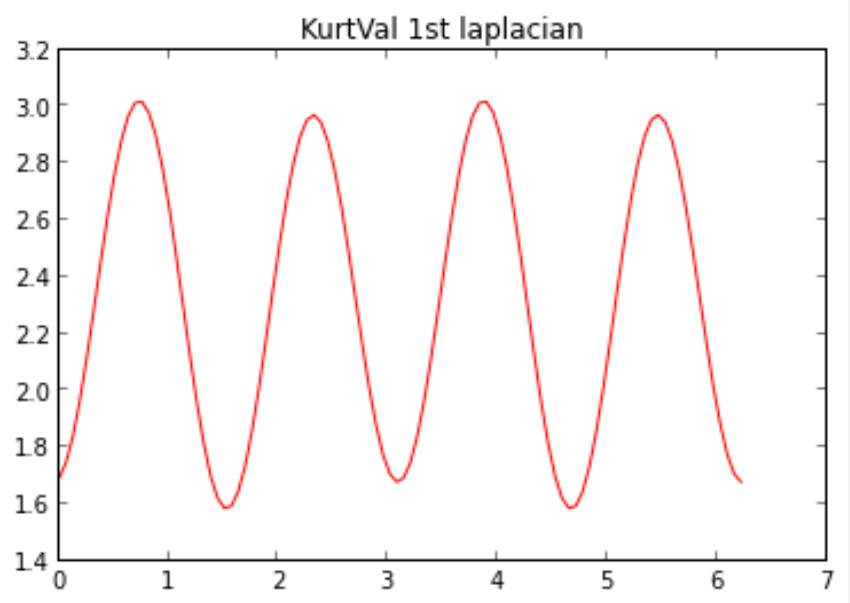

MaxKurt normal
2nd Dimension

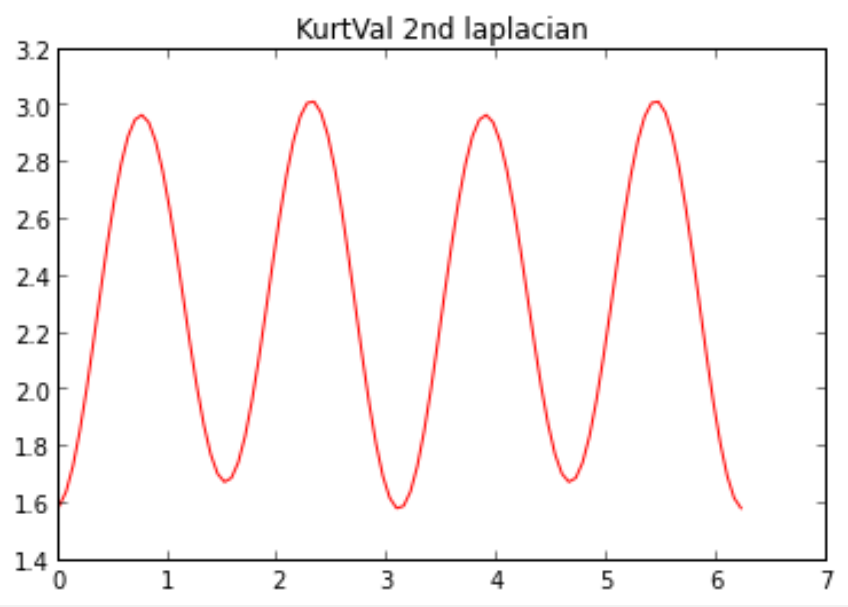

KurtVal 1st normal

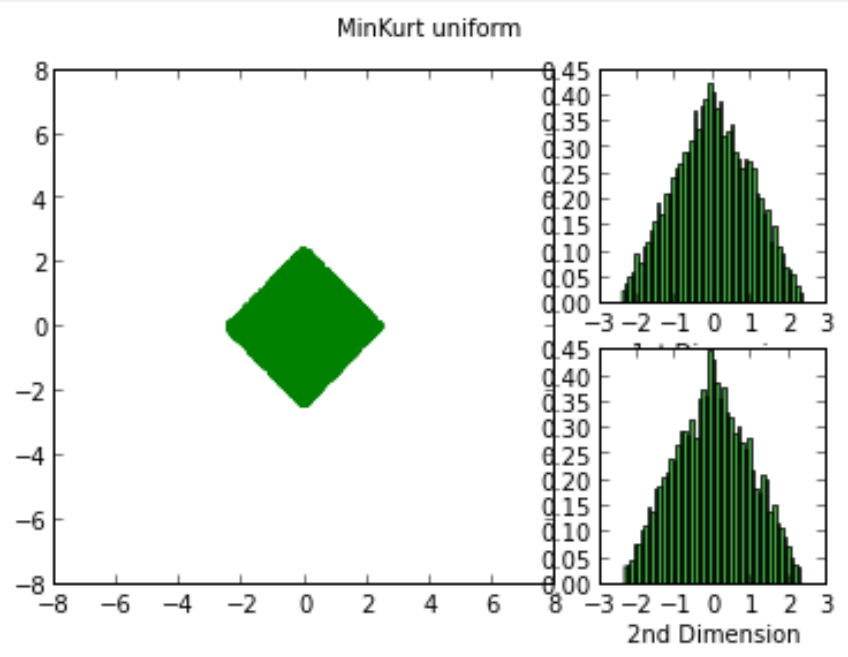

KurtVal 2nd normal
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
1
2
3
4
5
6
7

