# Αλγόριθμοι Προσαρμοστικών Φίλτρων LMS/RLS
# Εφαρμογή: Εξίσωση Καναλιού με χρήση προσαρμοστικών φίλτρων

Χάρης Γεωργίου, 2000

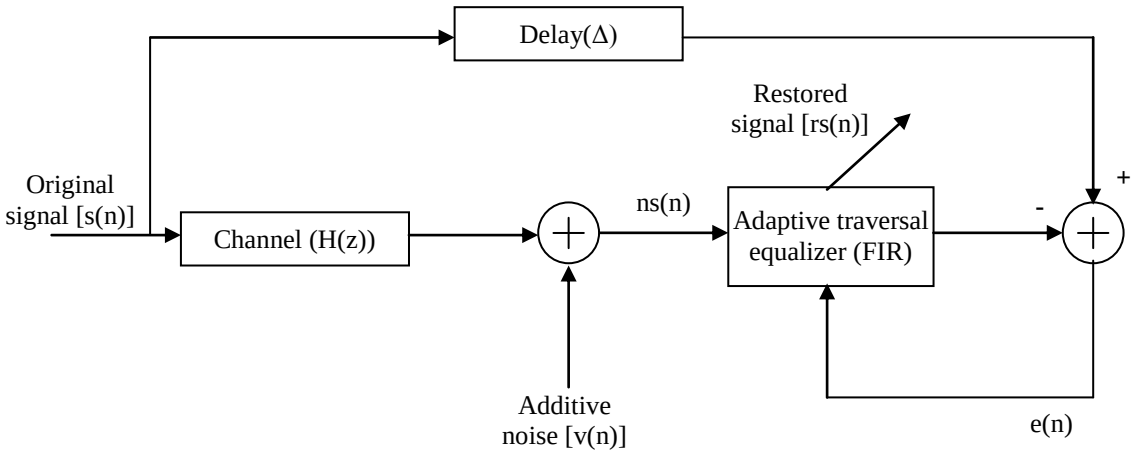
*Οι αναφορές στο [1] αφορούν το βιβλίο "Adaptive Filter Theory" (Simon Haykin, 3rd/ed.) του οποίου η ύλη καλύπτει πλήρως και λεπτομερώς όλα όσα έχουν αναφερθεί στο μάθημα.*

Το πρόβλημα της εξίσωσης καναλιού σε μέσο μετάδοσης για την καταστολή φαινομένων ISI προσεγγίζεται ως εξής. Δημιουργούμε μια ψευδοτυχαία ακολουθία [-1,+1] που αποτελεί το αρχικό σήμα πληροφορίας. Τα χαρακτηριστικά του καναλιού μετάδοσης περιγράφονται από τη συνάρτηση μεταφοράς H(z):

$$H(z) = 0.04 - 0.05z^{-1}\ 0.07z^{-2} - 0.21z^{-3} - 0.5z^{-4} + 0.72z^{-5}\ 0.36z^{-6}\ 0.21z^{-8}\ 0.03z^{-9}\ 0.07z^{-10}$$

Στην έξοδο του καναλιού (δέκτης) προστίθεται θόρυβος ισχύος $N_0$ έτσι ώστε το SNR (signal-to-noise ratio) να είναι 10dB.
Το πρόβλημα έγκειται στην σχεδίαση και υλοποίηση ενός βέλτιστου φίλτρου για την εξίσωση του καναλιού, χρησιμοποιώντας τους προσαρμοστικούς αλγορίθμους LMS (Least Mean Squares) και RLS (Recursive Least Squares).

Για την εξίσωση του καναλιού θα χρησιμοποιηθούν δείγματα από το αρχικό σήμα της πληροφορίας, καθυστερημένα κατά Δ, ως επιθυμητή έξοδος του φίλτρου ενώ ο υπολογισμός των παραμέτρων θα γίνεται επαναληπτικά με βάση τους αλγορίθμους LMS και RLS. Το παρακάτω διάγραμμα παρουσίαζει τη μορφή του συστήματος που πρόκειται να υλοποιηθεί.



