Some Thoughts on Eigen Filtering
As a method of optimal filtering

Jon Bell
CSIRO ATNF

1 Introduction

The correlation matrix ***R*** of discrete time sampled data is the matrix formed when a vector of such data ***x(n)*** and its transpose are multiplied together:

$$R = \begin{bmatrix} x_0x_0 & x_0x_1 & . & . & x_0x_N \\ x_1x_0 & x_2x_2 & . & . & x_1x_N \\ . & . & x_3x_3 & . & . \\ . & . & . & . & . \\ x_Nx_0 & . & . & . & x_Nx_N \end{bmatrix}$$

This matrix has eigen values λ_i with corresponding eigen vectors $\boldsymbol{q}_i$ such that $(\boldsymbol{R} - \lambda_i \boldsymbol{I})\,\boldsymbol{q}_i = 0$.

Consider the following system

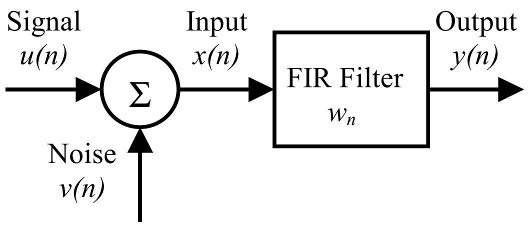

Figure 1: Linear filtering to remove additive white Gaussian noise

The average power of the of the signal component of the output $y(n)$ is $P_0 = \boldsymbol{w}^H\boldsymbol{R}\boldsymbol{w}$, while the average power of the noise component is $N_0 = \sigma^2\boldsymbol{w}^H\boldsymbol{w}$. We want to maximise (find the optimal filter) the output signal to noise ratio $(SNR)_0 = P_0/N_0$ with the constraint that $\boldsymbol{w}^H\boldsymbol{w} = 1$. The coefficients $\boldsymbol{w}_0$ of the impulse response of the optimal FIR filter in this case can be shown to be:

$$\boldsymbol{w}_0 = \boldsymbol{q}_{max}$$

where $\boldsymbol{q}_{max}$ is the eigen vector corresponding to the largest eigen value λ_{max} of the correlation matrix ***R***.

These notes are based on “Adaptive Filter Theory” by Simon Haykin.

2 How well do such eigen filters work ?

Figure 1 shows some synthetic test data u(n) to which random noise v(n) is added. The coefficients of the optimal filter derived as described above are shown in Figure 2, (with the additional step that the correlation matrix was averaged over 10 blocks of noisy data). Curiously this appears to be no more than a scaled version of the average of the signal over the 10 blocks of noisy data. Not surprising the spectral response of this filter looks like a noisy version of the spectral shape of the raw data as shown in Figure 3. Also shown in Figure 3 is spectral response of a smoothed version of this filter and digital filter formed using the Remez algorithm. These spectral responses suggest that the eigen filter is a rather bad filter when compared to the Remez filter. Figures 4 and 5 show the results of apply these two filters to the individual blocks of data, which were subsequently averaged. The result of the eigen filter is a very poor representation of the data. The result of the remez filter is much better (including a substantially lower RMS) when compared to a simple average of the data blocks, however, it does suffer from an increased DC content.

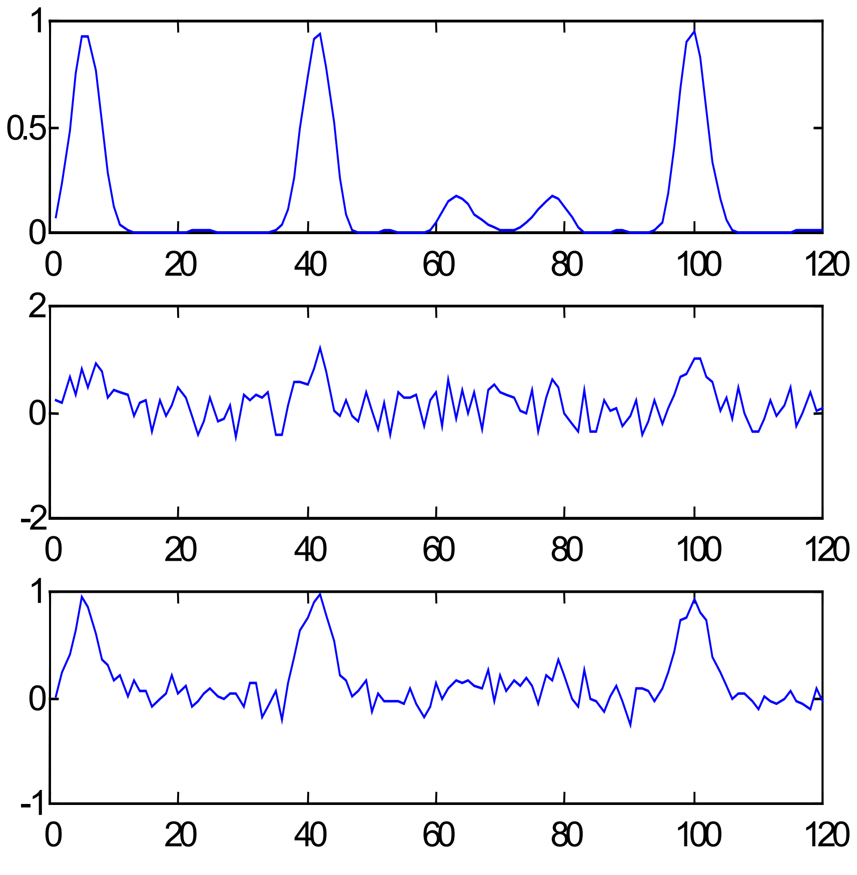

Figure 1: Test data. Top: raw data u(n). Middle: 1 block of noisy input data x(n). Bottom: Average over 10 blocks of noisy input data.

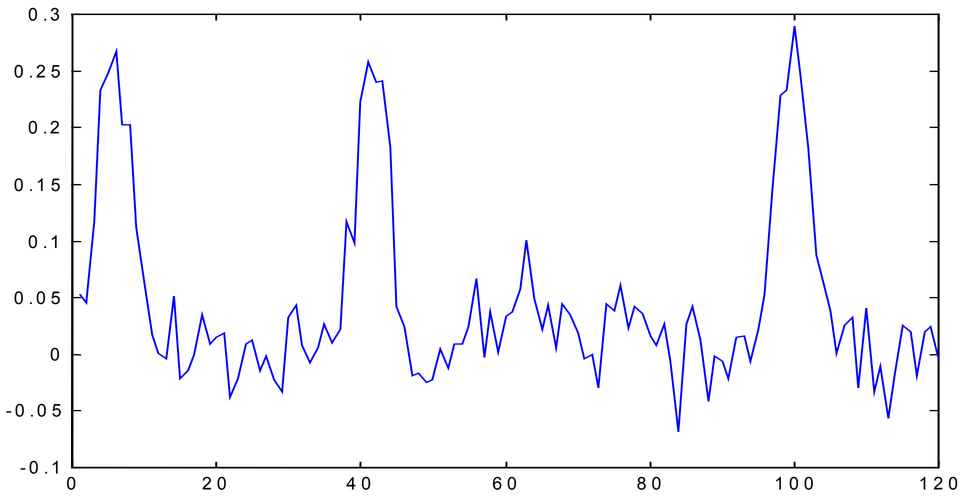

Figure 2: Largest eigen vector of R

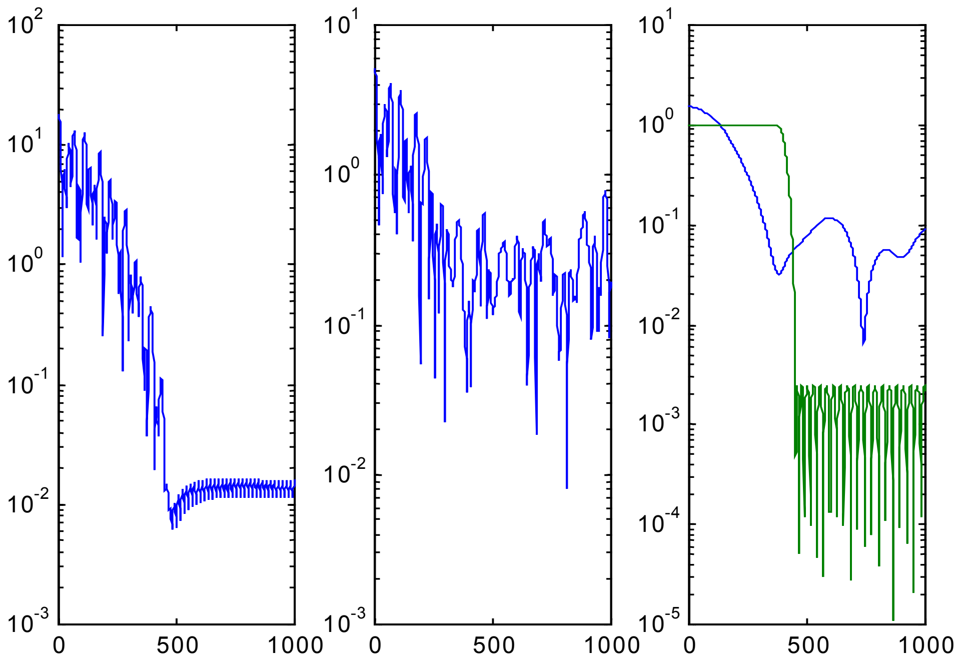

Figure 3: Left: Spectrum of raw data. Middle: Spectral response of FIR filter corresponding to q_{max}. Right: Spectral response of FIR filter when only the first 10% of q_{max} are used as coefficients ans spectral response of a 70th order FIR filter.

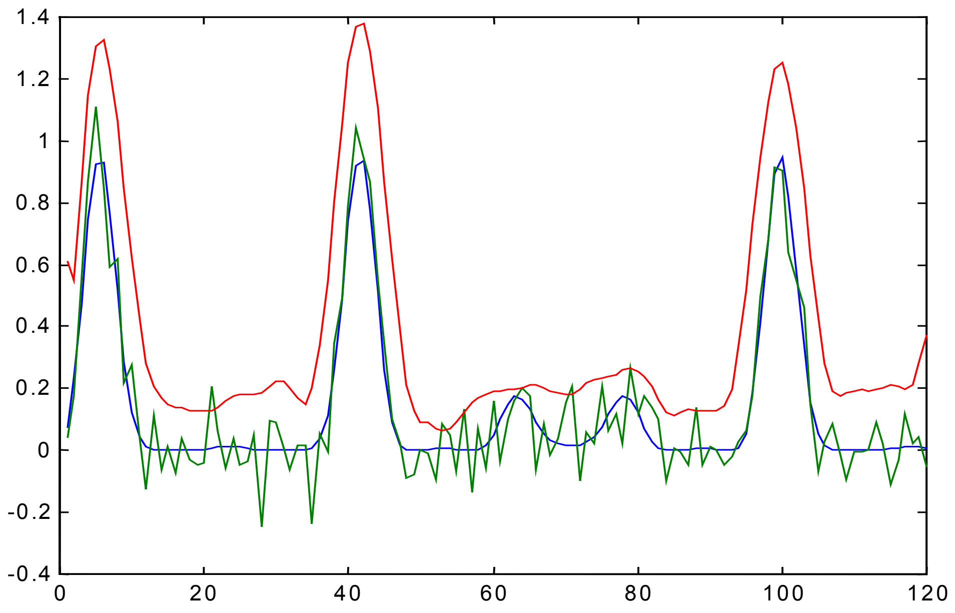

Figure 4: Result of using q_{max} FIR filter, compared to raw data and average over 10 blocks of noisy input data.

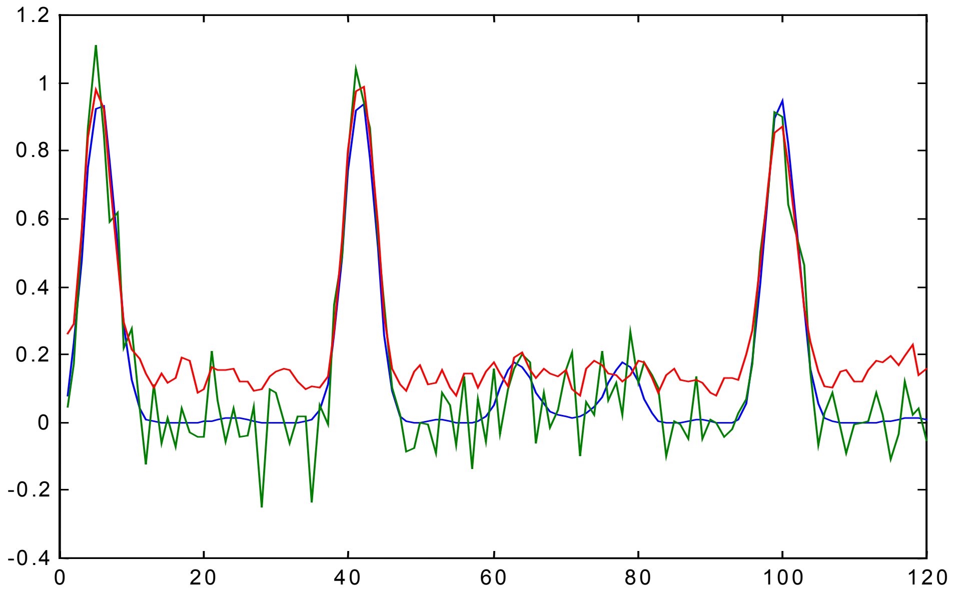

Figure 5: Result of Remez FIR filter compared with raw data and average over 10 blocks of noisy input data.

2 MATLAB code used for the above simulations

```
% Explore the use of eigenanalysis techniques in post detection
% digital signal processing for noise rejection
% Potential application include spectroscopy and imaging
%
% Jon Bell 24 Jul 2000

clear

% Make some fake raw data
nspec = 120;
ntime = 10;
scale = 1.0;
noisescale = 1.0;
dcoff = 0.0;
psi =1.0;
j=1:1:nspec;
k=1:1:ntime;
rawdata = sin(j/3) + sin(j/5) + 1;
rawdata = (rawdata.^4)/70;
ncoef=12; %number of coefficient of the optimal filter to use
%subplot(211)
%plot(rawdata);

% Turn this on for a null test with noise as an input signal
%rawdata=rand(1,nspec);

%Add some random noise to the data and scale it for image colourmap
for i=1:ntime
   noisydata(i,:) = scale*(rawdata + noisescale*(rand(1,nspec)-0.5))
+dcoff;
end

% Make the correlation matrix and decompose it find lambdamax and
qmax
R = zeros(nspec);
for i=1:ntime
   R = R + (noisydata(i,:).'*noisydata(i,:))/ntime;
end
%subplot(311)
%plot(rawdata)
%subplot(312)
%plot(noisydata(1,:))
%subplot(313)
%plot(mean(noisydata))
[VR,D,FLAGS] = eigs(R,1);
wc=0.40;
V=remez(38,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
%plot(j,mean(noisydata),j,VR);
%subplot(131)
%S=abs(fft(rawdata,2048));
%semilogy(S(1:1000));
%subplot(132)
%S=abs(fft(VR,2048));
%semilogy(S(1:1000));
```

```
%subplot(133)
%S=abs(fft(VR(1:ncoef),2048));
%SR=abs(fft(V,2048));
%semilogy([1:1000],S(1:1000),[1:1000],SR(1:1000));
%figure(1)
AR = zeros(ncoef,1); AR(1)=1;
A = zeros(ntime,1); A(1)=1;
for i=1:ntime
   cleanVR(i,:) = abs(filtfilt(VR(1:ncoef),AR,noisydata(i,:)));
   cleanV(i,:) = abs(filtfilt(V,A,noisydata(i,:)));
end
plot(j,rawdata,j,mean(noisydata),j,mean(cleanVR))
figure
plot(j,rawdata,j,mean(noisydata),j,mean(cleanV))
```