APPENDIX B. FUNCTION AUTOFAM

```
function [Sx,alphao,fo]=autofam(x,fs,df,dalpha)
%         AUTOFAM(X,FS,DF,DALPHA) computes the spectral auto-correlation
%         density function estimate of the signal X, by using the FFT
%         Accumulation Method(FAM). Make sure that DF is much bigger
%         than DALPHA in order to have a reliable estimate.
%
%         INPUTS:
%         X      - input column vector;
%         FS     - sampling rate;
%         DF     - desired frequency resolution; and
%         DALPHA - desired cyclic frequency resolution.
%
%         OUTPUTS:
%         SX     - spectral correlation density function estimate;
%         ALPHAO - cyclic frequency; and
%         FO     - spectrum frequency.
%
%         Author: E.L.Da Costa,9/28/95.

if nargin ~= 4
   error('Wrong number of arguments.');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definition of Parameters %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Np=pow2(nextpow2(fs/df));       % Number of input channels, defined
                                % by the desired frequency
                                % resolution(df) as follows:
                                % Np=fs/df, where fs is the original
                                % data sampling rate. It must be a
                                % power of 2 to avoid truncation or
```

```
                                        % zero-padding in the FFT routines;
L=Np/4;                                 % Offset between points in the same
                                        % column at consecutive rows in the
                                        % same channelization matrix. It
                                        % should be chosen to be less than
                                        % or equal to Np/4;
P=pow2(nextpow2(fs/dalpha/L));          % Number of rows formed in the
                                        % channelization matrix, defined
                                        % by the desired cyclic frequency
                                        % resolution(dalpha) as follows:
                                        % P=fs/dalpha/L.It must be a power
                                        % of 2;
N=P*L;                                  % Total number of points in the
                                        % input data.

%%%%%%%%%%%%%%%%%%%%%%%%
% Input Channelization %
%%%%%%%%%%%%%%%%%%%%%%%%
if length(x)<N
   x(N)=0;
elseif length(x)>N
   x=x(1:N);
end
NN=(P-1)*L+Np;
xx=x;
xx(NN)=0;
xx=xx(:);
X=zeros(Np,P);
for k=0:P-1
    X(:,k+1)=xx(k*L+1:k*L+Np);
end

%%%%%%%%%%%%%
% Windowing %
%%%%%%%%%%%%%
```

```
a=hamming(Np);
XW=diag(a)*X;
XW=X;


%%%%%%%%%%%%%
% First FFT %
%%%%%%%%%%%%%
XF1=fft(XW);
XF1=fftshift(XF1);
XF1=[XF1(:,P/2+1:P) XF1(:,1:P/2)];

%%%%%%%%%%%%%%%%%%
% Downconversion %
%%%%%%%%%%%%%%%%%%
E=zeros(Np,P);
for k=-Np/2:Np/2-1
    for m=0:P-1
        E(k+Np/2+1,m+1)=exp(-i*2*pi*k*m*L/Np);
    end
end
XD=XF1.*E;
XD=conj(XD');

%%%%%%%%%%%%%%%%%%
% Multiplication %
%%%%%%%%%%%%%%%%%%
XM=zeros(P,Np^2);
for k=1:Np
    for l=1:Np
        XM(:,(k-1)*Np+l)=(XD(:,k).*conj(XD(:,l)));
    end
end

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

```
% Second FFT %
%%%%%%%%%%%%%%
XF2=fft(XM);
XF2=fftshift(XF2);
XF2=[XF2(:,Np^2/2+1:Np^2) XF2(:,1:Np^2/2)];
XF2=XF2(P/4:3*P/4,:);
M=abs(XF2);
alphao=-1:1/N:1;
fo=-.5:1/Np:.5;
Sx=zeros(Np+1,2*N+1);
for k1=1:P/2+1
    for k2=1:Np^2
        if rem(k2,Np)==0
           l=Np/2-1;
        else
           l=rem(k2,Np)-Np/2-1;
        end
        k=ceil(k2/Np)-Np/2-1;
        p=k1-P/4-1;
        alpha=(k-l)/Np+(p-1)/L/P;
        f=(k+l)/2/Np;
        if alpha<-1 | alpha>1
           k2=k2+1;
        elseif f<-.5 | f>.5
           k2=k2+1;
        else
           kk=1+Np*(f+.5);
           ll=1+N*(alpha+1);
           Sx(kk,ll)=M(k1,k2);
        end
    end
end
```