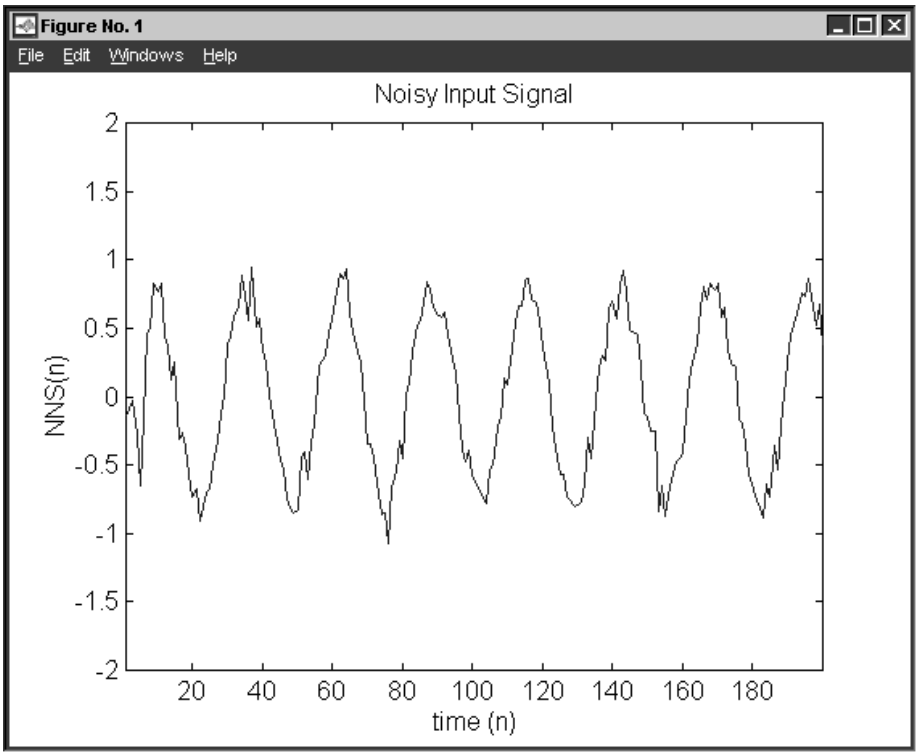

Η υλοποίηση των αλγορίθμων LMS και RLS έγινε με βάση το ίδιο γενικό σχήμα. Για τον προγραμματισμό τους χρησιμοποιήθηκαν αναλυτικοί τύποι (βρόγχοι) για τις συνελίξεις και τις πράξεις με διανύσματα και πίνακες, έτσι ώστε να είναι πλήρως προσαρμόσιμοι σε κάθε επιλογή των παραμέτρων (filter size, signal size, tap delay, SNR). Επίσης υλοποιήθηκε ξεχωριστή ρουτίνα υπολογισμού των ακράιων ορίων για τις παραμέτρους προσαρμογής (μ, λ) και επαναληπτική διερεύνηση για τη βέλτιστη δυνατή επιλογή (ως προς το MSE).

Για τη δοκιμή της ορθότητας και της απόδοσης των δύο υλοποιήσεων χρησιμοποιήθηκαν δύο τύποι σήματος εισόδου. Το πρώτο είναι ένα απλό σήμα cosine με μικρή τυχαία απόκλιση ως προς τη φάση, ενώ το δεύτερο είναι το ίδιο σήμα κβαντισμένο ως προς [-1,+1], δηλαδή το αντίστοιχο τετραγωνικό.

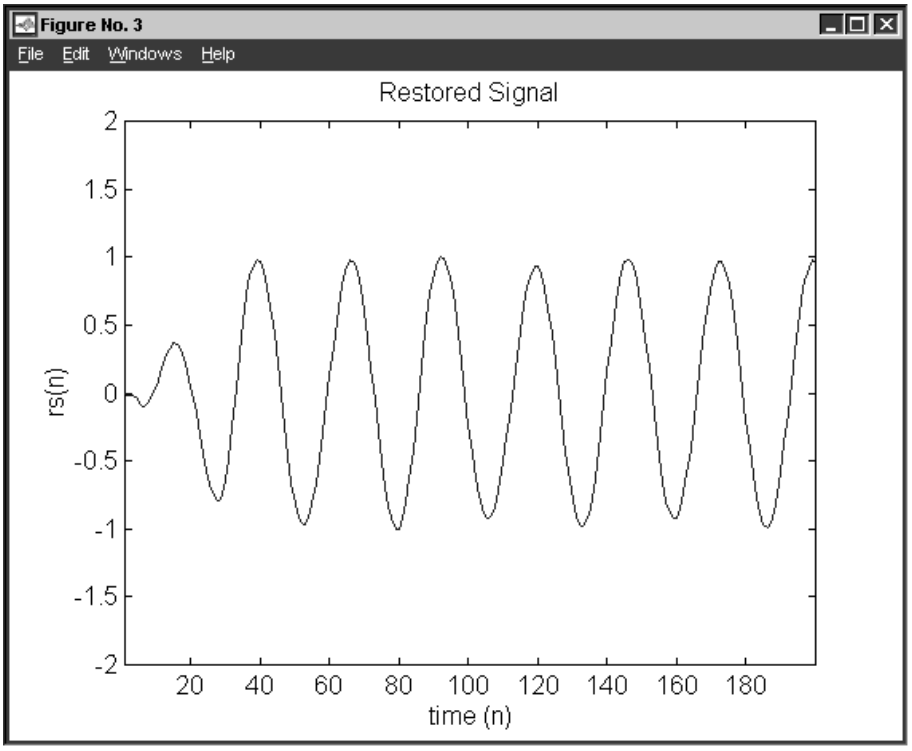
Οι παρακάτω γραφικές παραστάσεις παρουσιάζουν τα αποτελέσματα της εφαρμογής των υλοποιήσεων LMS και RLS για μήκος φίλτρου M=31, καθυστέρηση Δ=15 και τις βέλτιστες παραμέτρους προσαρμογής (μ και λ αντίστοιχα).

