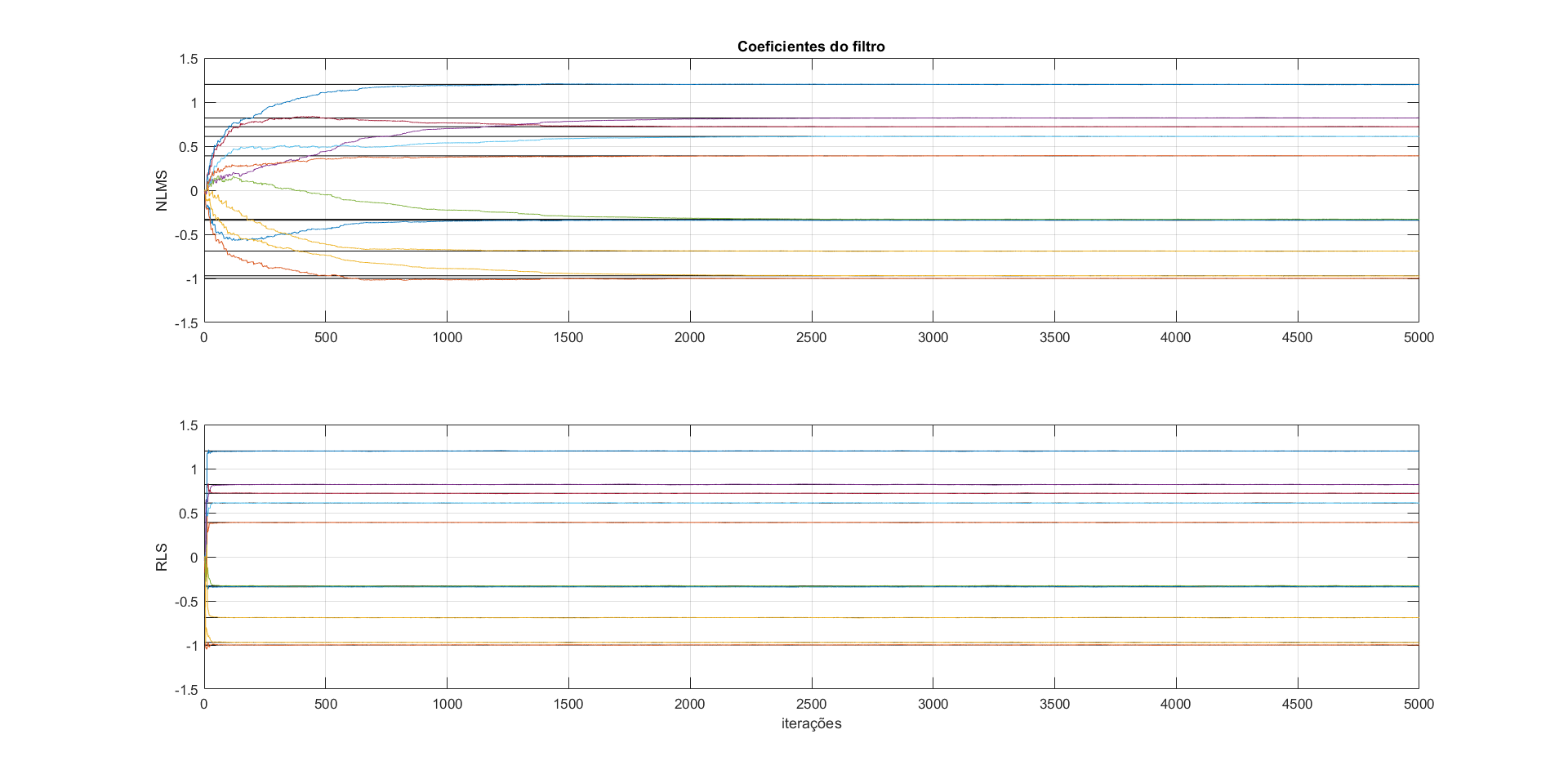
Os *Scripts* de MATLAB usados para alcançar os resultados mostrados abaixo estão nos Anexos.

**a)**

*Figura 1 –Gráficos das médias de conjunto dos coeficientes adaptados dos algoritmos NLMS e RLS ao longo das iterações.*

*As curvas mostradas acima explicitam o quão mais rápido os coeficientes adaptados do algoritmo RLS convergem para os coeficientes ótimos mais rápido. Enquanto o NLMS converge somente após aproximadamente 1600 iterações o RLS convergem em torno de 20 iterações, respeitando o valor teórico aproximado 2M iterações para convergência.*

