Nuestro filtro adaptativo, lo que hace en nuestro sistema es aprender cómo se comportaría dicha interferencia a través de su sistema predicitivo. Y es así como funciona nuestro ejemplo práctico, a continuación se explicará cómo se lo ha realizado, en MatLab:

Para tomar el sonido de audio, reproducirlo y graficarlo:

[se,fs]=wavread('prueb1.wav');

soundsc(aux,fs);

axes(handles.axes1);

plot(se);

Este sonido de audio esta a una tasa de muestreo fs=6000.

Graficamos nuestra interferencia una onda seno.

clc;

fs=6000;

t=0:1/fs:10009/fs;

w=sin(2\*pi\*t\*10010);

Sumamos la señal original con el ruido

[se,fs]=wavread('prueb1.wav');

fs=6000;

t=0:1/fs:10009/fs;

w=sin(2\*pi\*t\*10010);

w=w';

r=se+w;

Par a realizar el filtro adaptativo y entrenarlo se realizó lo siguiente.

Primero transformamos un vector de matriz en un array a cada valor.

P=con2seq(w);

T=con2seq(r);

Graficamos la Señal para ello hacemos la siguiente conversión:

plot(t,cat(2,T{:}),'g',t,cat(2,P{:}),'--'); %Convierte de array a double

Creamos la Red Adaline, definiendo los valores máximos, mínimos, error y el número de neuronas. Damos el número de iteraciones mediante epoch.

wts=net.IW{1,1};

bias=net.b{1};

net.adaptParam.epoch=20;

Entrenamos a la red

[net,Y,E,Pf]=adapt(net,P,T);

Cambiamos de formato a la variable Y que es de tipo cell a un array

Y=cell2mat(Y);

Y restamos el valor que nos ha salido, el ruido de la señal original y la escuchamos:

for i=1:10010

e(:,i)=r(:,i)-Y(:,i);

end

soundsc(e,fs);

**CODIGO M.FILE**

function varargout = adapt\_neuro(varargin)

% ADAPT\_NEURO M-file for adapt\_neuro.fig

% ADAPT\_NEURO, by itself, creates a new ADAPT\_NEURO or raises the existing

% singleton\*.

%

% H = ADAPT\_NEURO returns the handle to a new ADAPT\_NEURO or the handle to

% the existing singleton\*.

%

% ADAPT\_NEURO('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in ADAPT\_NEURO.M with the given input arguments.

%

% ADAPT\_NEURO('Property','Value',...) creates a new ADAPT\_NEURO or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before adapt\_neuro\_OpeningFunction gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to adapt\_neuro\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help adapt\_neuro

% Last Modified by GUIDE v2.5 09-Dec-2007 23:35:53

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @adapt\_neuro\_OpeningFcn, ...

'gui\_OutputFcn', @adapt\_neuro\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before adapt\_neuro is made visible.

function adapt\_neuro\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to adapt\_neuro (see VARARGIN)

% Choose default command line output for adapt\_neuro

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes adapt\_neuro wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = adapt\_neuro\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[se,fs]=wavread('prueb1.wav');

aux=se;

soundsc(aux,fs);

%aux1=aux(:,1);

tam1=size(se)

axes(handles.axes1);

plot(se);

grid on;

title('Señal de Audio');

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

clc;

fs=6000;

t=0:1/fs:10009/fs;

w=sin(2\*pi\*t\*10010);

tam2=size(w)

w=w';

%graficar señal interferente. seno w

axes(handles.axes2);

plot(w);

grid on;

title('Señal Interferente');

grid on;

soundsc(w,fs);

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

clc;

[se,fs]=wavread('prueb1.wav');

fs=6000;

t=0:1/fs:10009/fs;

w=sin(2\*pi\*t\*10010);

w=w';

r=se+w;

R=size(r)

axes(handles.axes3);

plot(r);

title('Voz + Interferencia');

grid on;

soundsc(r,fs);

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

clc;

[se,fs]=wavread('prueb1.wav');

aux=se';

fs=6000;

t=0:1/fs:10009/fs;

w=sin(2\*pi\*t\*10010);

r=aux+w;

P=con2seq(w); %Hacemos en filas, no en columnas

T=con2seq(r);

axes(handles.axes4);

plot(t,cat(2,T{:}),'g',t,cat(2,P{:}),'--'); %Convierte de array a double

title('T: Señal Esperada/P: Señal Ruido'); %P=ruido

grid on; %T=señal esperada

net=newlin([-2,2],1,[0,1],0.2); %Definimos valores max, min y 1 neurona

%Error mínimo: 0.2

wts=net.IW{1,1};

bias=net.b{1};

%net.adaptParam.passes=10;

net.adaptParam.epoch=20;

[net,Y,E,Pf]=adapt(net,P,T);

axes(handles.axes5);

plot(t,cat(2,Y{:}),'c');

title('Señal Filtrada');

grid on;

Y=cell2mat(Y);

for i=1:10010

e(:,i)=r(:,i)-Y(:,i);

end

soundsc(e,fs);