import numpy as np  
import pandas as pd  
import scipy.io  
import matplotlib.pyplot as plt  
from fri import VPW\_FRI,FRI\_estimator  
from matplotlib import gridspec  
import iaf  
import scipy.io  
np.random.seed(32)  
#u = scipy.io.loadmat('Patient\_1\_ecg\_GDN0020.mat')  
#u = u['x']  
#u = u[0]  
u = scipy.io.loadmat('DSOX1204G\_TEM\_pat3.mat')  
u = u['y1']  
#u = scipy.io.loadmat('1.mat')  
#u = u['tfm\_ecg2']  
#u = u[690000:691200]  
  
#u = scipy.io.loadmat('signal\_123\_fri.dat')  
#u = u['rawSignal']  
#u = np.transpose(u)  
#u = u[1500:2000]  
#u = np.transpose(u)  
#u = u-np.mean(u)  
#u = u[1118+509:1118+1018]  
u = u+np.ones(np.shape(u))\*np.min(u)  
#u = u+np.random.random([1600,1])  
#u = scipy.signal.resample(u,20000)  
#u = scipy.io.loadmat('signal\_123\_fri.dat')  
#u = u['rawSignal']  
#u = u.transpose()  
#u = u[1500:2000]  
bias=1.75  
thresh=1.133  
kappa=0.014  
bias=0.78  
thresh=0.99  
kappa=0.018  
#u = scipy.signal.detrend(np.transpose(u),type='linear')  
#u = u.transpose()  
#u = signal  
  
#dt = sampling interval  
#u = scipy.io.loadmat('ecgHW6.mat')  
#u = u['C']  
u[np.isnan(u)] = 0  
#u = u+10\*\*(-50/10)\*np.random.random([2000,1])  
u = scipy.signal.resample(u,200)  
dt = 1/200  
s = iaf.iaf\_encode(u, dt, bias, thresh, np.inf, kappa)  
ts = np.cumsum(s)  
print(len(ts))  
s= pd.DataFrame(ts)  
print(len(s))  
s.to\_csv('signal.csv')  
K = 9  
T = 1.0  
N = 200  
frequencies = np.fft.fftfreq(int(N), T / N)  
omega = 2 \* np.pi \* frequencies / T  
spectrum\_noisy = scipy.io.loadmat('spectrum.mat')  
spectrum\_noisy = spectrum\_noisy['spectrum']  
print(len(spectrum\_noisy))  
  
spectrum\_noisy = spectrum\_noisy.reshape(len(spectrum\_noisy),)  
print(len(spectrum\_noisy))  
print(np.size(spectrum\_noisy))  
print(np.size(u))  
fri\_estimated = FRI\_estimator(K, T, T / N, T / N).estimate\_parameters\_iqml2(u,spectrum\_noisy)  
fri\_estimated = fri\_estimated[0]  
#fri\_estimated = FRI\_estimator(K, T, T / N, T / N).estimate\_parameters(spectrum\_noisy)  
spectrum\_estimated = fri\_estimated.evaluate\_Fourier\_domain(omega)  
gs = gridspec.GridSpec(1, 2)  
classical\_vpw = plt.subplot(gs[0, 0])  
time = np.linspace(0, T, N)  
classical\_vpw.plot(time,u/np.max(u))  
signal\_estimated = np.real(np.fft.ifft(spectrum\_estimated))  
#signal\_estimated = scipy.signal.savgol\_filter(signal\_estimated, 10, 3)  
#max\_loc = np.where(signal\_estimated==signal\_estimated.max())  
#min\_loc = np.where(signal\_estimated==signal\_estimated.min())  
#temp = signal\_estimated[max\_loc]  
#signal\_estimated[max\_loc] = np.abs(signal\_estimated[min\_loc])  
#signal\_estimated[min\_loc] = -signal\_estimated[max\_loc]  
#signal\_estimated[signal\_estimated<-0.5] = 0  
recovered\_vpw = plt.subplot(gs[0, 1])  
#recovered\_vpw.plot(time,scipy.signal.detrend((signal\_estimated-0.2)/np.max(signal\_estimated-0.2))/np.max(scipy.signal.detrend((signal\_estimated-0.2)/np.max(signal\_estimated-0.2))))  
recovered\_vpw.plot(time,(u)/np.max(u))  
recovered\_vpw.plot(time,((signal\_estimated)/max(signal\_estimated)-0.2)/max(((signal\_estimated)/max(signal\_estimated)-0.2)))  
#recovered\_vpw.set\_ylim(-0.5, 1.05 \* np.max(signal\_estimated))  
plt.show()  
SRR = 20\*np.log10(np.linalg.norm(u/np.max(u)-np.mean(u/np.max(u)))/np.linalg.norm(u/np.max(u)-signal\_estimated/np.max(signal\_estimated)))  
RMSE = np.square(np.subtract(signal\_estimated/np.max(signal\_estimated),(u-np.mean(u))/np.max(u-np.mean(u)))).mean()  
print(RMSE)  
save\_file= pd.DataFrame((signal\_estimated)/np.max(signal\_estimated))  
save\_file.to\_csv('pulse33\_100\_hw\_recon.csv')