**b)**

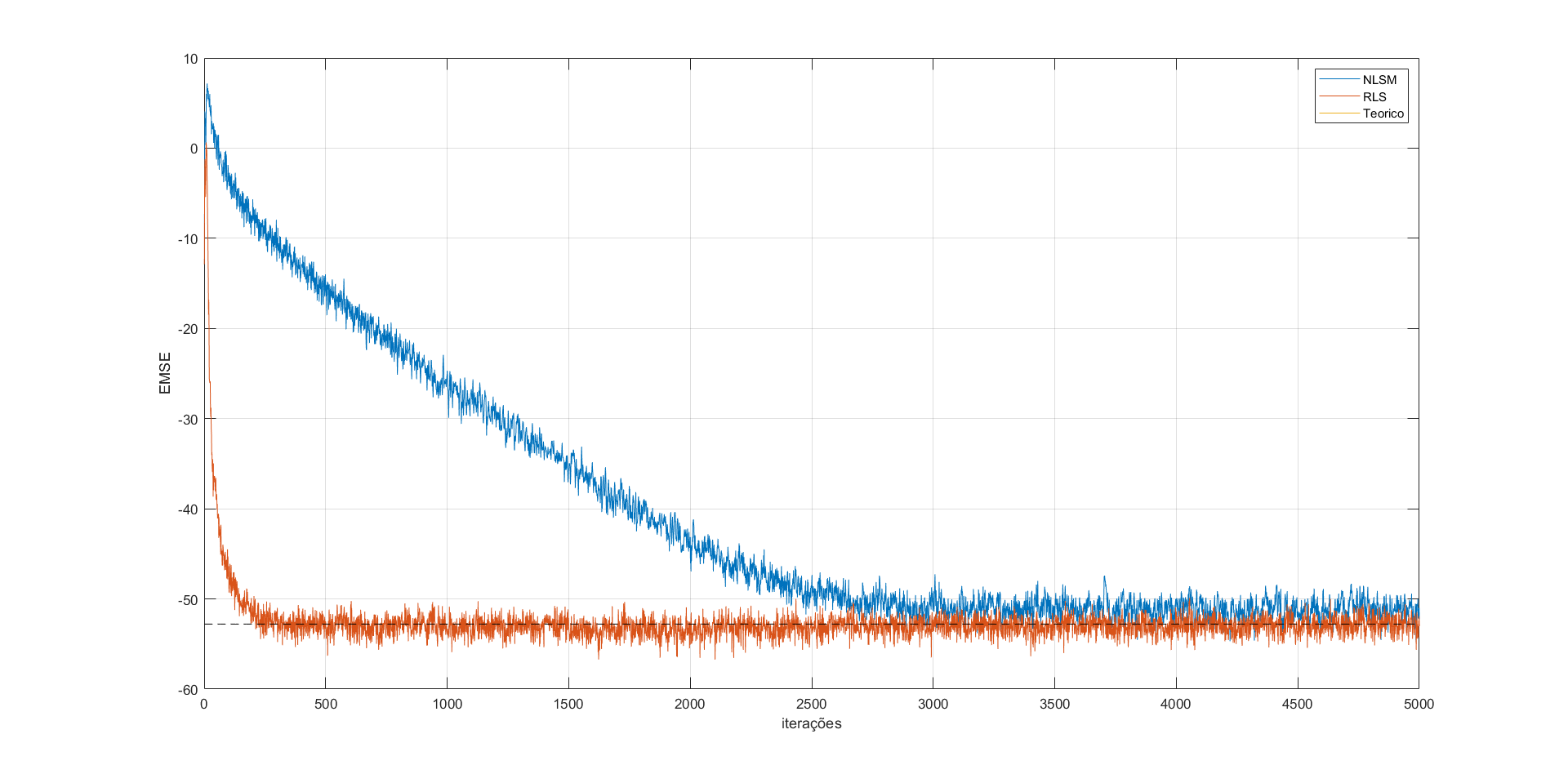
Sendo:

Para que os dois algoritmos convirjam para o mesmo patamar de EMSE, temos:

Assim:

Para e , chegamos:

**c)**



*Figura 2 – Curvas de EMSE ao longo das iterações para os algoritmos NLMS e RLS comparados ao valor teórico em regime.*

É percebido uma diferença aceitável de aproximadamente 1 dB em regime entre os EMSEs do NLMS e do RLS. Isso decorre da diferença entre as aproximações (hipóteses e outras considerações) usadas para obter os modelos teóricos dos dois algoritmos resultando, por fim, numa diferença no fator de esquecimento λ.