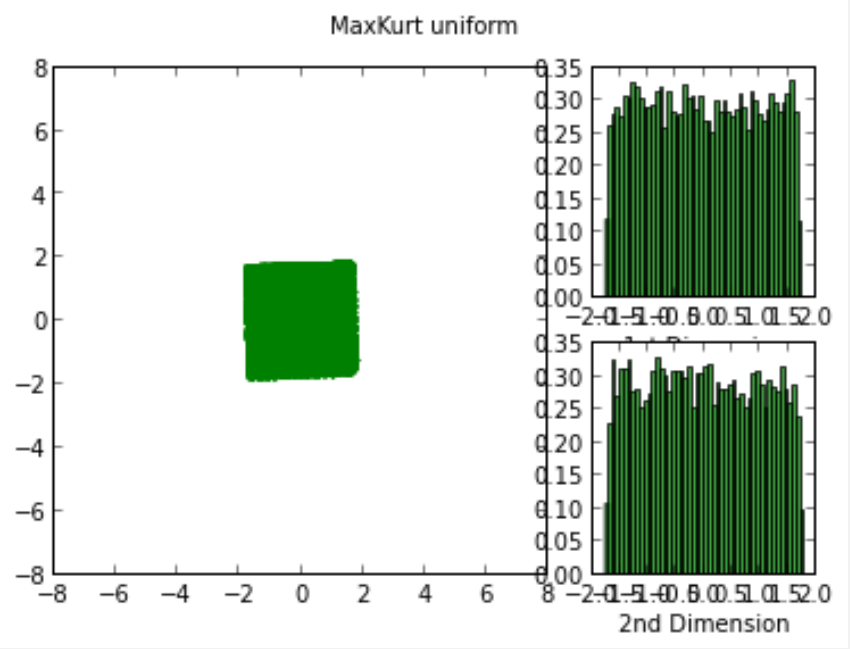

MinKurt laplacian
8
6
4
2
0
−2
−4
−6
−8
−8 −6 −4 −2 0 2 4 6 8
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−6 −4 −2 0 2 4 6
2nd Dimension

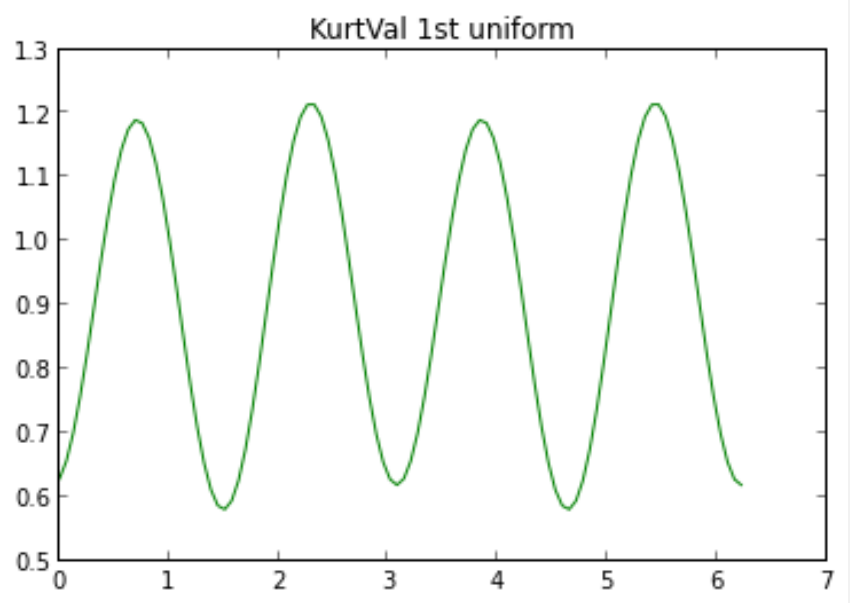

MaxKurt laplacian
2nd Dimension

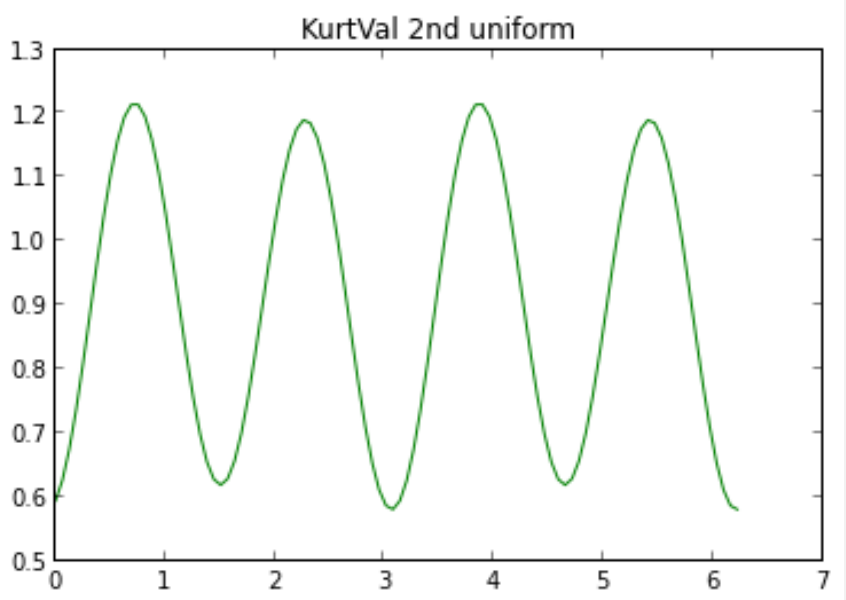

KurtVal 1st laplacian


KurtVal 2nd laplacian
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
0
1
2
3
4
5
6
7


MinKurt uniform
8
6
4
2
0
−2
−4
−6
−8
−8 −6 −4 −2 0 2 4 6 8
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−3 −2 −1 0 1 2 3
2nd Dimension


