```
clear all
addpath ../scatnet-0.2a
addpath_scatnet;
addpath ../minFunc_2012/minFunc
addpath ../minFunc_2012/minFunc/compiled

name = 'bubbles';
Ktrain = 1; % 10;
Kbins = 1; % each bin contains Kbins samples to estimate the beta
Delta = 2;
J = 5;
L = 8;
plotmode=1;
odir='./out/';
tkt = sprintf('pwregress_maxent_bumps2d_dj0_nor_Delta
%d_%s_J%d_L%d_K%d_m%d',Delta,name,J,L,Ktrain,plotmode);

%% get data and estimate spectral
switch name
    case 'tur2a'
        load('../data/ns_randn4_train_N256.mat')
    case 'anisotur2a'
        load('../data/ns_randn4_aniso_train_N256.mat')
    case 'mrw2dd'
        load('../data/demo_mrw2dd_train_N256.mat')
    case 'bubbles'
        load('../data/demo_brDuD111_N256.mat')
end

N = size(imgs,1);
K = Ktrain;
assert(Ktrain<=K)

spImgs = zeros(N,N,K);
for k=1:Ktrain
    spImgs(:,:,k)=(abs(fft2(imgs(:,:,k))).^2)/(N^2);
end
estpsd=mean(spImgs,3);

%% define filters
filtopts = struct();
filtopts.J=J;
filtopts.L=L;
filtopts.full2pi=1;
filtopts.fcenter=0.425; % om in [0,1], unit 2pi
filtopts.gamma1=1;
[filnew,lpal]=bumpsteerableg_wavelet_filter_bank_2d([N N], filtopts);

% compute filters's power spectrum (transfer function)
pwfilters = {};

% nbcov: count (la,la') and (la',la) only once when la!=la'.
nbcov = 0;
% add low pass
```

```
fil = filnew.phi.filter.coefft{1};
filJ = fil / sqrt(sum(sum(spImgs(:,:,1).*(fil.*fil))));
pwfilters{end+1}=filJ.^2;
nbcov = nbcov + 1;

% add high pass
filid = 1;
fftpsi = cell(J,2*L);
for j=1:J
    for q = 1:2*L
        fil=filnew.psi.filter{filid}.coefft{1};
        fftpsi{j,q} = fil / sqrt(sum(sum(spImgs(:,:,1).*(fil.*fil))));
        pwfilters{end+1}=fftpsi{j,q}.^2;
        filid = filid + 1;
        nbcov = nbcov + 1;
    end
end

assert(length(filnew.psi.filter)==filid-1);

% delta_n = Delta
[Omega1,Omega2] = meshgrid(0:2*pi/N:2*pi*(N-1)/N,0:2*pi/N:2*pi*(N-1)/N);
% add low pass
fil = filJ;
for dn1 = -Delta:Delta
    for dn2 = 0:Delta
        if dn1~=0 || dn2~=0
            nbcov = nbcov + 1;
            pwfilters{end+1} = (fil.^2) .* ...
                cos(2^(j-1)*(Omega1*dn1+Omega2*dn2)); % no need for sin since Phi_J is real
        end
    end
end
% add high pass
for j=1:J
    for q = 1:2*L
        fil = fftpsi{j,q};
        for dn1 = -Delta:Delta
            for dn2 = 0:Delta
                if dn1~=0 || dn2~=0
                    nbcov = nbcov + 2;
                    pwfilters{end+1} = (fil.^2) .* ...
                        cos(2^(j-1)*(Omega1*dn1+Omega2*dn2));
                    pwfilters{end+1} = (fil.^2) .* ...