**Anexo A – Script de MATLAB para solução do exercício**

%PTC 5890 - 2019

%Prof: Magno T. M. Silva e Maria D. Miranda

%Aluno: Stéfano Albino Vilela Rezende (Ouvinte)

%P4 - Parte Computacional

clear all;

close all;

clc;

M=10;

N = 5000;

rlzc = 50;

H = [-0.34 -1 -0.97 0.82 -0.33 0.61 0.72 1.2 0.39 -0.69];

b = 0.8;

sigmav2=10^-4;

mu\_til = 0.1;

delta\_NLMS = 10^-5;

delta\_RLS = 0.1;

lambda = 0.99;

WNmean = zeros(N+1,M);

WRmean = zeros(N+1,M);

EMSE\_NLMS\_simul = zeros(N,rlzc);

EMSE\_RLS\_simul = zeros(N,rlzc);

EMS\_NLMS\_simul = zeros(N,1);

EMS\_RLS\_simul = zeros(N,1);

h=waitbar(0,'Computando...');

for i=1:rlzc

x = randn(1,N);

u = filter (sqrt(1-b^2),[1 -b],x);

v=sqrt(sigmav2)\*randn(1,N);

d=filter(H,1,u)+v;

[yN,eN,eaN,WN] = NLMS (u,d,M,mu\_til,N,delta\_NLMS,H);

[yR,eR,eaR,WR] = RLS (u,d,M,N,delta\_RLS,lambda,H);

WNmean = (WN+WNmean)/2;

WRmean = (WR+WRmean)/2;

% EQM

EMS\_NLMS\_simul = EMS\_NLMS\_simul+eN.^2;

EMS\_RLS\_simul = EMS\_RLS\_simul+eR.^2;

% EQME

EMSE\_NLMS\_simul= EMSE\_NLMS\_simul+eaN(:).^2;

EMSE\_RLS\_simul = EMSE\_RLS\_simul +eaR(:).^2;

waitbar(i/rlzc);

end

close(h);

%Média EQME

EMSE\_NLMS\_simul\_mean = EMSE\_NLMS\_simul/rlzc;

EMSE\_RLS\_simul\_mean = EMSE\_RLS\_simul/rlzc;

%Média EQM

EMS\_NLMS\_simul\_mean = EMS\_NLMS\_simul/rlzc;

EMS\_RLS\_simul\_mean = EMS\_RLS\_simul/rlzc;

%Valor Teórioco do EQME

EMSE\_NLMS\_teor = mu\_til\*sigmav2/(2-mu\_til);

EMSE\_RLS\_teor = (1-lambda)\*M\*sigmav2/2;

%a)

figure(1)

subplot(211)

plot(kron(ones(N+1,1),H),'k');

hold on;

plot(WN);

xlim([0,N]);

grid;

ylabel('NLMS')

title('Coeficientes do filtro')

subplot(212);

plot(kron(ones(N+1,1),H),'k');

hold on;

plot(WR);

xlim([0,N]);

ylabel('RLS')

xlabel('iterações')

grid;

%c)

figure(2);

plot(10\*log10(abs(EMSE\_NLMS\_simul\_mean)));

hold on;

plot(10\*log10(abs(EMSE\_RLS\_simul\_mean)));

plot(10\*log10(EMSE\_NLMS\_teor)\*ones(N,1),'--k');

hold off;

grid; legend('NLSM','RLS','Teorico');ylabel('EMSE');xlabel('iterações')

figure(3);

plot(10\*log10(abs(EMS\_NLMS\_simul\_mean)));

hold on;

plot(10\*log10(abs(EMS\_RLS\_simul\_mean)));

% plot(10\*log10(EMSE\_NLMS\_teor)\*ones(N,1),'--k');

hold off;

grid; legend('NLSM','RLS');ylabel('EMS (dB)');xlabel('iterações')

**Anexo B – Função de MATLAB para o algoritmo NLMS**

function [y,e,W] = NLMS (u,d,M,mu,N,delta)

uM = zeros(M,1)

y = zeros(N,1)

e = zeros(N,1)

W = zeros(N+1,M)

for n=1:N

uM=[u(n);uM(1:M-1)];

y(n)=W(n,:)\*uM;

e(n)= d(n)-y(n);

W(n+1,:) = W(n,:)+mu/(norm(uM)^2+delta)\*e(n)\*uM';

end

end

**Anexo C – Função de MATLAB para o algoritmo RLS**

function [y,e,ea,W] = RLS (u,d,M,N,delta,lambda,wo)

uM = zeros(M,1);

y = zeros(N,1);

e = zeros(N,1);

W = zeros(N+1,M);

R = delta^-1\*eye(M,M);

for n=1:N

uM=[u(n);uM(1:M-1)];

ea(n)=(wo-W(n,:))\*uM;

k\_til=R\*uM;

gama=1/(lambda+uM'\*R\*uM);

k = gama\*k\_til;

%R = lambda^-1\*R-lambda^-1\*k\*k\_til' %VERSÃO 1

R=lambda^-1\*R-(k\_til\*k\_til')\*lambda^-1\*gama; %VERSÃO 2

e(n) = d(n)-uM'\*W(n,:)';

W(n+1,:) = W(n,:)+k'\*e(n);

end

end