MaxKurt uniform
2nd Dimension


KurtVal 1st uniform

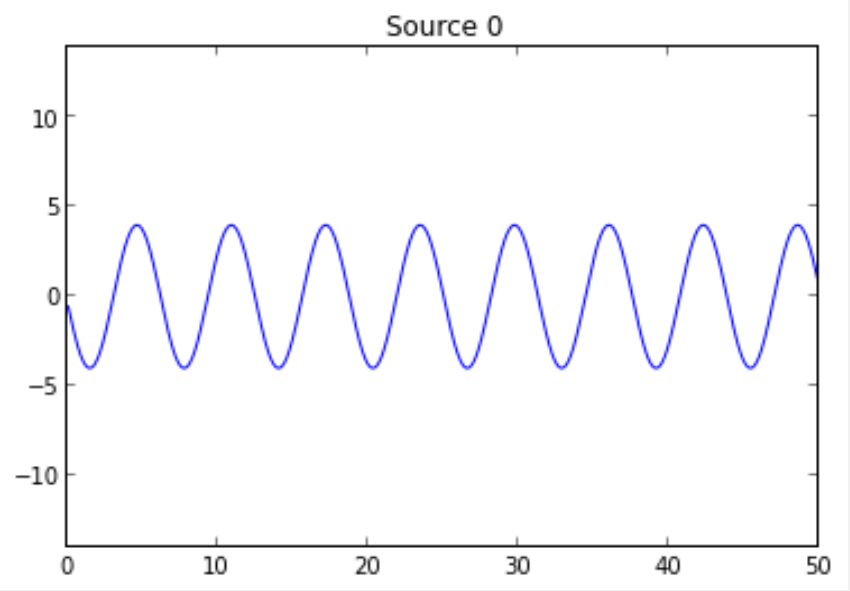


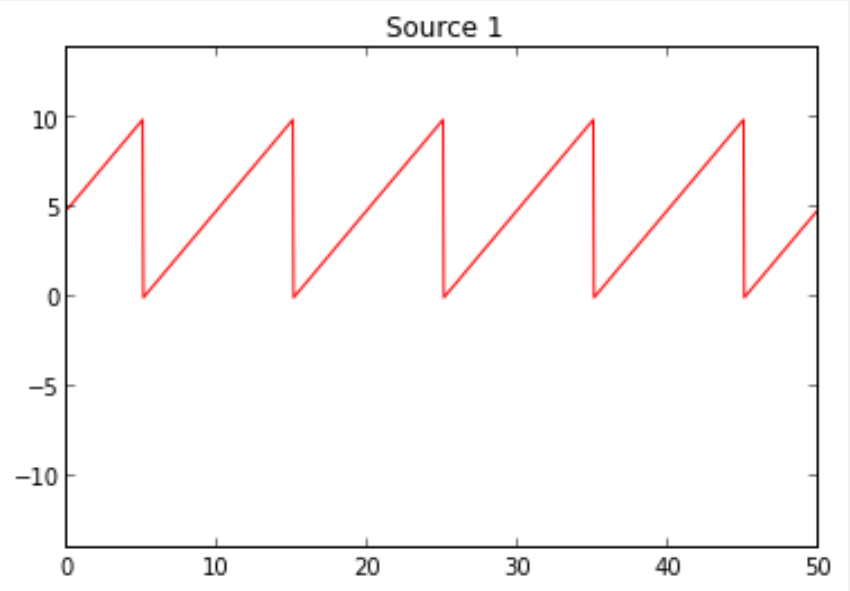
2 7.2 Toy Singal Seperation

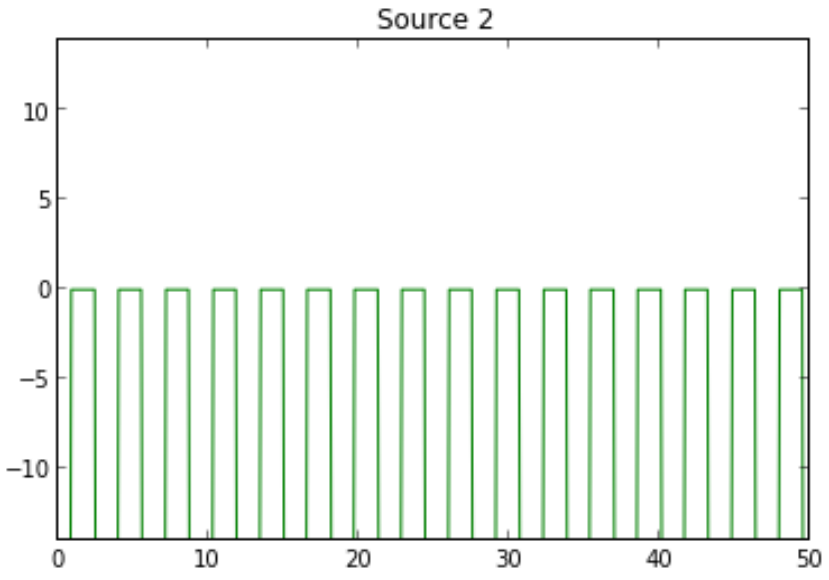
```
xaxis = [i * 0.05 for i in range(1000)]

def plotSig(X, Name, c, ran):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(xaxis, X, c + '-')
    ax.set_title(Name)
    ax.axis([0, 50, -ran, ran])
    #fig.savefig(Name + ".png")
```

```
color = ['b', 'r', 'g']
s = [0 for i in range (3)]
s[0] = [4 * sin(i * 0.05 - 3) for i in range(1000)]
s[1] = [(i * 0.05 + 5) % 10 for i in range(1000)]
s[2] = [(-14 if cos(2 * 0.05 * i) > 0 else 0) for i in range(1000)]
```

