clear;

% main function for HW04

% all function implementations are added in .zip file

% full code for all functions is added in a .doc file

N\_signal = 64;

[x,y\_,real\_psd\_ma,real\_psd\_ar] = generateSig(N\_signal);

estimatePSD(N\_signal,x,real\_psd\_ma);

myWelch(x,N\_signal,16,8,real\_psd\_ma);

ParametricEstBeta(N\_signal,x,real\_psd\_ma, "MA");

estimatePSD(N\_signal,y\_,real\_psd\_ar);

myWelch(y\_,N\_signal,16,8,real\_psd\_ar);

ParametricEstBeta(N\_signal,y\_,real\_psd\_ar, "AR");

N\_signal = 1024;

[x,y\_,real\_psd\_ma,real\_psd\_ar] = generateSig(N\_signal);

estimatePSD(N\_signal,x,real\_psd\_ma);

myWelch(x,N\_signal,16,8,real\_psd\_ma);

ParametricEstBeta(N\_signal,x,real\_psd\_ma, "MA");

estimatePSD(N\_signal,y\_,real\_psd\_ar);

myWelch(y\_,N\_signal,16,8,real\_psd\_ar);

ParametricEstBeta(N\_signal,y\_,real\_psd\_ar, "MA");

function parametricEstBeta(N\_signal,x,real\_psd,modelType)

N = N\_signal\*4; portion = N/2; wgrid = 2\*pi\*(0:N-1)/N;

r\_x = xcorr(x,"biased");

%first order AR

[b1,a1] = parametricARModel(r\_x,1);

den1 = abs(fft(a1,N)).^2; num1 = b1;

est\_psd1 = num1./den1;

figure(); plot(wgrid(1:portion),est\_psd1(1:portion),wgrid(1:portion),real\_psd(1:portion), "black--")

grid on; axis tight; title("Parametric Estimation of P\_x(w) for AR(1) Model", ["Real psd is for " + modelType + " Model","N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); xlabel("w [rad/sec]"); ylabel("P\_x(w)");

legend("est PSD", "real PSD");

%second order AR

[b2,a2] = parametricARModel(r\_x,2);

den2 = abs(fft(a2,N)).^2; num2 = b2;

est\_psd2 = num2./den2;

figure(); plot(wgrid(1:portion),est\_psd2(1:portion),wgrid(1:portion),real\_psd(1:portion), "black--")

grid on; axis tight; title("Parametric Estimation of P\_x(w) for AR(2) Model",["Real psd is for " + modelType + " Model","N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); xlabel("w [rad/sec]"); ylabel("P\_x(w)");

legend("est PSD", "real PSD");

% first order ma

func = @(r,n,k) sum(r(N\_signal -k:N\_signal +k)'.\*exp(-j\*((2 \* pi)\*[-k:k]'\*n)/N),1);

est\_psd\_ma1 = abs(func(r\_x,[0:N],1));

figure(); plot(wgrid(1:portion),est\_psd\_ma1(1:portion),wgrid(1:portion),real\_psd(1:portion),"black--");

grid on; axis tight; title("Parametric Estimation of P\_x(w) for MA(1) Model",["Real psd is for " + modelType + " Model", "N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); xlabel("w [rad/sec]"); ylabel("P\_x(w)");

legend("est PSD", "real PSD");

% second order ma

est\_psd\_ma2 = abs(func(r\_x,[0:N],2));

figure(); plot(wgrid(1:portion),est\_psd\_ma2(1:portion),wgrid(1:portion),real\_psd(1:portion),"black--");

grid on; axis tight; title("Parametric Estimation of P\_x(w) for MA(2) Model",["Real psd is for " + modelType + " Model" , "N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); xlabel("w [rad/sec]"); ylabel("P\_x(w)");

legend("est PSD", "real PSD");

end

function [x,y\_,real\_psd\_ma,real\_psd\_ar] = generateSig(N\_signal)

fs = 44100;

sigma = 1;

mu = 0;

N\_ma = N\_signal; N\_ar = 15\*N\_signal; N = N\_signal\*4;

w\_ma = sigma\*randn(1,N\_ma) + mu;

w\_ar = sigma\*randn(1,N\_ar) + mu;

b\_ma = [1, 0.3, 0.8]; b\_ar = 1;

a\_ma = 1; a\_ar = [1, 0.2, 0.6];

x = filter(b\_ma,a\_ma,w\_ma);

y = filter(b\_ar,a\_ar,w\_ar); y\_ = y(1,N\_ar-(N\_signal-1):N\_ar);

