%%%%%%%%%%%%%%%%%

%debruitage d'un signal par filtre de wiener

%23 06 2021

%%%%%ùùù%%%%

clear all;

close all;

clc;

%%%%%%%%%param de simulation

f0=1e3;%[Hz] frequence de signal util

fe=1e5;%[Hz] frequence d'echantillonnage >=2 \* fmax

T=100/f0;

t=0:1/fe:T;%[1/fe]=temps d'echantillonnage

SNR=5;%[dB]

snr=10^(SNR/10);%[valeur lineaire]

N=20;%nbre du coefficient du filtre de wiener h=[h(0),h(1),....h(9)]

sk=sin(2\*pi\*f0\*t(1:end-1));%signal utile

P\_sk=var(sk);%puissance du signal s(k)[var pour variance]

sigma\_n=sqrt(P\_sk/snr);%ecart type du bruit

nk = sigma\_n \* randn(1,length(sk));%bruit de variance sigma\_n^2

rk = sk + nk; %observation bruitée r(k)

%%%%% estimation de R\_r et r\_sr

R\_r=zeros(N,N);

R\_s=zeros(N,N);

R\_n=zeros(N,N);

r\_sr=zeros(N,1);

for iteration=1:length(rk)-N+1

R\_r = R\_r + rk(iteration:iteration+N-1)' \* rk(iteration:iteration+N-1);

R\_s = R\_s + sk(iteration:iteration+N-1)' \* sk(iteration:iteration+N-1);

R\_n = R\_n + nk(iteration:iteration+N-1)' \* nk(iteration:iteration+N-1);

r\_sr = r\_sr + sk(iteration) \* rk(iteration:iteration+N-1)';

end

R\_r\_est = R\_r / length(rk)-N+1; %[length(rk)-N+1] c'est M c'est le nbre de segments ou de decalage possible (combien tu as utilisé de vecteurs)

r\_sr\_est = r\_sr / length(rk)-N+1;

R\_s\_est = R\_s / length(rk)-N+1;

R\_n\_est = R\_n / length(rk)-N+1;

h\_opt = inv(R\_r\_est) \* r\_sr\_est;

d\_hat = filter(h\_opt,1,rk);% c'est un filtre RIF

%%%%%%%%% affichage

figure(1)

plot(t(1:end-1),sk,'-k','LineWidth',2)

hold on

plot(t(1:end-1),rk,'-r','LineWidth',2)

hold on

plot(t(1:end-1),d\_hat,'-b','LineWidth',2)

xlabel('temps [sec]'), ylabel('Amplitude')

legend('s(k)','r(k)','d\_hat(k)')% bleu c'est signal filtré noir ideal et rouge notre signal donné

axis([0 t(end) -2 2]),grid

%xlabel('temps [sec]'), ylabel('Amplitude')

%legend('d\_{hat}(k)')

%axis([0 t(end) -2 2]),grid

ax = gca;

ax.FontSize = 16;

ax.TitleFontWeight = 'bold';

%%%%%%%%%%%%%%%

SNR\_in = SNR

SNR\_out = 10 \* log10(var(filter(h\_opt,1,sk))/var(filter(h\_opt,1,nk)))

%SNR\_in=5

% quand N=20 SNR\_out=13,\*\*\*\* trop grand

% quand N=2 SNR\_out=7.9041 mieux

%si T diminue alors M diminue alors SNR\_out dimunue donc c’est mieux

%plus SNRout se rapproche de SNRin plus c’est mieuuuuuuuuuuux, pour cela il faut diminuer T et N <- nrmlmt