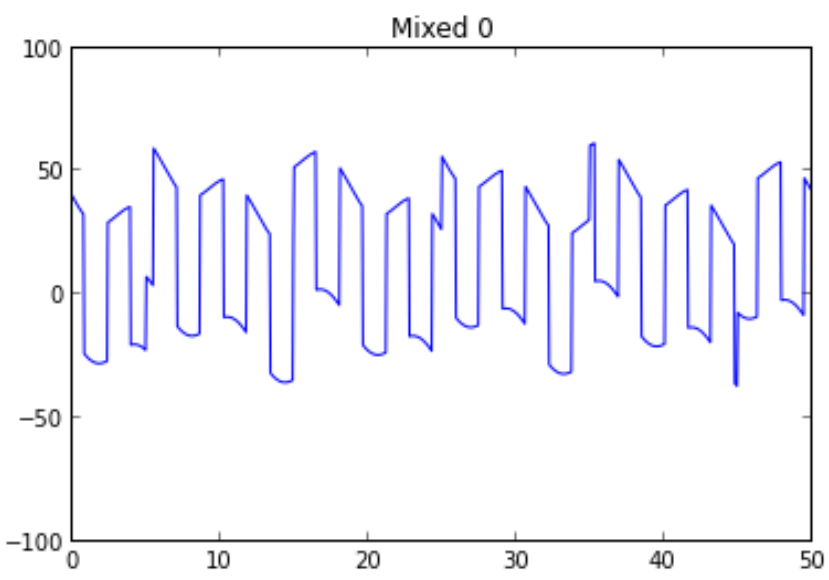
```
for i in range(3):
    plotSig(s[i], "Source " + str(i), color[i], 14)

A = array([[2, -3, -4], [7, 5, 1], [-4, 7, 5]])
x = dot(A , s)
```

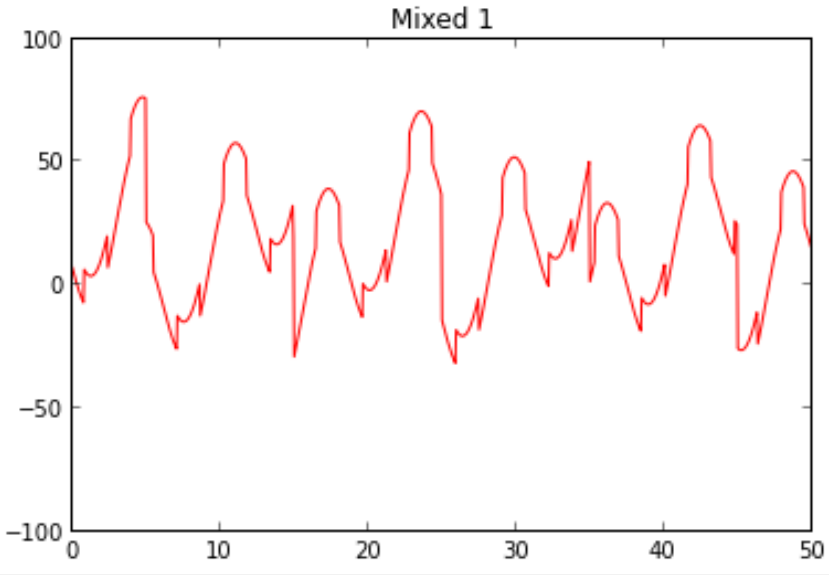

Source 0
10
5
0
−5
−10
0
10
20
30
40
50

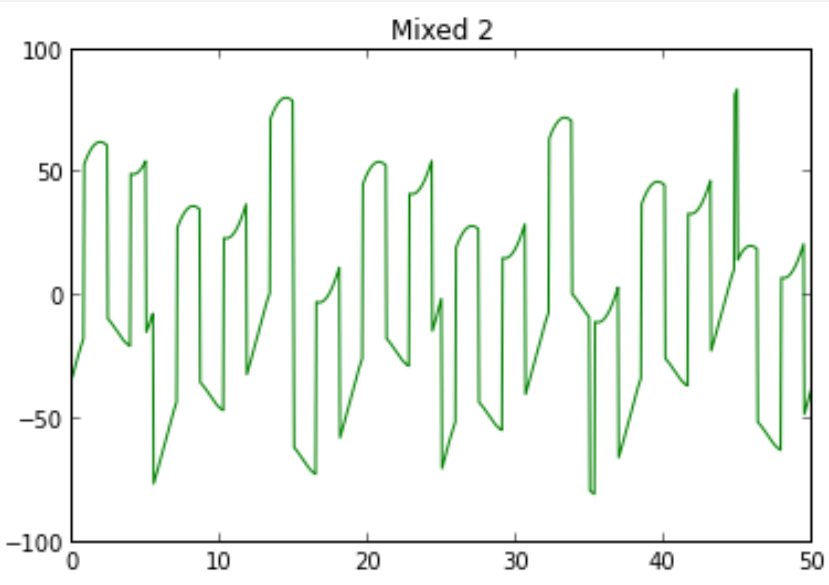

Source 1
10
5
0
−5
−10
0
10
20
30
40
50



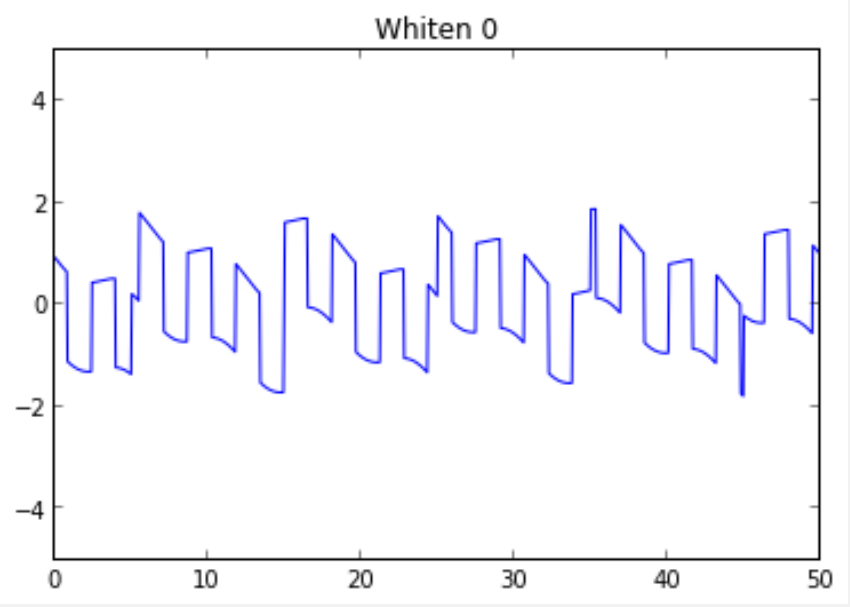

```
for i in range(3):
    plotSig(x[i, :], "Mixed " + str(i), color[i], 100)
```