                        sin(2^(j-1)*(Omega1*dn1+Omega2*dn2));
                end
            end
        end
    end
end

Kd=length(pwfilters);
F=zeros(N*N,Kd);
for kid=1:Kd
```

```
    F(:,kid)=pwfilters{kid}(:);
end

estY=zeros(Kd,K);
for kid=1:Kd
    for k=1:K
        estY(kid,k)=sum(sum(spImgs(:,:,k).*pwfilters{kid}));
    end
end
nbins = Ktrain/Kbins;
Ybin=zeros(Kd,nbins);
for kb = 1:nbins
    Ybin(:,kb)=mean(estY(:,(kb-1)*Kbins+1:kb*Kbins),2);
end

%% regress
for kb = 1:nbins
    % compute Y, the constraints
    Y = Ybin(:,kb);

    B=zeros(Kd,1);
    B(1:J*2*L+1) = 1;

    hXrec0=reshape(((F*B).^(-1)),N,N);
    assert(sum(hXrec0(:) > 0)==N*N)
    min_options = struct();
    min_options.Method = 'lbfgs';
    min_options.optTol = 1e-4; % 1e-8;
%     min_options.progTol = 1e-12;
    min_options.Display = '(iter)';
    min_options.MaxIter = 50000;
    min_options.MaxFunEvals = min_options.MaxIter*2;
    [B,loss,exitflag,output] = minFunc(@pwregress_maxent_2d_objfun,B,min_options,F,Y);

    %% plot and save
    bnorm = norm(B)^2 / Kd;
    hX=estpsd;
    % hX=oripsd;
    hXrec=reshape(((F*B).^(-1)),N,N);
    entX=(N*N)/2*(log(2*pi)+1)+sum(log(hX(:)))/2;
    entXrec=(N*N)/2*(log(2*pi)+1)+sum(log(hXrec(:)))/2;
    Yrec = (hXrec(:)'*F)';
    residuerec=max(abs(Yrec'-Y'));
    lossdiffent=0.5*(B'*Y-N*N);
    fprintf('maxent:name=%s,J= %d, loss=%.2e,residuerec=%g,lossdiffent=%g,bnorm=%g\n',...
        name,J,loss,residuerec,lossdiffent,bnorm);
    fprintf('entX=%g,entXrec=%g\n',entX,entXrec);

    figure(44);
    if plotmode == 2
        % inrag=[min(hX(:)),max(hX(:))];
        inrag=[min(hXrec(:)),max(hXrec(:))];
        subplot(131)
        imagesc(fftshift(hX),inrag); colorbar; axis square
        title('Empirical: P(\omega)','FontSize',20)
        % title('Groundtruth: P(\omega)','FontSize',20)
        subplot(132)
        imagesc(fftshift(hXrec),inrag); colorbar; axis square
        title('Macrocanonical: hat P(\omega)','FontSize',20)
        subplot(133)
%         subplot(132)
%         imagesc(fftshift(hX-hXrec)); colorbar; axis square
%         title('bias: P(\omega)- hat P(\omega)','FontSize',20)
%         subplot(133)
        om2=linspace(-pi,pi,N+1);
        plot(om2(1:end-1),fftshift(hX(1,:)-hXrec(1,:)));
        title('bias: P(0,\omega_2)- hat P(0,\omega_2)','FontSize',20)
        xlabel('\omega_2 \in [-\pi,\pi]','FontSize',20)
        axis tight
    elseif plotmode==1
        loghX=log10(hX);
        loghXrec = log10(hXrec);
        inrag=[min(loghXrec(:)),max(loghXrec(:))];
%         inrag=[min(loghX(:)),max(loghX(:))];
        subplot(131)
        imagesc(fftshift(loghX),inrag); colorbar; axis square
        title('Empirical: log10 P(\omega)','FontSize',20)
        subplot(132)
        imagesc(fftshift(loghXrec),inrag); colorbar; axis square
        title('Macrocanonical: log10 hat P(\omega)','FontSize',20)
        subplot(133)
        % imagesc(fftshift(hX-hXrec)); colorbar; axis square
        % title('bias: P(\omega)- hat P(\omega)','FontSize',20)
        plot(Y)
        hold on
        plot(Yrec,'o')
        hold off
        legend({'Y','Yrec'})
%         om2=linspace(-pi,pi,N+1);
%         plot(om2(1:end-1),fftshift(hX(1,:)-hXrec(1,:)));
%         title('bias: P(0,\omega_2)- hat P(0,\omega_2)','FontSize',20)
%         xlabel('\omega_2 \in [-\pi,\pi]','FontSize',20)
        axis tight
    else
        assert(false)
    end

    set(gcf,'Position',[0 0 1600 400])
    savefig(gcf,sprintf('%s/%s_J%d_bias_kb%d.fig',odir,tkt,J,kb))
    saveas(gcf,sprintf('%s/%s_J%d_bias_kb%d.eps',odir,tkt,J,kb),'eps')

    save(sprintf('%s/%s_kb%d.mat',odir,tkt,kb),'B','Y','Yrec','hX','hXrec','loss',...
        'entX','entXrec','residuerec','lossdiffent','bnorm')
end
save(sprintf('%s/%s.mat',odir,tkt),'estpsd','spImgs','filtopts')

```