mat:

clear all

close all

M = readmatrix('C:\Users\97253\PycharmProjects\Pan\_denoiser\_test\signal.csv');

M = M(:,2)';

M = load('C:\Users\97253\PycharmProjects\Pan\_denoiser\_test\DSOX1204G\_TEM\_pat1.mat');

%M = M(:,2)';

M = M.y3';

M = load('C:\Users\97253\PycharmProjects\Pan\_denoiser\_test\DSOX1204G\_TEM\_pat3.mat');

%M = M(:,2)';

M = M.y3';

[x,tau] = findpeaks(M,'MinPeakDistance',40,'MinPeakHeight',1);

M = tau/length(M);

%M = readmatrix('C:\Users\97253\PycharmProjects\Pan\_denoiser\_test\tn.csv')';

%M = readmatrix('scope\_34\_ecg\_best.csv');

%M = readmatrix('scopehw1.csv');

%M = readmatrix('hwscope2.csv');

%M = load('ecg\_HW\_v2.mat');

%M = readmatrix('scope\_29.csv');

%M = M.C;

%x`M = M(3:end,2);

%M = circshift(M,-500);

%M = resample(M,33,100);

%M = resample(M,601,661);

%M = M';

%C = readmatrix('scope\_18.csv');

%M = C(3:end,4)./max(C(3:end,4));

%M2 = C(200:800,3)./max(C(200:800,3));

%M = M';

%M = interp(M,10);

%M = tn;

%[~,locs] = findpeaks(M,'MinPeakDistance',13);

%tn = locs/length(M);

%Ntn = length(tn);

%D = zeros(1,length(M));

%D(locs) = max(M);

tn = M;

%tn = tn+0.05;%optional

%M = resample(M,500,2000);

%M = readmatrix('C:\Users\97253\PycharmProjects\Pan\_denoiser\_test\signal.csv');

%M = M(:,2)';

%tn2 =M(2:end); %tn2-0.016;

%1pullse: b=1.805;d=0.993; kappa=0.01;

%2pulse: b=1.690;d=0.929; kappa=0.010;

%3pulses:b=1.655; d=0.910; kappa=0.01;

%4pulses: b=1.780; d=0.979; kappa=0.010;

%b=0.780; d=0.979; kappa=0.010;

%b= 2.275;d= 1.133;kappa = 0.031;

%b=.65; d=1.133; kappa=0.014;

%b=.52; d=1.133; kappa=0.014;

%b=.55; d=1.133; kappa=0.014;

b=1.75;

d=1.133;

kappa=0.014;

b=0.78;d=0.99;kappa=0.018;

dt = 1/50;

yDel = -b\*diff(tn) + kappa\*d;

K =5;

T = 1;

N =50;

K = 4\*K+2;

w0 = 2\*pi/T;

F = exp(1j\*w0\*tn(2:end)'\*([-K:-1,1:K]))-exp(1j\*w0\*tn(1:end-1)'\*[-K:-1,1:K]);

s = T./(1j\*2\*pi\*([-K:-1,1:K]));

S = diag(s);

%z = cumsum(yDel);

%A = exp(1j\*w0\*tn(2:end)'\*(-K:K));

%s = T./(1j\*2\*pi\*(-K:K)); s(K+1) = 1;

%S = diag(s);

%ytnHat = pinv(A\*S)\*z';

%

F = exp(1j\*w0\*tn(2:end)'\*(-K:K)) - exp(1j\*w0\*tn(1:end-1)'\*(-K:K));

F(:,K+1) = tn(2:end) - tn(1:end-1);

s = T./(1j\*2\*pi\*(-K:K)); s(K+1) = 1;

S = diag(s);

ytnHat = pinv(F\*S)\*yDel';

ytnHat = ytnHat'\*N;

spectrum = transpose(conj(ytnHat(K+1:end)));

spectrum = cadzow(spectrum,inf,K);

save('C:/Users/97253/PycharmProjects/Pan\_denoiser\_test/spectrum.mat','spectrum');

disp("saved!");