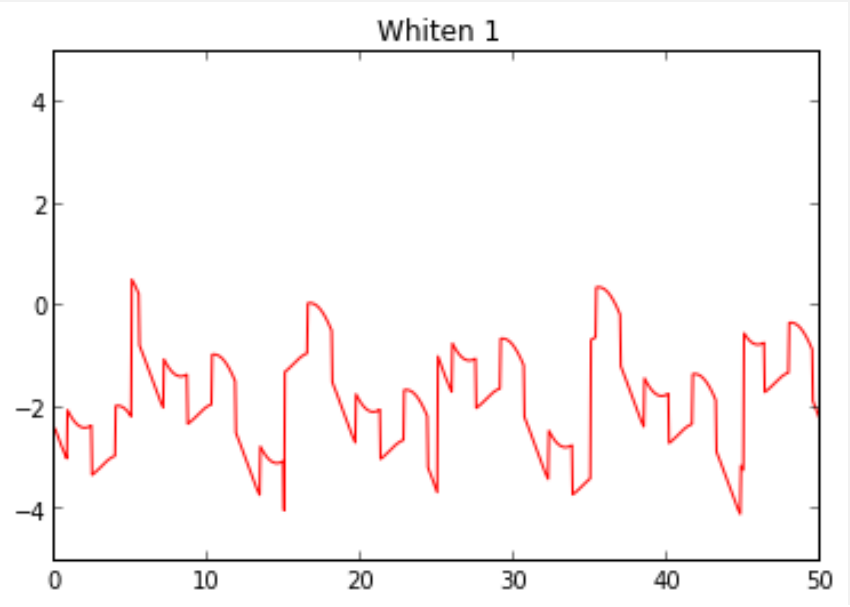


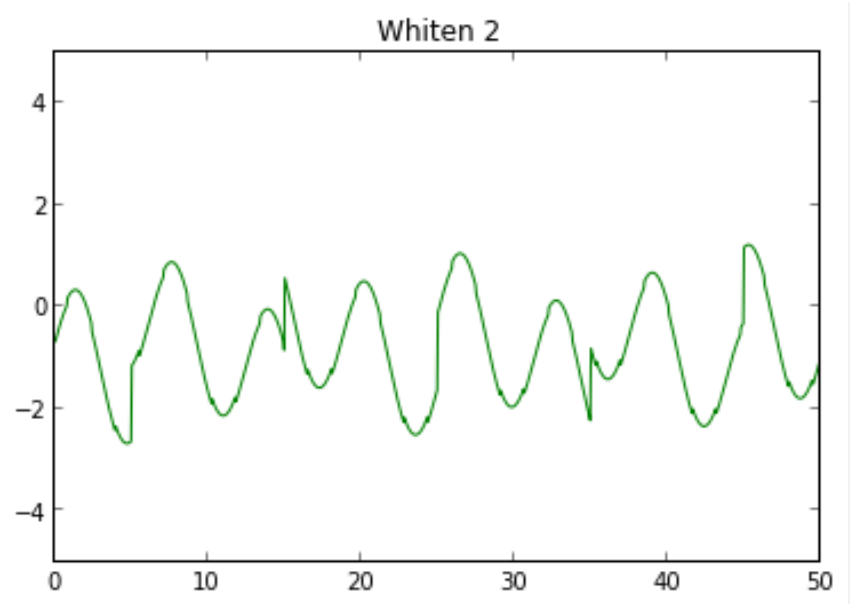




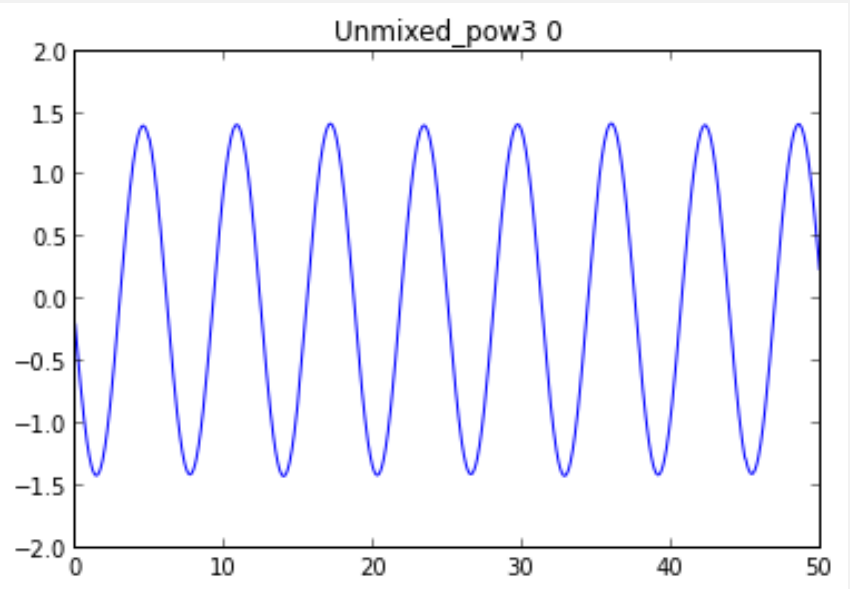

```
evals, evecs = get_PC(x, 3, 3)
whiten = dot(dot(x.T, array(evecs)), inv(sqrtm(diag(evals))).real).T
for i in range(3):
    plotSig(whiten[i, :], "Whiten " + str(i), color[i], 5)
```