## **<u>Πείραμα 1ο:</u> s(n)=cos((n-1)*0.075π+randn*0.001), μ=0.116, λ=0.5**

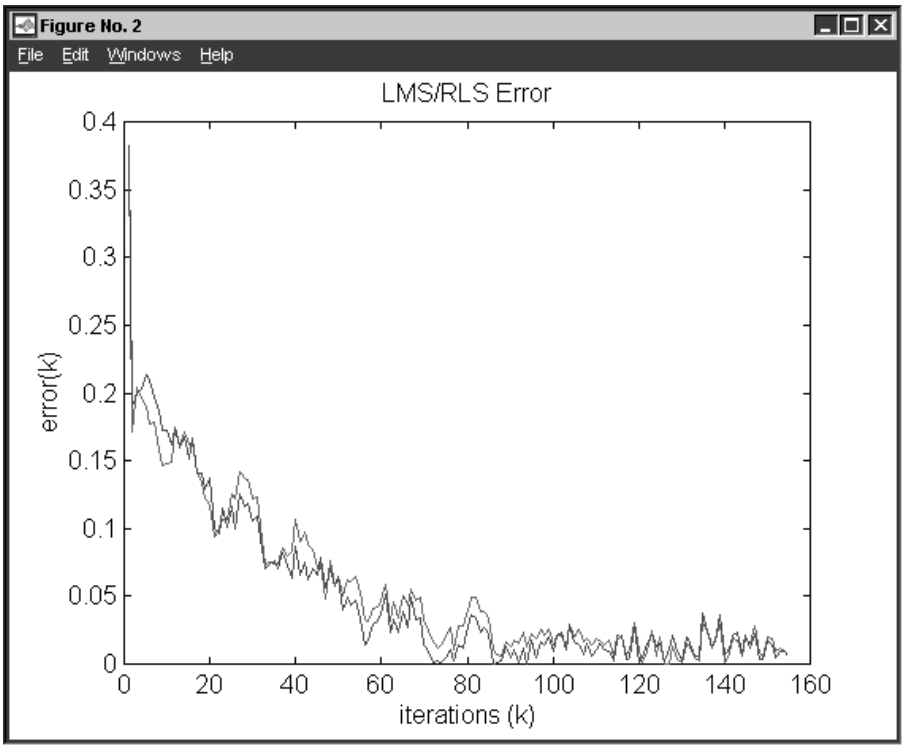
Σήμα δέκτη με προσθήκη θορύβου:



Ανακατασκευασμένο σήμα από εξισωτή:

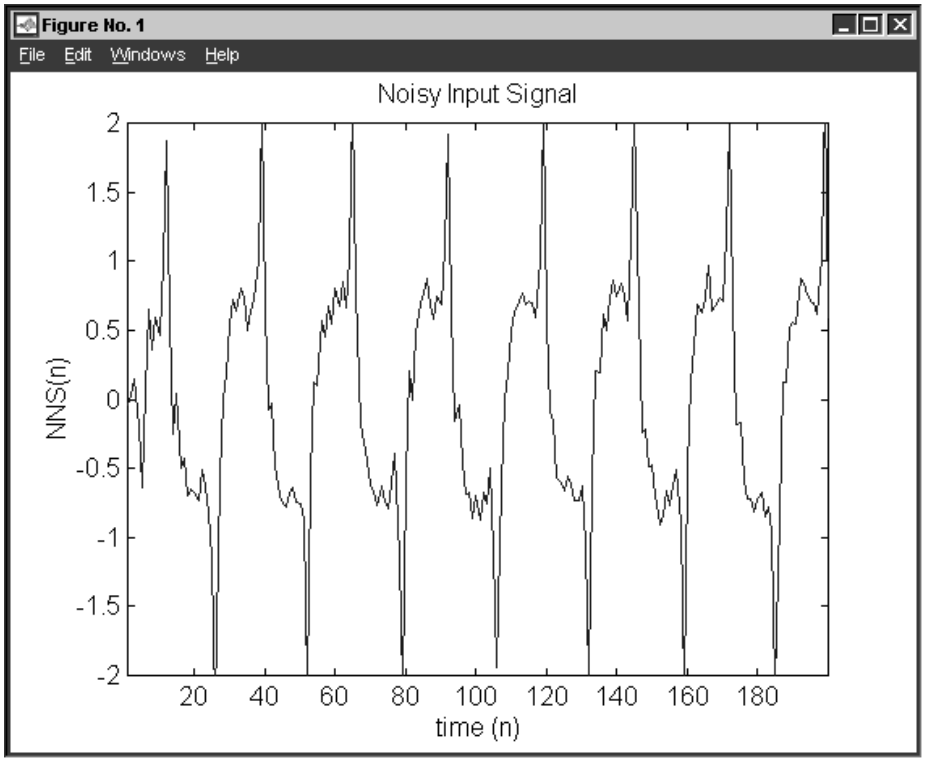



Σφάλμα σύγκλισης για τους LMS/RLS:

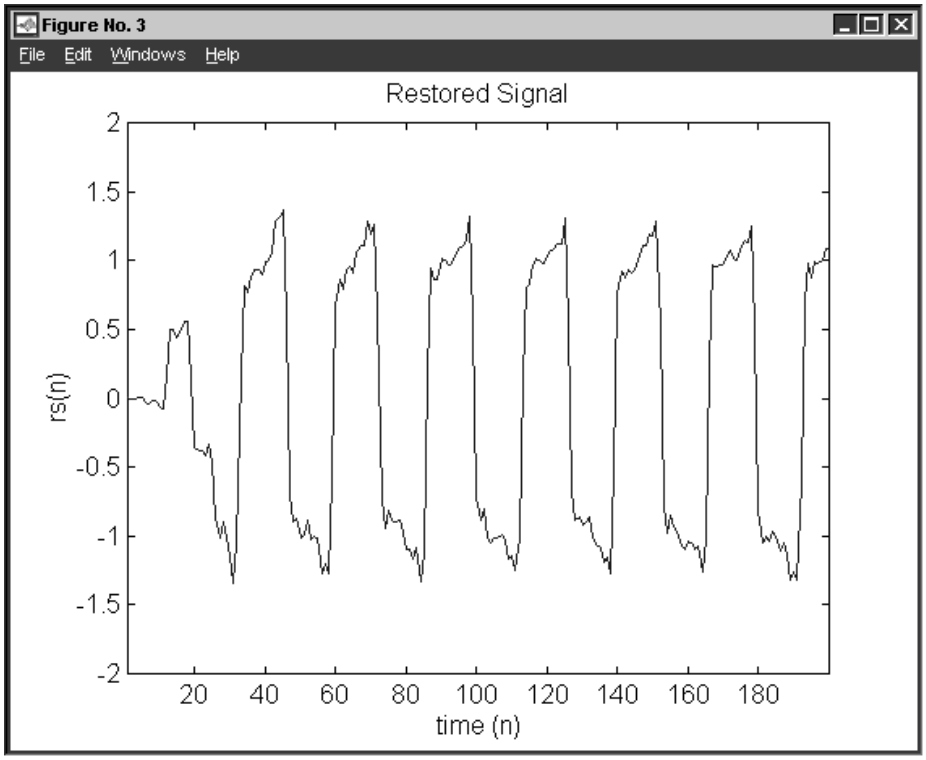


**<u>Πείραμα 2ο:</u> s(n)=[cos((n-1)*0.075π+randn*0.001)], μ=0.026, λ=1.0**

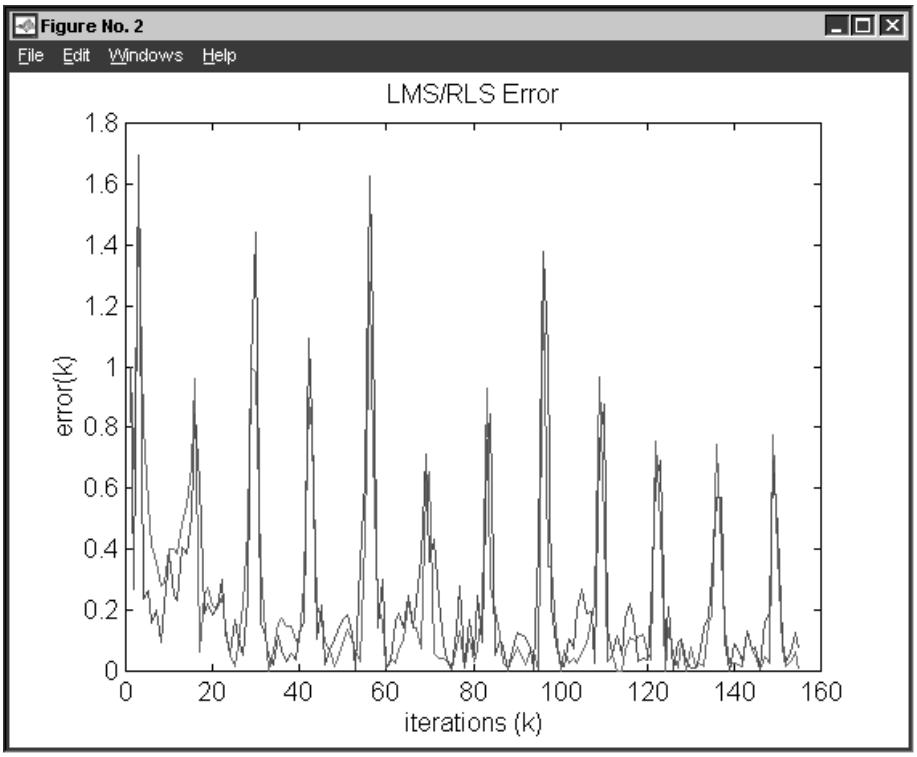
Σήμα δέκτη με προσθήκη θορύβου:



Ανακατασκευασμένο σήμα από εξισωτή:



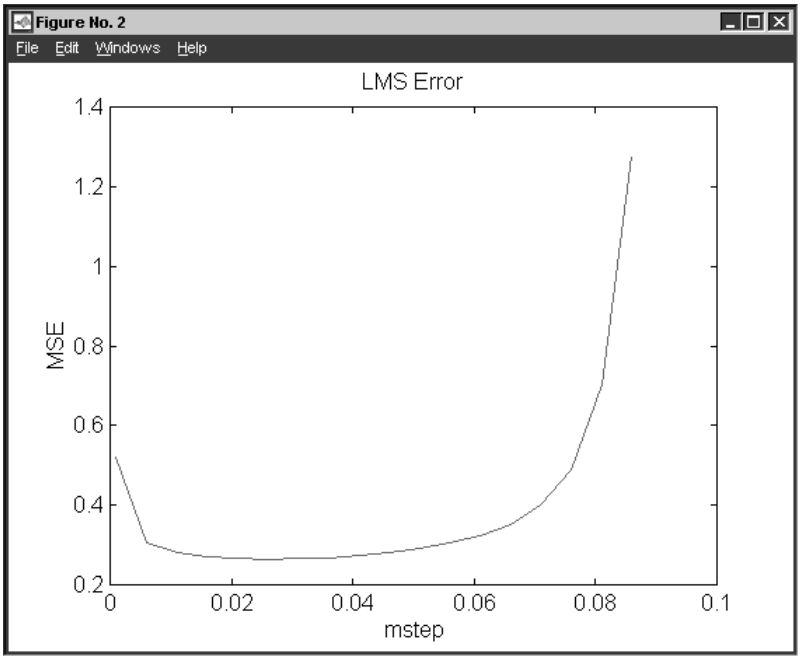

Σφάλμα σύγκλισης για τους LMS/RLS:



## Σχόλια & Παρατηρήσεις

Τα παραπάνω διαγράμματα είναι ενδεικτικά της λειτουργίας και της συμπεριφοράς των δύο υπό εξέταση αλγορίθμων. Στο πρώτο πείραμα το σήμα της πληροφορίας είναι αρκετά απλό και κατά συνέπεια και οι δύο αλγόριθμοι συγκλίνουν χωρίς ιδιαίτερη δυσκολία. Στη δεύτερη περίπτωση το σήμα εισόδου χαρακτηρίζεται από απότομες αλλαγές (λόγω κβάντισης) και έτσι η σύγκλιση είναι δυσκολότερη. Αν και η συμπεριφορά των δύο αλγορίθμων ως προς το σφάλμα σύγκλισης φαίνεται ουσιαστικά ίδια, παρόλα αυτά ο αλγόριθμος RLS φαίνεται να συγκλίνει με πιο σταθερό ρυθμό, γεγονός που επιβεβαιώνεται καλύτερα σε χαμηλότερα επίπεδα θορύβου (SNR=30dB).

Σε σχέση με την επιλογή της παραμέτρου προσαρμογής μ για τον LMS, στην υλοποίηση περιλαμβάνονται ειδικές ρουτίνες για την εύρεση της μέγιστης τιμής με βάση το "tap input power" TIP (πειραματικά [1]: $TIP=\Sigma(E[|u(n-\kappa)|^2])$ ), δηλαδή: $0 < \mu < 2/TIP$. Η διερεύνηση γίνεται επαναληπτικά με γραμμικό τρόπο, με κριτήριο την τιμή MSE. Για τον αλγόριθμο RLS υπάρχει αντίστοιχη ρουτίνα αναζήτησης της βέλτιστης τιμής της παραμέτρου λ στο πεδίο [0.5 , 1.0], αλλά δεν παρατηρήθηκαν σημαντικές διαφοροποιήσεις την απόδοση. Το παρακάτω σχήμα παρουσιάζει τη μεταβολή του MSE του LMS για διάφορες τιμές του μ:

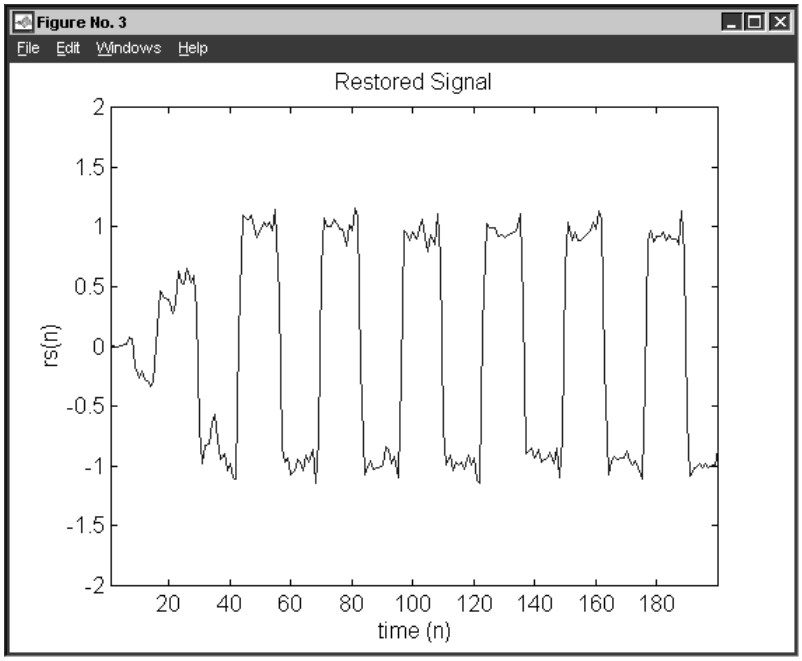



Μια ακόμη σημαντική παράμετρος στη σχεδίαση του συστήματος είναι η καθυστέρηση των δειγμάτων (Δ). Για σταθερό μήκος φίλτρου και παραμέτρους προσαρμογής, δοκιμές με βάση το τετραγωνικό σήμα εισόδου έδειξαν ότι αύξηση στην τιμή του Δ έχει ως αποτέλεσμα μικρότερη ακρίβεια στην ανακατασκευή του αρχικού σήματος από το φίλτρο, καθώς και (οριακά) μεγαλύτερο σφάλμα σύγκλισης. Τέλος, ο συνολικός χρόνος επεξεργασίας εξαρτάται από το μήκος του φίλτρου (M=31), με τον αλγόριθμο RLS να είναι, όπως ήταν αναμενόμενο, μια τάξη μεγέθους πιο αργός από τον LMS.

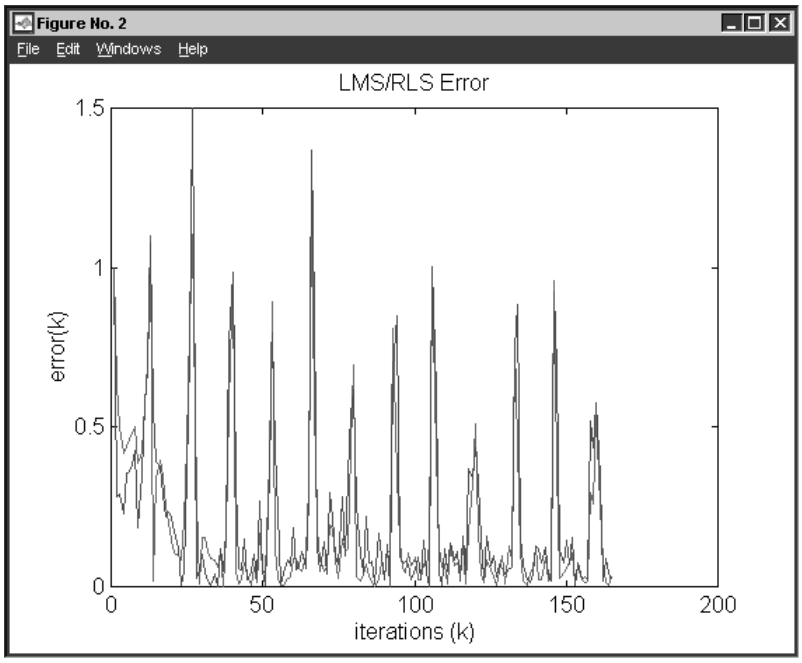
Παρακάτω παρουσιάζονται διαγράμματα του ανακατασκευασμένου σήματος και των σφαλμάτων σύγκλισης LMS/RLS για Δ=5 και Δ=25:

## Καθυστέρηση Δ=5

Ανακατασκευασμένο σήμα:

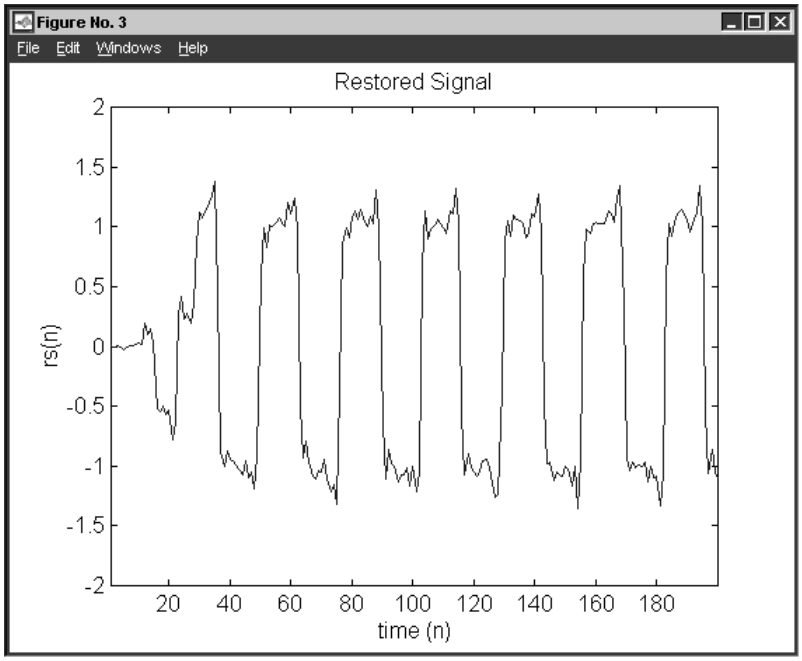



Σφάλματα σύγκλισης LMS/RLS:

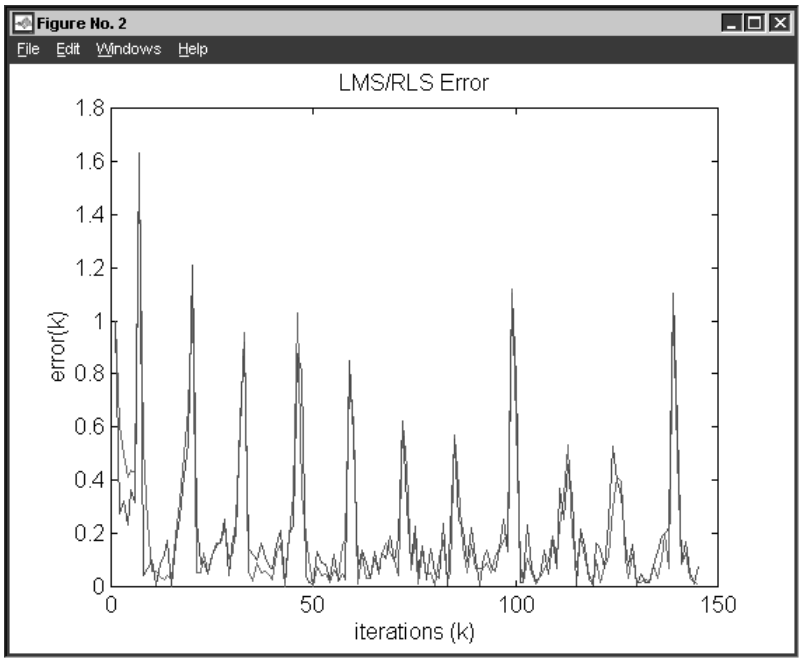


## Καθυστέρηση Δ=25

Ανακατασκευασμένο σήμα:




Σφάλματα σύγκλισης LMS/RLS:



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                          %%%
%%%   Adaptive Channel Equalization (LMS/RLS)                %%%
%%%   ---------------------------------------                %%%
%%%                                                          %%%
%%%                                                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------  LMS tap input power  --------%
function [mstep_max] = filt_tip(Ins,ftaps)

N=size(Ins,2); M=ftaps;
%... calculate tap input power & max step-size value ...%
SSum=0;
for k=0:M-1,
    sum=0;
    for i=k+1:N, sum=sum+Ins(i)*Ins(i); end;
    SSum=SSum+sum/(N-k);
end;

mstep_max=2/SSum;

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


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                          %%%
%%%   Adaptive Channel Equalization (LMS/RLS)                %%%
%%%   ---------------------------------------                %%%
%%%                                                          %%%
%%%                                                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------  find best LMS step size  --------%
function [msteps,merrs,bestmpos] =
filt_mst(Ins,Des,ftaps,dtaps,ms_min,ms_max,ms_step,verbose)

N=size(Ins,2); M=ftaps; D=dtaps;
bestmpos=1;  besterr=inf;  c=1;

for mstep=ms_min:ms_step:ms_min+ms_max,
    [W,E]=filt_lms(Ins,Des,M,D,mstep,0);
    errssum=0.0;
    for k=1:size(E,2), errssum=errssum+E(k)*E(k); end;
    errssum=errssum/size(E,2);
    if (errssum<=besterr) bestmpos=c; besterr=errssum; end;
    msteps(c)=mstep; merrs(c)=errssum; c=c+1;
    if (verbose~=0)
        mstep
        errssum
    end;
end;

if (verbose~=0)
    bestmpos
    bestmstep=msteps(bestmpos)
    besterr=merrs(bestmpos)
end;

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                          %%%
%%%   Adaptive Channel Equalization (LMS/RLS)                %%%
%%%   ---------------------------------------                %%%
```

```
%%%                                                         %%%
%%%                                                         %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------  LMS recursive adaptation  --------%
function [W,E] = filt_lms(Ins,Des,ftaps,dtaps,mstep,verbose)

N=size(Ins,2); M=ftaps; D=dtaps;
%... begin LMS iterations ...%

W=zeros(1,M); Uest=zeros(1,N);
ERR=zeros(1,N);

for i=M:N-D,
    if (verbose~=0) disp(N-D-i+1); end;

    ut=0;
    for j=1:M, ut=ut+W(j)*Ins(i-j+1); end;    % calculate current estimate
    Uest(i+D)=ut;

    ERR(i)=Des(i+D)-Uest(i+D);                % calculate current error

    for j=1:M,
        W(j)=W(j)+mstep*ERR(i)*Ins(i-j+1);       % adjust tap weights
    end;
end;

E=Des-Uest;

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



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                         %%%
%%%   Adaptive Channel Equalization (LMS/RLS)               %%%
%%%   ---------------------------------------               %%%
%%%                                                         %%%
%%%                                                         %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------  find best RLS lambda value  --------%
function [lsteps,lerrs,bestlpos] =
filt_lst(Ins,Des,ftaps,dtaps,delta,ls_min,ls_max,ls_step,verbose)

N=size(Ins,2); M=ftaps; D=dtaps;
bestlpos=1;  besterr=inf; c=1;

for lstep=ls_min:ls_step:ls_max,
    [W,E]=filt_rls(Ins,Des,M,D,delta,lstep,0);
    errssum=0.0;
    for k=1:size(E,2), errssum=errssum+E(k)*E(k); end;
    errssum=errssum/size(E,2);
    if (errssum<=besterr) bestlpos=c; besterr=errssum; end;
    lsteps(c)=lstep; lerrs(c)=errssum; c=c+1;
    if (verbose~=0)
        lstep
        errssum
    end;
end;

if (verbose~=0)
    bestlpos
    bestlstep=lsteps(bestlpos)
    besterr=lerrs(bestlpos)
```

```
end;

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                         %%%
%%%   Adaptive Channel Equalization (LMS/RLS)               %%%
%%%   ---------------------------------------               %%%
%%%                                                         %%%
%%%                                                         %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------  RLS recursive adaptation  --------%
function [W,E] = filt_rls(Ins,Des,ftaps,dtaps,delta,lambda,verbose)

%... begin RLS iterations ...%
N=size(Ins,2); M=ftaps; D=dtaps;
P=eye(M)/delta;
W=zeros(1,M); Uest=zeros(1,N);
ERRx=zeros(1,N); z=zeros(1,M); g=z;

for i=M:N-D,
    if (verbose~=0) disp(N-D-i+1); end;

    for j=1:M,
        z(j)=0;
        for k=1:M, z(j)=z(j)+P(j,k)*Ins(i-j+1); end;
    end;
    zz=0;  for j=1:M, zz=zz+Ins(i-j+1)*z(j); end;
    g=z/(lambda+zz);

    ut=0;
    for j=1:M, ut=ut+W(j)*Ins(i-j+1); end;    % calculate current estimate
    Uest(i+D)=ut;

    ERRx(i)=Des(i+D)-Uest(i+D);                % calculate current error

    for j=1:M,
        W(j)=W(j)+ERRx(i)*g(j);       % adjust tap weights
    end;

    P=(P-g'*z)/lambda;
end;

E=Des-Uest;

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                         %%%
%%%   Adaptive Channel Equalization (LMS/RLS)               %%%
%%%   ---------------------------------------               %%%
%%%                                                         %%%
%%%                                                         %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('Adaptive Channel Equalization, LMS & RLS adaptive filters');
```

```
%... initialize constants and variables ...%
clear all;

disp('Initializing variables');

N=200;          % number of samples
M=31;           % filter order
D=25;           % tap delay

H=[0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0.0 0.21 0.03 0.07];
HN=size(H);
SNRdb=10;       % selected SNR level for receiver noise

for i=1:N, s(i)=cos((i-1)*0.075*pi+0.001*randn); end;
for i=1:N,
    if (s(i)>=0) s(i)=1; else s(i)=-1; end;
end;

%... plot original signal ...%
disp('Creating original signal plot');
figure(1);
plot(s);
axis([1,N,-2,2]);
xlabel('time (n)');
ylabel('S(n)');
title('Original Signal');
disp('...<press any key to continue>...');
pause;

ns=filter(H,1,s);

%... plot noisy signal signal ...%
%Hsqr=H.^2; HSS=0;
%for i=1:HN, HSS=HSS+Hsqr(i); end
%v=sqrt(HSS/(10^(SNRdb/10)));
%HSS
v=1/(10^(SNRdb/10));
rcvn_var=v
for i=1:N, nns(i)=ns(i)+randn*v; end;

%... plot noisy input signal ...%
disp('Creating noisy input signal');
figure(1);
plot(nns);
axis([1,N,-2,2]);
xlabel('time (n)');
ylabel('NNS(n)');
title('Noisy Input Signal');
disp('...<press any key to continue>...');
pause;


%--------  LMS recursive adaptation  --------%

%... begin LMS iterations ...%
disp(' ');
disp('Beginning LMS iterations...');

Ins=nns; Des=s; mstep=1.0;

min_mstep=0.001; step_mstep=0.005; max_mstep=filt_tip(Ins,M);

[MS,ME,BMP]=filt_mst(Ins,Des,M,D,min_mstep,max_mstep,step_mstep,1);
%... plot LMS error vs mstep ...%
figure(2);
```

```
plot(MS,ME,'g');
xlabel('mstep');
ylabel('MSE');
title('LMS Error');
disp('...<press any key to continue>...');
pause;

mstep=MS(BMP);
[W1,E1]=filt_lms(Ins,Des,M,D,mstep,1);

disp('Best LMS configuration:');
mstep
W1

%... plot LMS error ...%
ens=abs(E1); fns=ens(M+D:N);
disp('Creating LMS error plot');
figure(2);
plot(fns);
xlabel('time (n)');
ylabel('MSE(n)');
title('LMS Error');
disp('...<press any key to continue>...');
pause;

%... plot restored signal ...%
rs=filter(W1,1,nns);
disp('Creating restored signal');
figure(3);
plot(rs);
axis([1,N,-2,2]);
xlabel('time (n)');
ylabel('rs(n)');
title('Restored Signal');
disp('...<press any key to continue>...');
pause;

%--------  RLS recursive adaptation  --------%

%... begin RLS iterations ...%
disp('Beginning RLS iterations...');

delta=0.001;  lambda = 1.0;
Ins=nns; Des=s;

min_lstep=0.5; step_lstep=0.1; max_lstep=1.0;

[LS,LE,BLP]=filt_lst(Ins,Des,M,D,delta,min_lstep,max_lstep,step_lstep,1);
%... plot RLS error vs mstep ...%
figure(2);
plot(LS,LE,'c');
xlabel('lstep');
ylabel('MSE');
title('RLS Error');
disp('...<press any key to continue>...');
pause;

lambda=LS(BLP);
[W2,E2]=filt_rls(Ins,Des,M,D,delta,lambda,1);

disp('Best RLS configuration:');
lambda
W2

%... plot RLS error ...%
```

```
ens=abs(E2); fns=ens(M+D:N);
disp('Creating RLS error plot');
figure(2);
plot(fns);
xlabel('time (n)');
ylabel('MSE(n)');
title('RLS Error');
disp('...<press any key to continue>...');
pause;

%... plot restored signal ...%
rs=filter(W2,1,nns);
disp('Creating restored signal');
figure(3);
plot(rs);
axis([1,N,-2,2]);
xlabel('time (n)');
ylabel('rs(n)');
title('Restored Signal');
disp('...<press any key to continue>...');
pause;


%... plot combined error graph ...%
disp('Creating combined error graph');
idx=[1:1:N-M-D+1]; LMSerr=abs(E1(M+D:N)); RLSerr=abs(E2(M+D:N));
figure(2);
plot(idx,LMSerr,'g',idx,RLSerr,'c');
xlabel('iterations (k)');
ylabel('error(k)');
title('LMS/RLS Error');
%disp('...<press any key to continue>...');
%pause;


disp('Process completed.');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Adaptive Channel Equalization, LMS & RLS adaptive filters
            Harris Georgiou [AM:M177], grad0177@di.uoa.gr

Initializing variables
Creating noisy input signal

Beginning LMS iterations...
Best LMS configuration:

mstep = 0.0310

W1 =
```

```
  Columns 1 through 7

    0.0059    0.0145    0.0052   -0.0115   -0.0171   -0.0857   -0.0334

  Columns 8 through 14

   -0.1294   -0.0914   -0.2613   -0.3121    0.2160    0.0686   -0.0084

  Columns 15 through 21

   -0.0092   -0.0120   -0.0066   -0.0038    0.0085    0.0138    0.0590

  Columns 22 through 28

    0.0844    0.1639    0.2606    0.0414   -0.1005   -0.0715   -0.0456

  Columns 29 through 31

   -0.0391   -0.0292   -0.0307


Beginning RLS iterations...
Best RLS configuration:

lambda = 1

W2 =

  Columns 1 through 7

    0.0052    0.0205    0.0041   -0.0234   -0.0269   -0.1089    0.0017

  Columns 8 through 14

   -0.1320   -0.0219   -0.2446   -0.3255    0.3202    0.0104   -0.0155

  Columns 15 through 21

   -0.0350   -0.0226   -0.0221   -0.0062    0.0048    0.0092    0.0652

  Columns 22 through 28

    0.0939    0.1441    0.2879    0.1078   -0.1174   -0.1157   -0.0599

  Columns 29 through 31

   -0.0564   -0.0363   -0.0429
```

9 Φεβρουαρίου 2000.