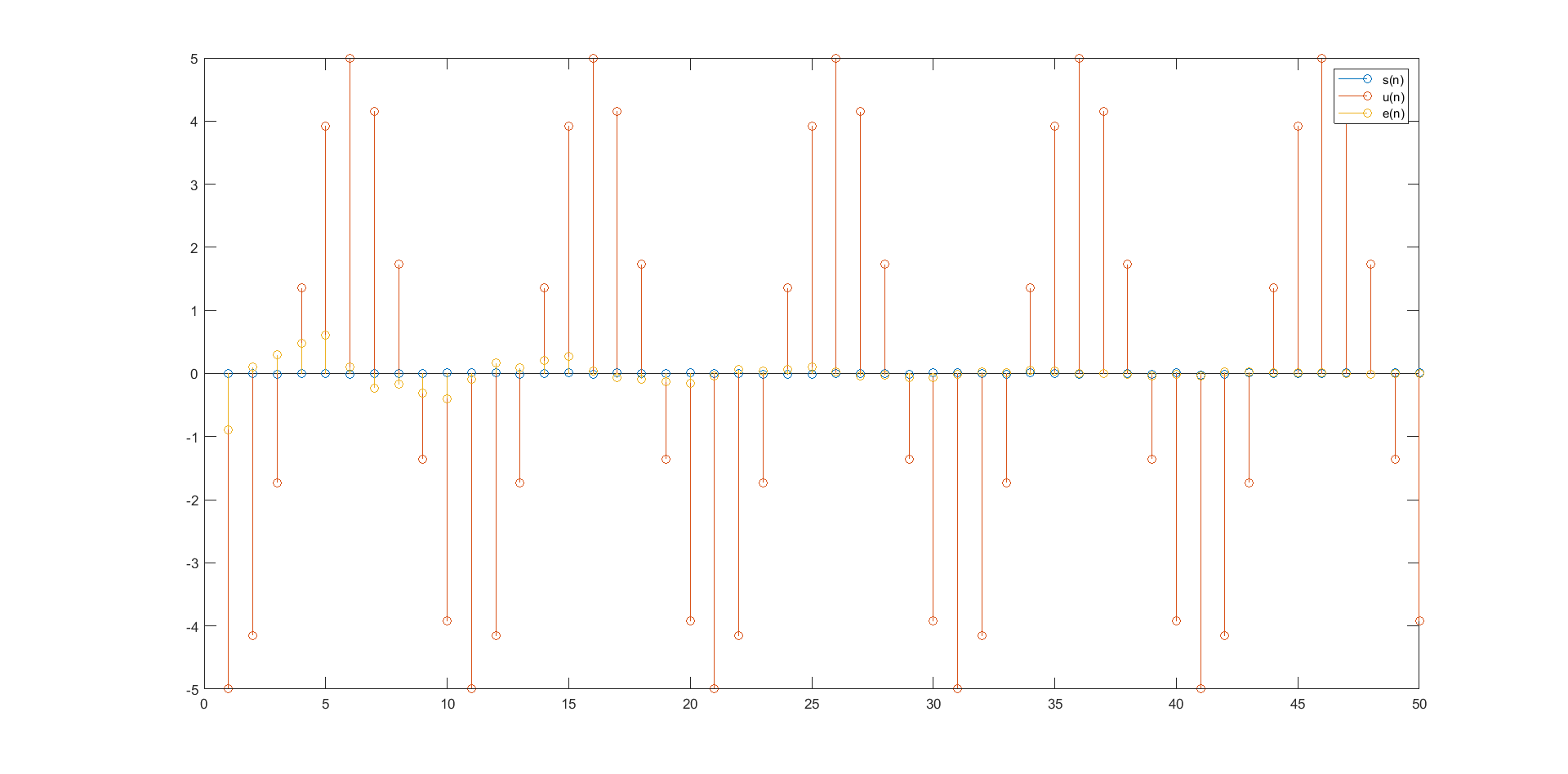
Os *Scripts* de MATLAB usados para alcançar os resultados mostrados abaixo estão nos Anexos.