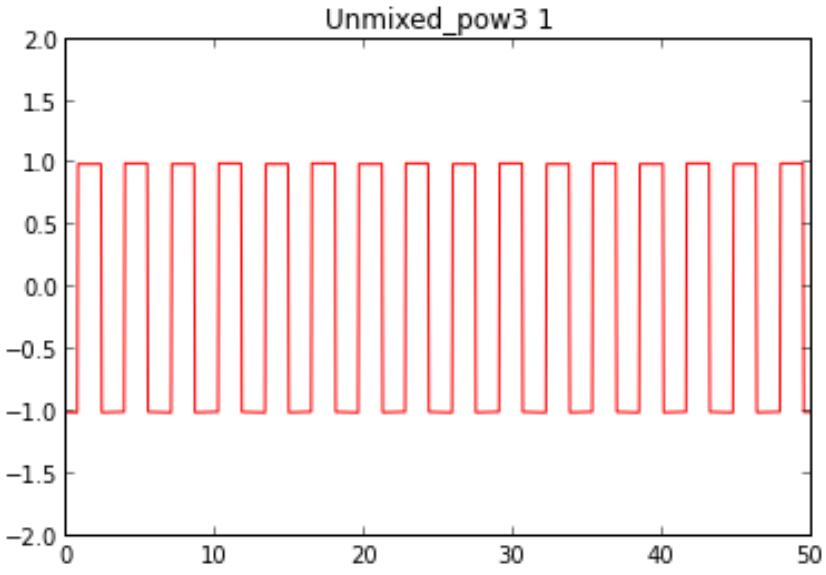

Whiten 0
4
2
0
−2
−4
0
10
20
30
40
50

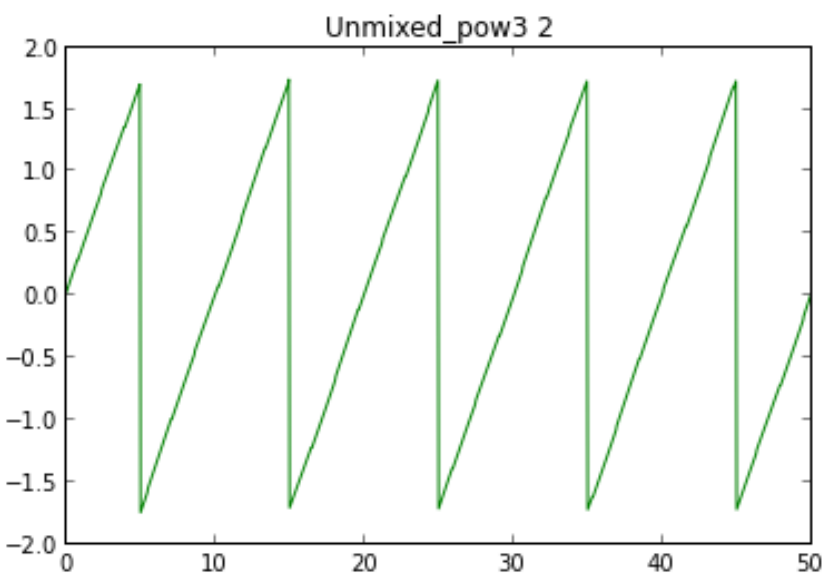

Whiten 1
4
2
0
−2
−4
0
10
20
30
40
50



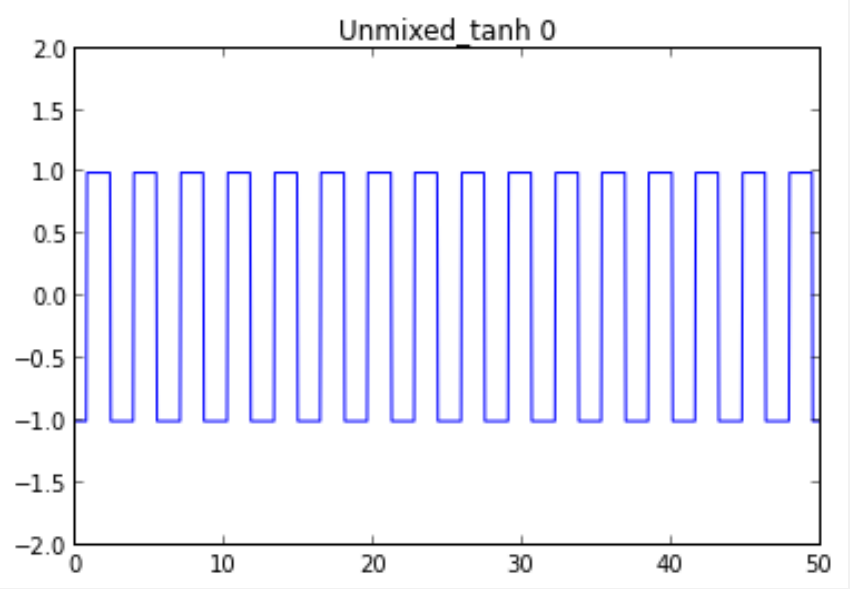

```
import mdp
y = mdp.fastica(whiten.T, g = 'pow3').T
for i in range(3):
    plotSig(y[i, :], "Unmixed_pow3 " + str(i), color[i], 2)
```