[H\_ma ~] = freqz(b\_ma,a\_ma,N/2,fs);

[H\_ar ~] = freqz(b\_ar,a\_ar,N/2,fs);

real\_psd\_ma = sigma\*(abs(H\_ma)).^2; real\_psd\_ma = real\_psd\_ma';

real\_psd\_ar = sigma\*(abs(H\_ar)).^2; real\_psd\_ar = real\_psd\_ar';

end

function estimatePSD(N\_signal,x,real\_psd)

N = N\_signal\*4; portion = N/2;

% first method: PSD = (1/N)\*|X(jw)|^2

X1 = fft(x,N); psd2 = (1/N\_signal)\*abs(X1).^2;

% second method: PSD = fourier transform of the autocorrelation (with

% xcorr)

r\_hat = xcorr(x,"normalized"); psd3 = abs(fft(r\_hat,N));

% third method: PSD = fourier transform of the autocorrelation (manually calculated):

r\_man = zeros(1,N\_signal);

for k = [1:N\_signal]

for n = [1:N\_signal-k]

r\_man(k) = r\_man(k) + x(n+k)\*conj(x(n));

end

r\_man(k) = (1/r\_man(1))\*r\_man(k);

end

r\_man\_ = [r\_man(1,[N\_signal:-1:2]), r\_man];

psd1 = (1/N\_signal)\*abs(fft(r\_man\_,N));

wgrid = 2\*pi\*(0:N-1)/N;

figure(); plot(wgrid(1:portion),psd1(1:portion),wgrid(1:portion),real\_psd,"black--"); % for loop computation

title("P\_p\_e\_r\_i\_o\_d\_o\_g\_r\_a\_m = FFT on Manualy Computed r(k)", ["N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); ylabel("estimated P\_1\_,\_x(f)"); axis tight; grid on;

figure(); plot(wgrid(1:portion),psd2(1:portion),wgrid(1:portion),real\_psd,"black--"); % fft squared

title("P\_p\_e\_r\_i\_o\_d\_o\_g\_r\_a\_m = (1/N)|X(f)|^2",["N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); ylabel("estimated P\_2\_,\_x(f)"); axis tight; grid on;

figure(); plot(wgrid(1:portion),psd3(1:portion),wgrid(1:portion),real\_psd,"black--"); % xcorr computation

title("P\_p\_e\_r\_i\_o\_d\_o\_g\_r\_a\_m = FFT on xcorr(x,x) output",["N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); ylabel("estimated P\_3\_,\_x(f)"); axis tight; grid on;

end

function myWelch(x,N\_signal,L,D,real\_psd)

fs = 44100; N = 4\*N\_signal; wgrid = 2\*pi\*(0:N-1)/N; portion = N/2; Nfft = 4\*N\_signal;

K = (N\_signal-L)/D + 1;

xMat = [];

for i = [0:K-1]

if(i\*D + L <= N\_signal)

xMat = [xMat; x(1,i\*D + 1:i\*D + L)];

else

xMat = [xMat; x(1,i\*D + 1:N\_signal)];

end

end % splitting of signal

xMat\_fft = (1/L)\*abs(fft(xMat,Nfft,2)).^2; % modified periodogram for each row

welchPsd = mean(xMat\_fft,1); % welch -> avg of all periodograms

figure(); plot(wgrid(1:portion),welchPsd(1:portion),wgrid(1:portion),real\_psd,"black--");

title("Welch Estimation of P\_x(e^j^w)",["N\_s\_i\_g\_n\_a\_l = " + num2str(N\_signal)]); xlabel("w [rad/sec]");

ylabel("P\_w\_e\_l\_c\_h\_,\_x(e^j^w)"); grid on; axis tight;

end

function [b,a] = parametricARModel(r,p)

N = length(r) + 1;

B = [];

for i = [1:p]

B = [B; [r(N/2 - (i-1):N/2 + p-1 - (i-1))]];

end

v = -1\*[r(N/2+1:N/2+p)]';

B\_ = inv(B);

a = B\_\*v; a = [1, a'];

b = sum(a.\*[r(N/2:N/2+p)]);

end