Signal-to-noise-ratio Enhancement

Prediction of harmonic models using AR filters
Program used to obtain Figures (5.1) and (5.2)

FX Processing Codes

```
% A program for linear prediction of harmonic models, 1D example.
                                        % First we prepare a synthetic

dt = 4./1000.; nt =290;
t = 0:dt:(nt-1)*dt;
f = 10;
lf =20;

                                        % Prepare a noise signal

D_in = cos(2*pi*f*t)' + 0.2*randn(nt,1);
D_pred = zeros(nt,1);
C = convmtx(D_in,lf);
d = zeros(nt+lf-1,1); d(1:nt-1) =D_in(2:nt);

                                        % Compute the pef

mu = .001;
f = inv(C'*C+mu*eye(lf))* C'*d;
aux = conv(f,D_in);

                                        % Forward prediction

D_pred(2:nt,1) = aux(1:nt-1);

                                        % Prediction error

E = D_in - D_pred;
```

Signal-to-noise-ratio Enhancement

Canales method for SNR enhancement

FX algorithm, Canales (1984)
Program used to prepare Figures (5.3) and (5.4)

```
% FX.m - needs function ar.m to compute the predicted at a given frequency
clear; clf
dt = 4./1000;                    % Make a synthetic t-x model
w = ricker(20.,dt);
nw=max(size(w));
nx = 32; nt = 128;
DATA = zeros(nx,nt);
for i=1:nx
for j=1:nw
DATA(i,20+j+i) = w(j);
end
end

NOISE = 0.2 * randn(nx,nt);      % Add noise to data
DATA = DATA + NOISE;
p = 20                           %
Length of the pef
DATA_FX = fft(DATA,[],2);        % Go to f-x
for i=1:nt;                      % Do
prediction at each freq
aux_in = DATA_FX(:,i);
aux_out = ar(aux_in,p);
DATA_FX(:,i) = aux_out;
end
DATA = real(ifft(DATA_FX,[],2)); % Back to t-x
subplot(211);wigb(DATA');
title('Input, \sigma_n=0.2')
subplot(212);wigb(DATA');
title('Output after FX filtering, p=20')
```

Signal-to-noise-ratio Enhancement

Linear prediction using AR filters

Function to perform 1D prediction

```
function [D_pred] = ar(D_in,lf);
% 1D prediction function used by Canales' method.
%
% D_in: Data in f-x (one column at freq. f)
% lf: length of the pef (AR)
% D_pred: predicted data (clean data)
%
n = max(size(D_in));
D_pred = zeros(n,1);
C = convmtx(D_in,lf);                          % convolution matrix
d = zeros(n+lf-1,1);                           % RHS vector
(desired output)
d(1:n-1) =D_in(2:n);
mu = .001;
f = inv(C'*C+mu*eye(lf))* C'*d;        % Filter (AR)
aux = conv(f,D_in);                            % apply filter
to data
D_pred(2:n,1) = aux(1:n-1);                % Prediction
return
```

Signal-to-noise-ratio Enhancement

ARMA filtering

Program used to generate Figures (5.10), (5.11), and (5.12)

```
function [s,w,g] = eigen_filtering(y,p,mu);
% Given a 1D noisy sequence y, the order p of
% the ARMA(p,p) model and the regularization parameter mu
% this function computes the clean signal s, an estimate of
% the noise sequence w, and the prediction error filter g.

N = length(y);
Y = convmtx(y,p+1);                % data convolution matrix
R = Y'*Y/N;                                % data correlation matrix
[g,Pw] = eigs(R,1,'SM');           % compute 1 eigenvalue (the SMallest)
                                           % and the associated eigenvector

g0 = g(1);
g = g/g0;
e = Y*g;                                   % non-white sequence e
G = convmtx(g,N);                  % convolution matrix of the filter
D = eye(N)*mu;                     % regularization term
w = inv(G'*G+D)*G'*e;              % estimate of the noise
s = y-w;                                   % estimate of the clean signal
return
```

Signal-to-noise-ratio Enhancement

Practicing – Assignments (Chapter 5)

1. Using the routines and creating your own synthetic data, as shown in previous slides, reproduce figures of Chapter 5. Try to use a real time series to re-run the algorithms.