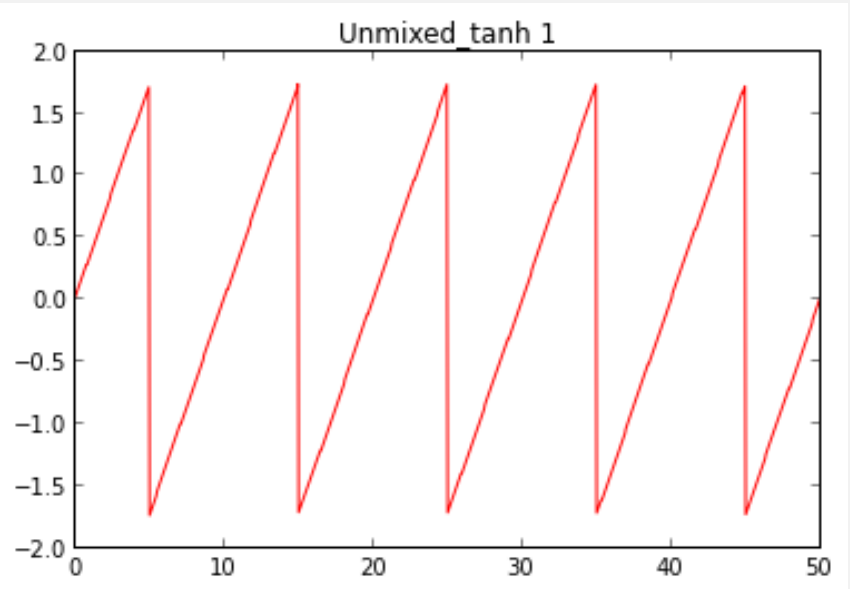


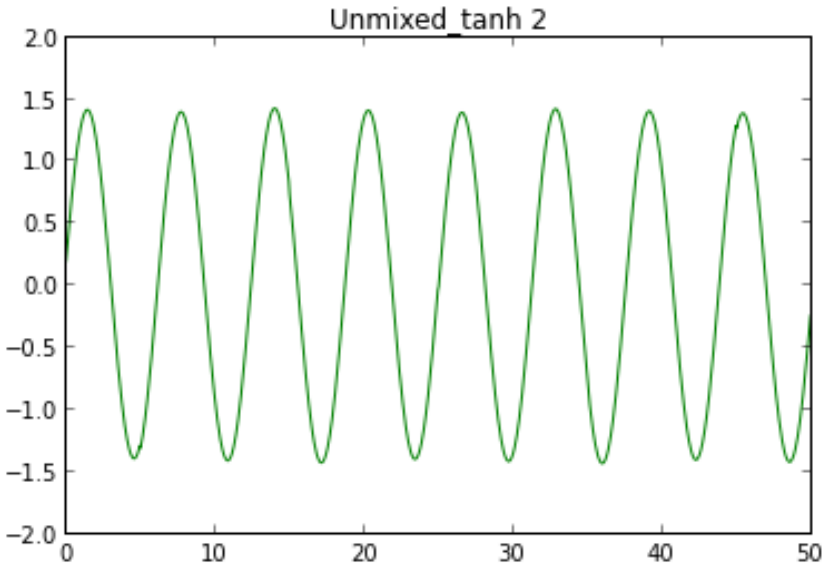





```
y = mdp.fastica(whiten.T, g='tanh').T
for i in range(3):
    plotSig(y[i, :], "Unmixed_tanh " + str(i), color[i], 2)
```


Unmixed_tanh 0
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
10
20
30
40
50


Unmixed_tanh 1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
10
20
30
40
50



3 7.3 ICA on Image Patches

```
from mahotas import imread
from numpy import *

N = 16
numP = 2000

buildingData = [[] for i in range(10 * numP)]
for i in range(10):
    readImage = imread('datafilesICA/images/b'+str(i+1)+'.jpg')
    for j in range(numP):
        startPosx = random.randint(0, readImage.shape[0] - N - 1)
        startPosy = random.randint(0, readImage.shape[1] - N - 1)
        sample = readImage[startPosx:startPosx + N, startPosy:startPosy + N]
        buildingData[i * numP + j] = [x for sublist in sample for x in sublist]

savetxt('buildingData', buildingData)

natureData = [[] for i in range(13 * numP)]
for i in range(13):
    readImage = imread('datafilesICA/images/n'+str(i+1)+'.jpg')
    for j in range(numP):
        startPosx = random.randint(0, readImage.shape[0] - N - 1)
        startPosy = random.randint(0, readImage.shape[1] - N - 1)
        sample = readImage[startPosx:startPosx + N, startPosy:startPosy + N]
        natureData[i * numP + j] = [x for sublist in sample for x in sublist]

savetxt('natureData', natureData)

textData = [[] for i in range(14 * numP)]
```

```
for i in range(14):
    readImage = imread('datafilesICA/images/t'+str(i+1)+'.jpg')
    for j in range(numP):
        startPosx = random.randint(0, readImage.shape[0] - N - 1)
        startPosy = random.randint(0, readImage.shape[1] - N - 1)
        sample = readImage[startPosx:startPosx + N, startPosy:startPosy + N]
        textData[i * numP + j] = [x for sublist in sample for x in sublist]

savetxt('textData', textData)
```

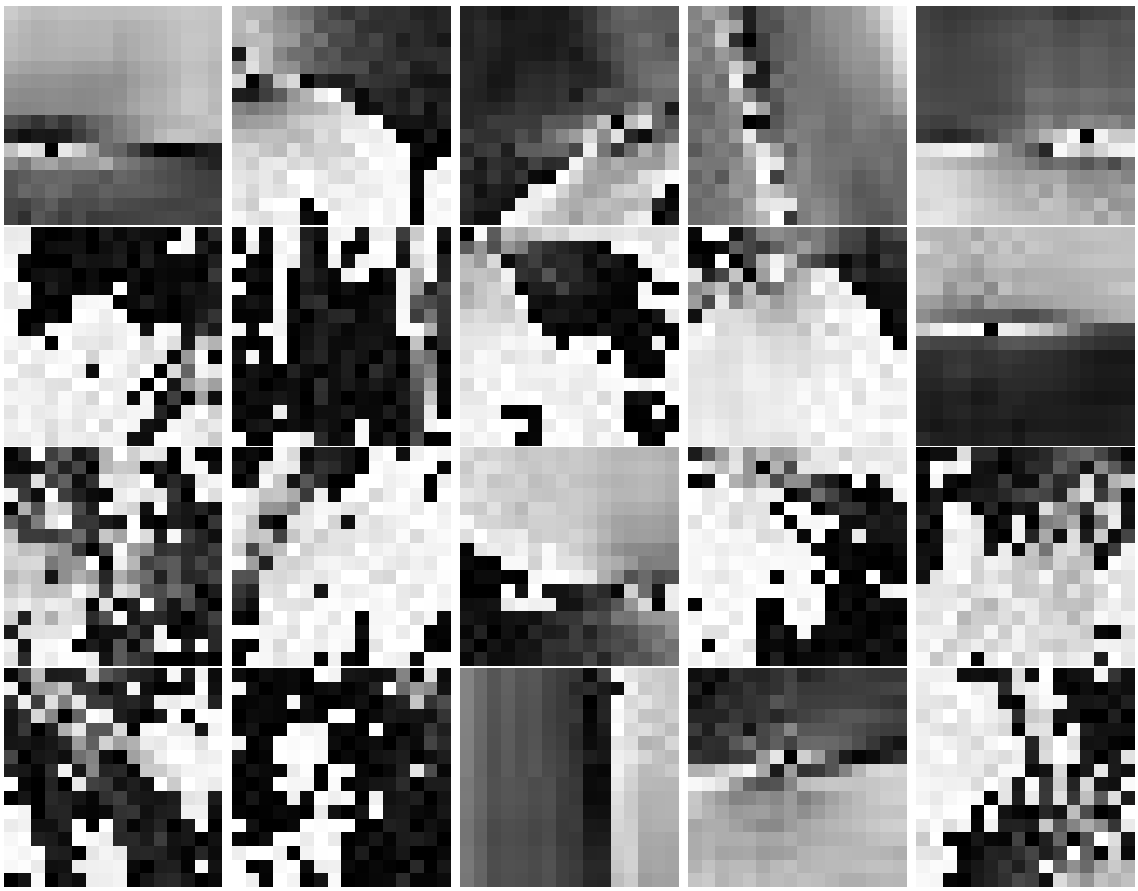
```
from sklearn.decomposition import FastICA
dic = ["building", "nature", "text"]
color = ['b', 'r', 'g']
for i in range(3):
    print "Processing " + dic[i]
    data = loadtxt(dic[i] + "Data")
    ica = FastICA()
    output = ica.fit(data).transform(data)
    features = ica.get_mixing_matrix()
    print features.shape
    savetxt(dic[i] + "feature", features)
```

```
Processing building
(256, 256)
Processing nature
(256, 256)
Processing text
(256, 256)
```