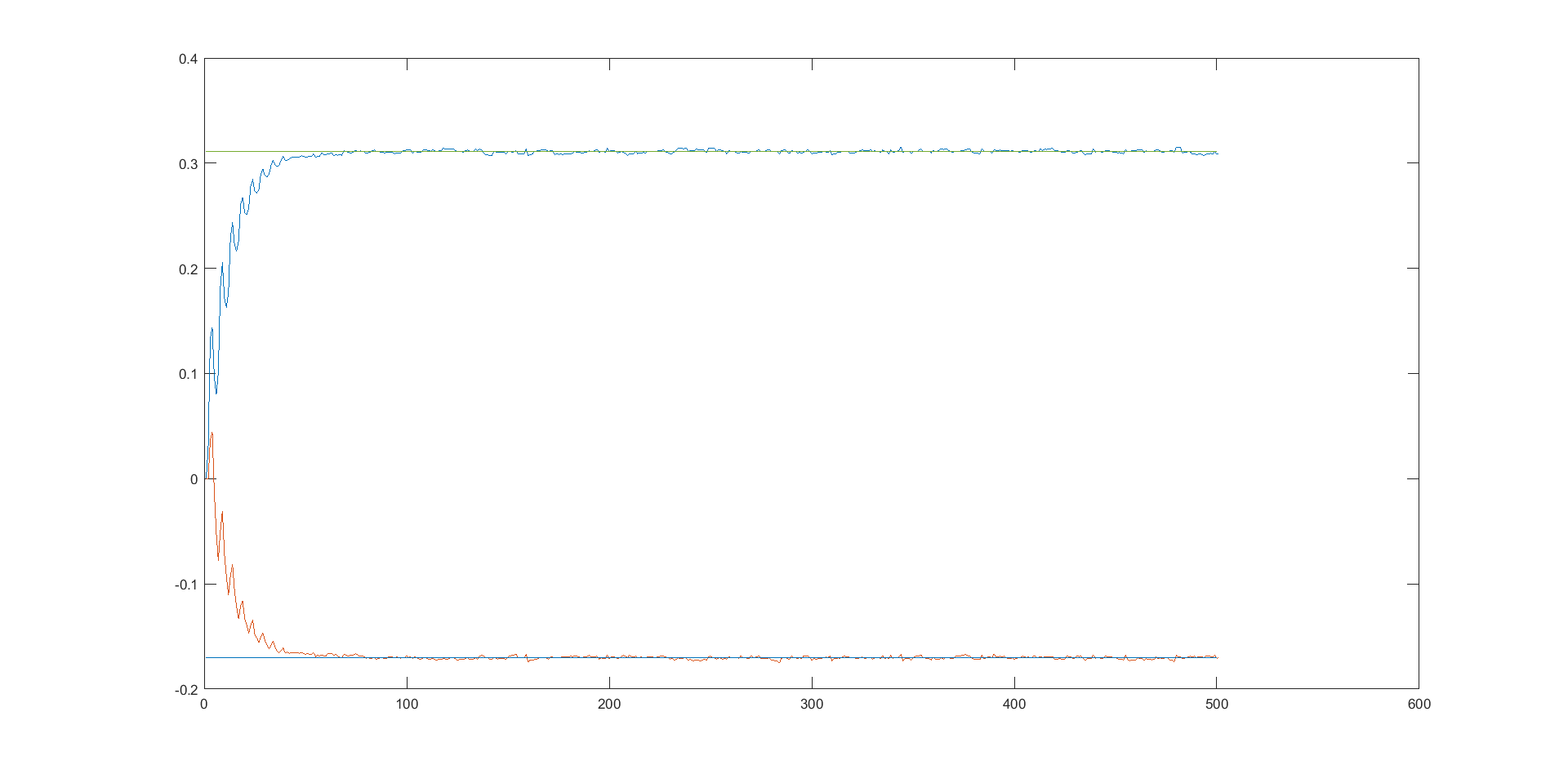
**b)**Sendo os limites do passo do algoritmo *Steepest Descent* definidos por:

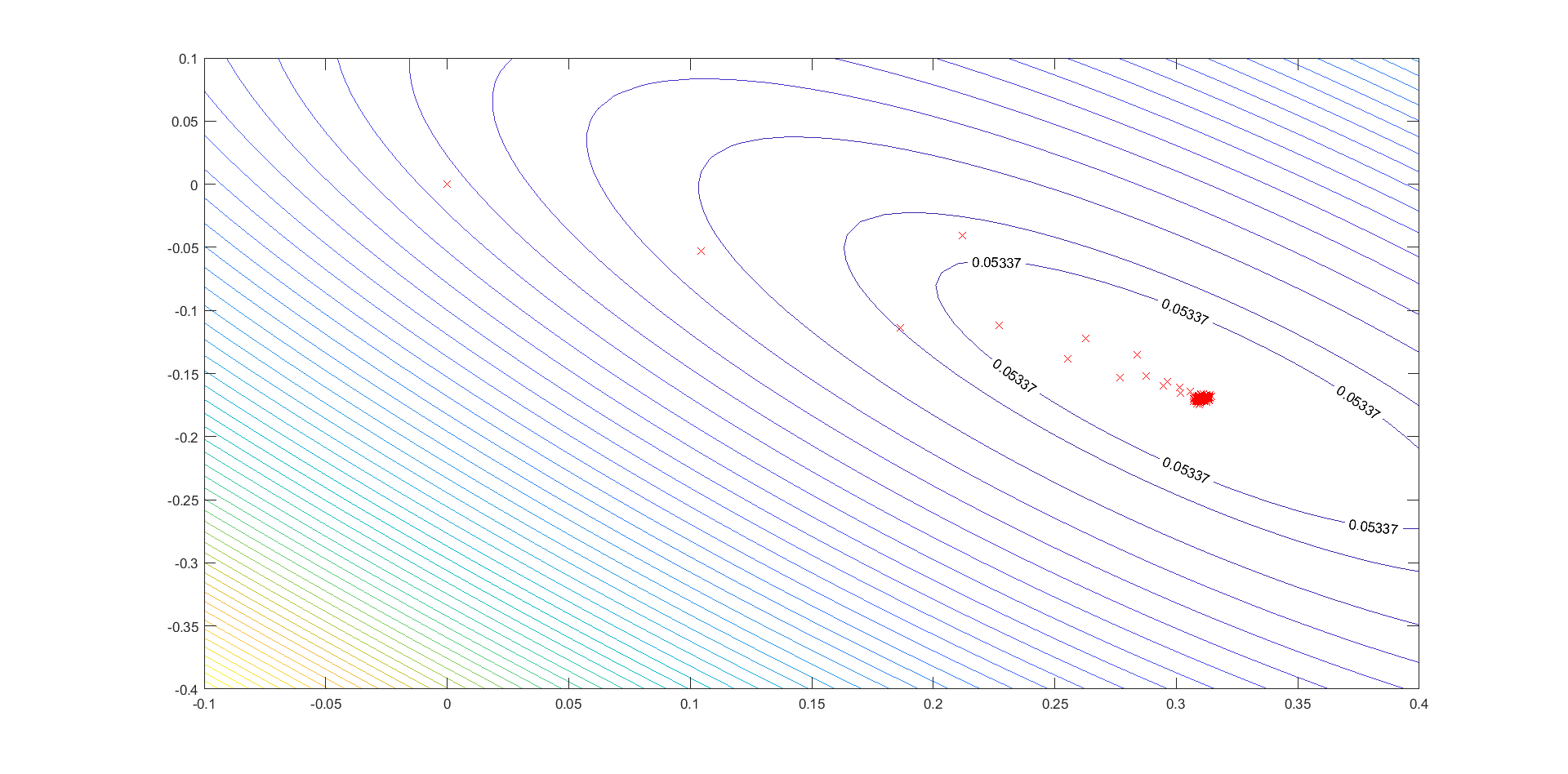
Temos:

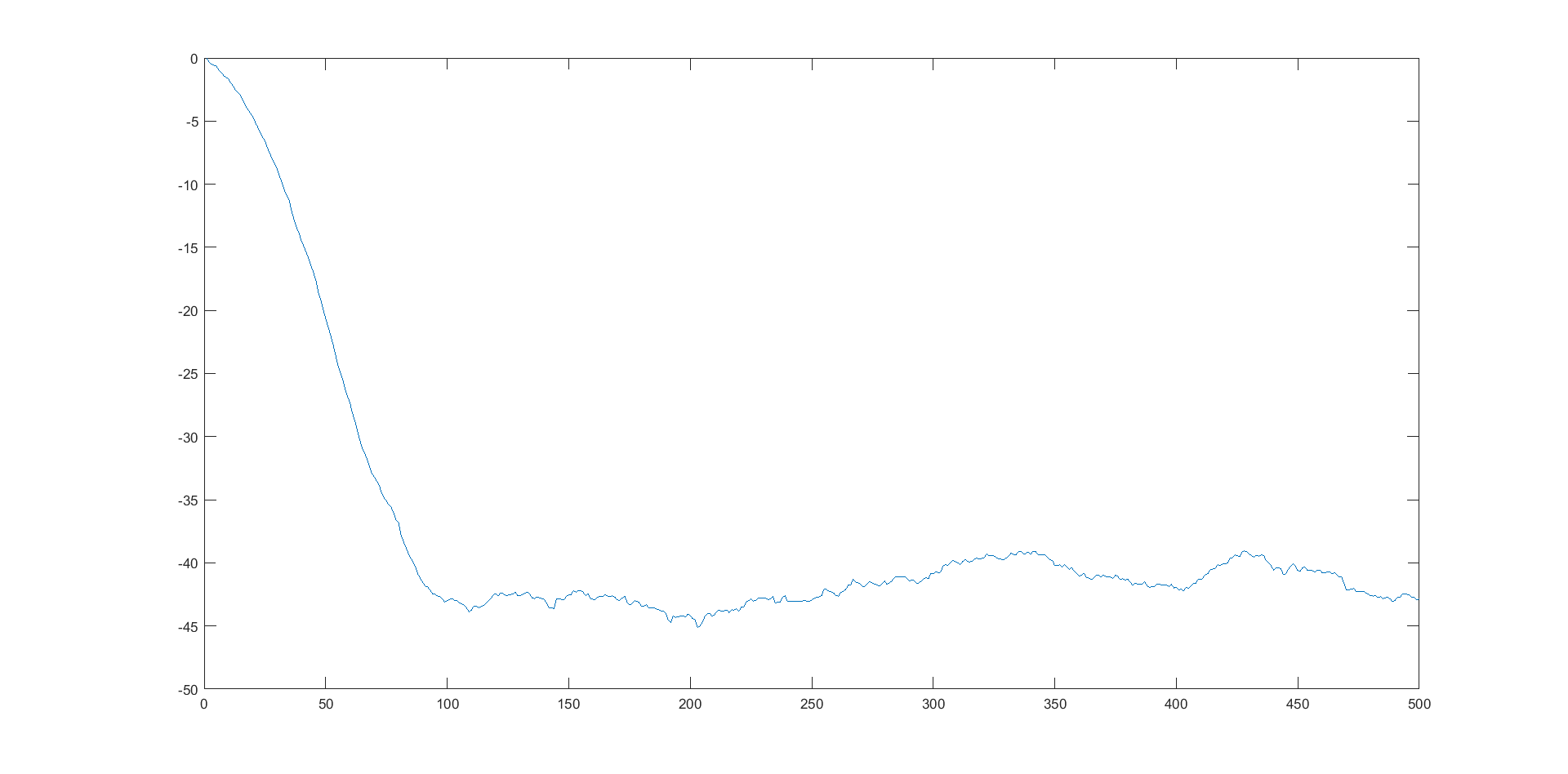
Portanto:

**c)**

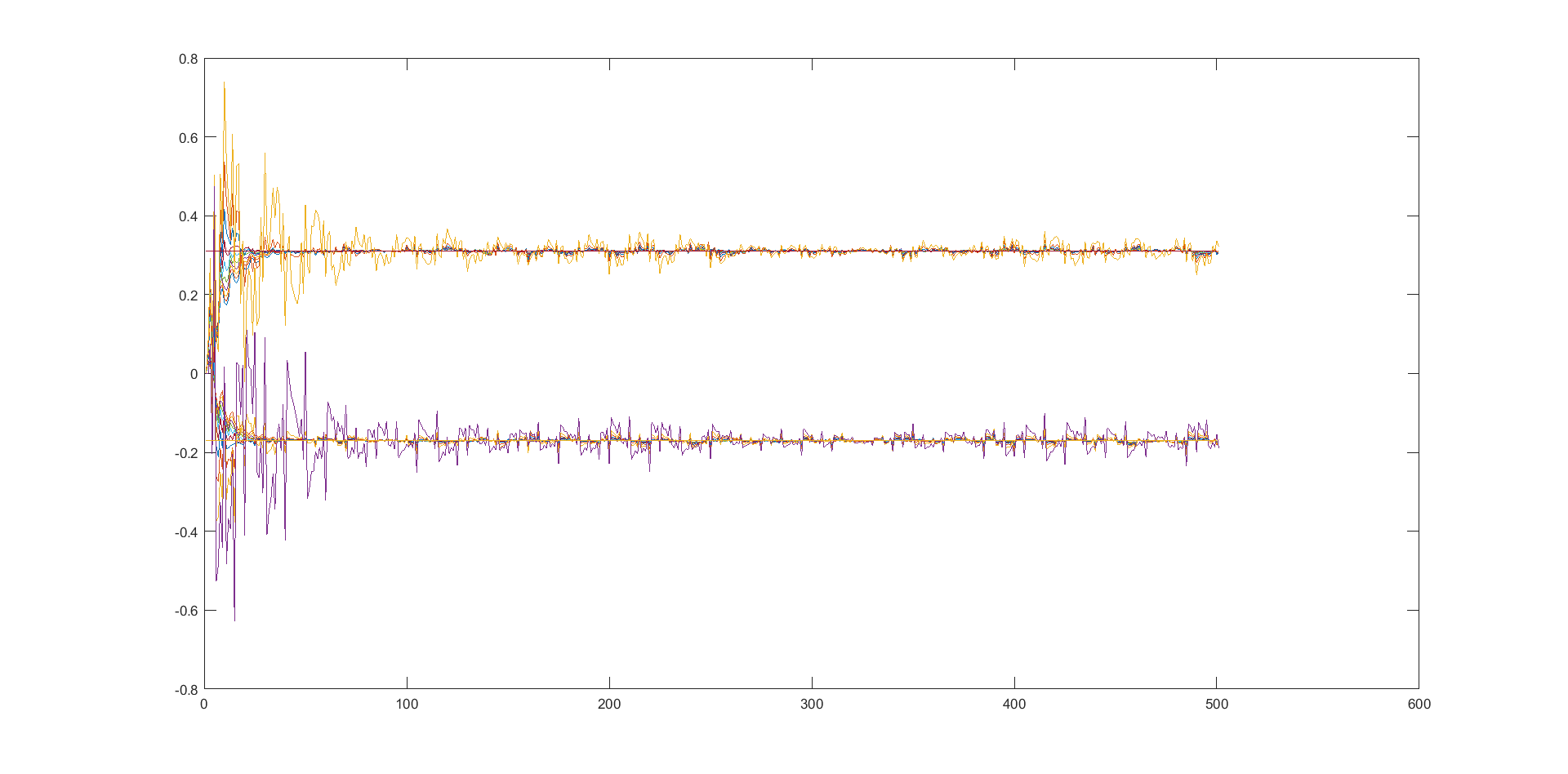
*Figura 1 – Gráfico contendo os sinals de entrada u(n), de erro e(n) e s(n)*

*Figura 2 – Gráfico comparativo entre os coeficientes do filtro adaptativo com os coeficientes ótimos.*

*Figura 3 – Curvas de nível da superfície de erro.*

*Figura 4 – Gráfico do erro quadrátivo em dB.*

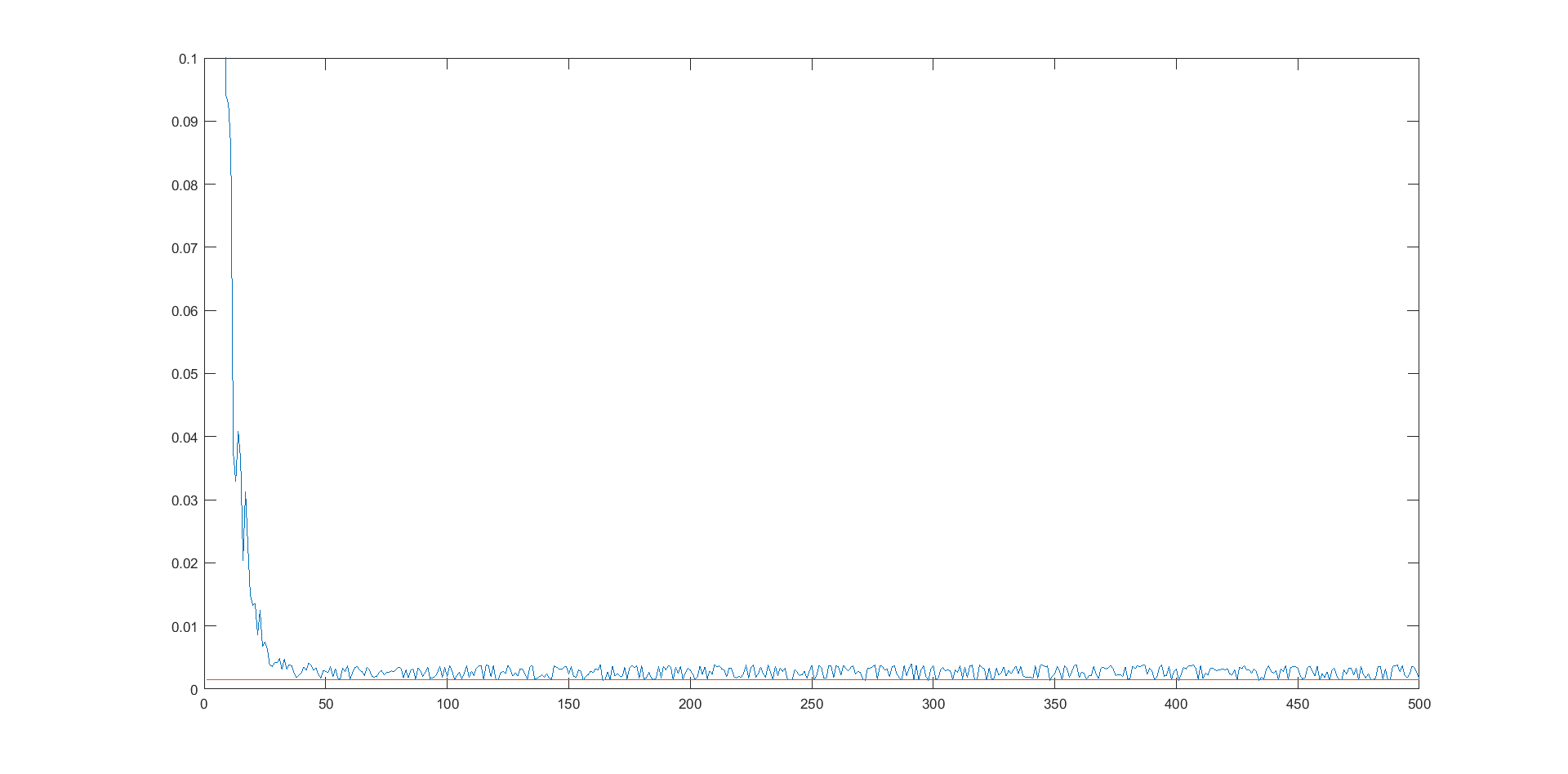
**d)**Foi utilizado uma variação no passo do algoritmo LMS para chegar ao valor mais alto que ainda há convergência. Os resultados são mostrados graficamente na *Figura 5*.

*Figura 5 – Gráfico dos coeficientes do filtro por iteração com diferentes passo*

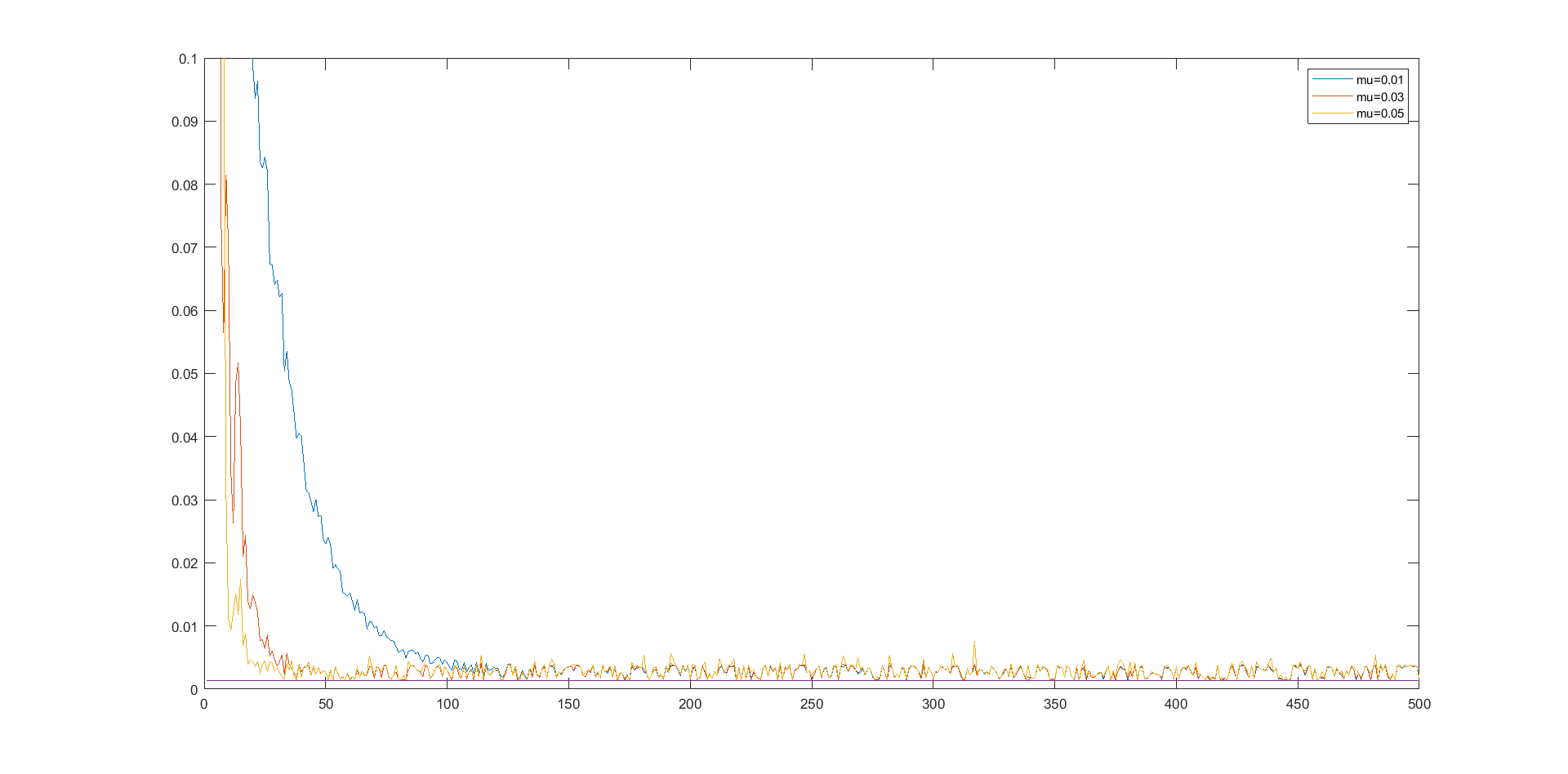
O valor máximo do passo no qual o algoritmo ainda converge alcançado experimentalmente foi:

Comparado ao passo do *Steepest Descent* calculado no item b)*:*

**e)**

*Figura 6 – Gráfico de J(n) por iterações, para μ = 0,03.*

**f)**

*Figura 7 – Gráfico de J(n) por iterações, para μ = 0,01, μ = 0,03 e μ = 0,05.*

Relativo ao erro quadrático médio temos os resultados para comparar:

Já a relação de velocidade de convergência temos:

**Anexo A – Script de MATLAB para os itens b), c) e d)**

%PTC 5890 - 2019

%Prof: Magno T. M. Silva

%Aluno: Stéfano Albino Vilela Rezende (Ouvinte)

%P2 - Parte Computacional

clear all

close all

clc

%o item a) foi feito a mão e será entregue ao professor.

n = [0:500]

phiv = 2\*pi\*rand(1,1)

phiu = phiv

%Número de coeficiente do Filtro

M=2

%ruído

s = 0.01\*randn(1,size(n,2))

%Interferência indesejada

x = sin(2\*pi.\*n./10+pi/6+phiv)

%sinal correlacionado à interferência

u = 5\*sin(2\*pi.\*n./10+phiu)

%sinal deseja d(n)

d = s+x

%b)

%Cálculo da Matriz R, vetor p e coeficientes ótimos

[R,p,Wo] = Wiener(u,d',M)

A = eig(R)

%Valor máximo de mu para o Algoritmo Steepest Descent

mu = 2/max(A)

%c)

N=500 %Número de iterações para o Algoritmo LMS

%Função LMS

[y,e,W] = LMS (u,d,M,0.03,N)

%Gráfico contendo u(n), e(n) e s(n)

figure(1)

stem(s(1:50))

hold on

stem(u(1:50))

stem(e(1:50))

legend('s(n)','u(n)','e(n)')

%Gráfico de comparação entre os coef ao longo da iterações e os coef ótimos

figure(2)

plot(W)

hold on

plot(ones(500)\*Wo(1))

plot(ones(500)\*Wo(2))

%Função de Custo

sigmad = var(d) %sigma de s(n)

Jmin = sigmad-Wo'\*p

[w0,w1]=meshgrid(-0.1:0.01:.4, -0.4:0.01:0.1)

for i=1:size(w0,1)

for j=1:size(w1,1)

J (i,j)= Jmin + [w0(i,j)-Wo(1,1) w1(i,j)-Wo(2,1)]\*R\*[w0(i,j)-Wo(1,1) w1(i,j)-Wo(2,1)]'

end

end

v=[Jmin+0.05 Jmin+0.05]

figure(3)

contour(w0,w1,J,50)%,'ShowText','on')

hold on

contour(w0,w1,J,v,'ShowText','on')

for i=1:3:size(W,1)

plot(W(i,1),W(i,2),'r -x')

end

%Cálculo do erro quadrático em dB

edB=10\*log10(e.^2)

%Filtragem do erro por um filtro de média móvel para melhor visualização

%dos resultados

Hmm = 1/64.\*ones(1,64)

fedB = filter(Hmm,[1],edB)

%Curva do erro quadrático em dB

figure(4)

plot(fedB)

%d)

for j=1:10

mumax=0.03+(j\*0.005)

[ymu,emu,Wmu] = LMS (u,d,M,mumax,N)

figure(5)

plot(Wmu)

hold on

plot(ones(500)\*Wo(1))

plot(ones(500)\*Wo(2))

end

**Anexo B – Script de MATLAB para o e)**

%PTC 5890 - 2019

%Prof: Magno T. M. Silva

%Aluno: Stéfano Albino Vilela Rezende (Ouvinte)

%P2 - Parte Computacional

clear all

close all

clc

%e)

J=zeros(1,500)

mu=0.03 %passo do LMS

N = 500

n = [0:500] %número de amostras do sinais

M=2 %número de coeficiente do Filtro

for i=1:500

phiv = 2\*pi\*rand(1,1)

phiu = phiv

%ruído

s = 0.01\*randn(1,size(n,2))

%Interferência indesejada

x = sin(2\*pi.\*n./10+pi/6+phiv)

%sinal correlacionado à interferência

u = 5\*sin(2\*pi.\*n./10+phiu)

%sinal deseja d(n)

d = s+x

[R,p,Wo] = Wiener(u,d',M)

[y,e,W] = LMS (u,d,M,mu,N)

sigmad = var(d)

Jmin(i) = sigmad-Wo'\*p

%J(n) = E{e^2}

J(i) = Jmin(i) + (W(i,:)'-Wo)'\*R\*(W(i,:)'-Wo)

end

figure(1)

plot (J)

hold on

plot(ones(1,500)\*Jmin(i))

ylim([0,0.1])

EMSE = mean(J)-mean(Jmin)

desaj = EMSE/mean(Jmin)

**Anexo C – Script de MATLAB para o f)**

%PTC 5890 - 2019

%Prof: Magno T. M. Silva

%Aluno: Stéfano Albino Vilela Rezende (Ouvinte)

%P2 - Parte Computacional

clear all

close all

clc

%f)

J=zeros(1,500)

%passos do LMS

mu1=0.01

mu3=0.03

mu5=0.05

N = 500

n = [0:500] %número de amostras do sinais

M=2 %número de coeficiente do Filtro

for i=1:500

phiv = 2\*pi\*rand(1,1)

phiu = phiv

%ruído

s = 0.01\*randn(1,size(n,2))

%Interferência indesejada

x = sin(2\*pi.\*n./10+pi/6+phiv)

%sinal correlacionado à interferência

u = 5\*sin(2\*pi.\*n./10+phiu)

%sinal deseja d(n)

d = s+x

sigmad = var(d)

[R,p,Wo] = Wiener(u,d',M)

Jmin(i) = sigmad-Wo'\*p

[y1,e1,W1] = LMS (u,d,M,mu1,N)

%J(n) = E{e^2}

J1(i) = Jmin(i) + (W1(i,:)'-Wo)'\*R\*(W1(i,:)'-Wo)

[y3,e3,W3] = LMS (u,d,M,mu3,N)

%J(n) = E{e^2}

J3(i) = Jmin(i) + (W3(i,:)'-Wo)'\*R\*(W3(i,:)'-Wo)

[y5,e5,W5] = LMS (u,d,M,mu5,N)

%J(n) = E{e^2}

J5(i) = Jmin(i) + (W5(i,:)'-Wo)'\*R\*(W5(i,:)'-Wo)

end

figure(1)

plot (J1)

hold on

plot (J3)

plot (J5)

plot(ones(1,500)\*Jmin(i))

ylim([0,0.1])

legend('mu=0.01','mu=0.03','mu=0.05')

EMSE1 = mean(J1)-mean(Jmin)

desaj1 = EMSE1/mean(Jmin)

EMSE3 = mean(J3)-mean(Jmin)

desaj3 = EMSE3/mean(Jmin)

EMSE5 = mean(J5)-mean(Jmin)

desaj5 = EMSE5/mean(Jmin)

**Anexo D – Função de MATLAB para a solução de Wiener**

function [R,p,Wo] = Wiener(u,d,M)

r=xcorr (u,M-1,'biased')

ru=r(M:end);

R=toeplitz(ru)

rdu=xcorr(d,u,M-1,'biased')

p=rdu(M:end)

Wo=R\p

**Anexo E – Função de MATLAB para o algoritmo LMS**

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

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\*e(n)\*uM';

end

end