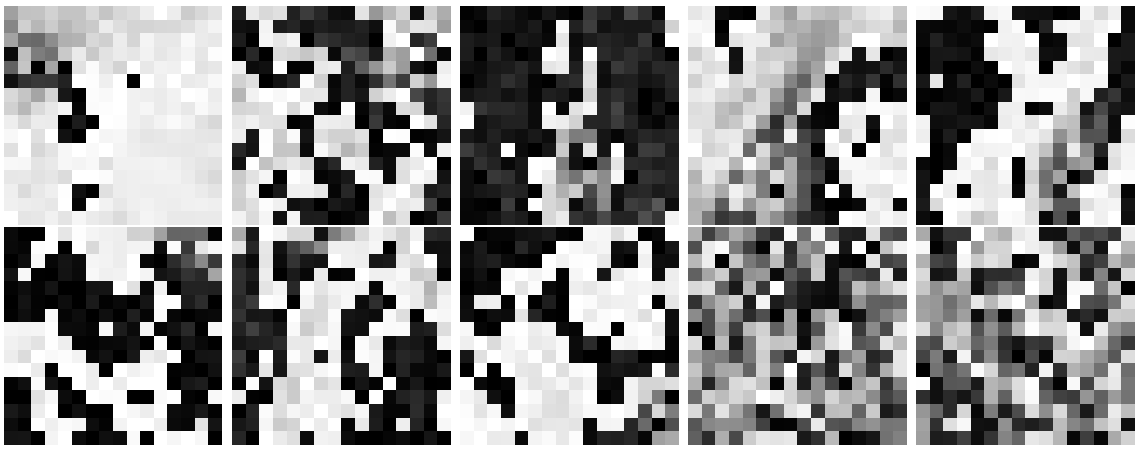
```
from mahotas import imsave
dic = ["building", "nature", "text"]

N = 16
img = array([[0 for i in range(N)] for j in range(N)])
for i in range(3):
    data = loadtxt(dic[i] + "feature").T
    for j in range(20):
        maxV = 0
        for a in range(N):
            for b in range(N):
                if data[j][a * N + b] > maxV:
                    maxV = data[j][a * N + b]
        for a in range(N):
            for b in range(N):
                img[a][b] = data[j][a * N + b] / maxV * 256
```

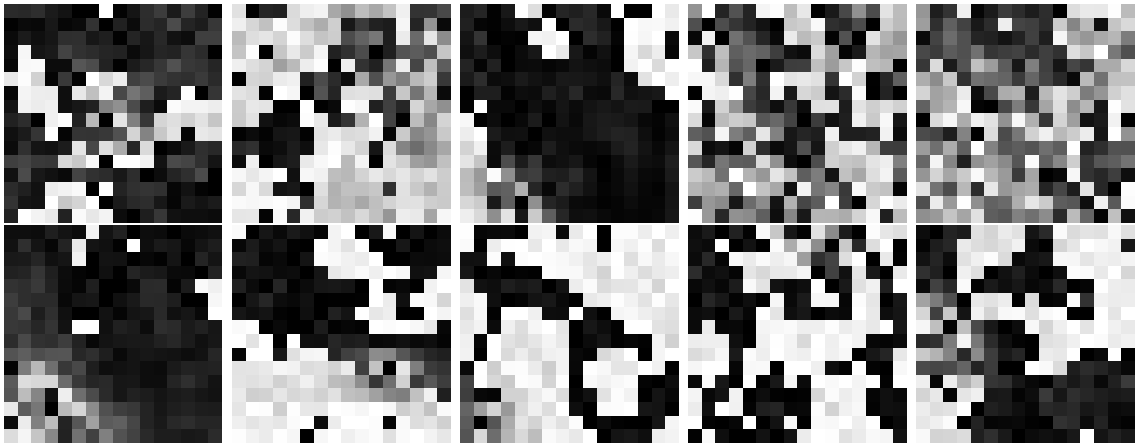
```
        imsave(dic[i] + str(j) + "_patch.jpg", img.astype(uint8))
```

3.1 Building Features:



3.2 Nature Featrues:





3.3 Text Features: