Robert Prior
ENEE 499
Advisor:
Dr. Jonathan Simon

Using PCA to Remove Biological Noise from MEG Data

Introduction:

Magnetoencephalography (MEG) is a promising new technique for observing neural activity in the brain. MEG is similar to its older counterpart, electroencephalography (EEG), but differs in focusing on magnetic fields generated by neural currents rather than electrical activity. To detect these fields, a network of 157 sensors is placed on the scalp of a subject. Three others are set up in an orthogonal reference system nearby to capture ambient magnetic fields.

As is apparent from the extremely weak fields in the brain (only a few hundred femtoteslas in most cases), noise in the signal recordings is a significant problem, as even fields of a few picoteslas can completely obscure the desired data. To reduce this noise in the output, a hardware notch filter is used to remove 60 Hz line noise, and Mu-metal shielding (which blocks magnetic fields) is often employed.

A major goal in MEG analysis is to efficiently remove as much noise as possible, without causing loss in signal. One such method, known as Principal Component Analysis (PCA) is highly effective at removing environmental noise, and offers potential for functionality without bulky and expensive shielding (de Cheveigné and Simon, 2006). However, biological noise is inevitable in recordings of the brain. Therefore, it must be processed out. The algorithms for removing environmental noise are very effective, but rely on external reference channels to provide them with the information they require.

External reference channels cannot be non-invasively obtained in a subject, however. If reference channels can be found, the PCA algorithm can be adapted to remove major biological noise sources as well, further expanding its potential as an analysis tool.

Methodology:

Denoising through Reference Channel Synthesis

Two forms of biological noise are most troublesome: heartbeat and eye blinks. Both of these signals occur frequently, and generate high-amplitude spikes in recorded data. Neither signal is regular enough to be removed easily by filtering. Eye blinks are highly erratic, and may occur sporadically, or several times in only a few seconds. Heartbeat cannot be suppressed. Blinking can be suppressed by asking a subject to close his or her eyes. This causes many subjects to fall asleep during the test, which changes brain activity dramatically.

In conscious patients, the areas of the brain that demonstrate the strongest interference from heartbeat and eye blinks are the peripheral regions near the eyes and extending back across the temples toward the ears. The sensors in these areas are directly affected by this noise, and their utility is reduced.

To solve the problem of heartbeat and blink noise in the sensors, reference channels must be created. One method of doing this has been explored with regard to Fast LMS (Ahmar, 2005; Ahmar and Simon, 2005). Synthesis for the modified PCA algorithm used here takes symmetrical pairs of peripheral channels that most strongly showed the effects of biological noise. In terms of the recorded data, these pairs are channels (0, 23), (1, 22) and (2, 21) (see Fig. 1, next page).

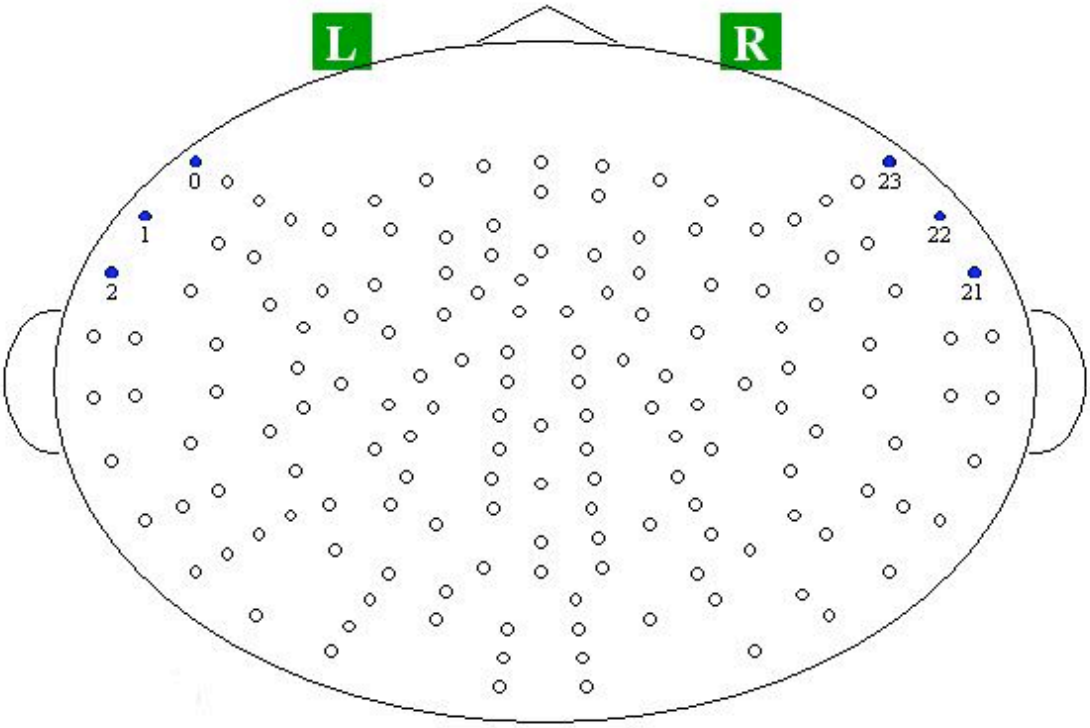

Fig. 1: The sensor network. The biological noise channels are labeled and highlighted. The channels surrounding the highlighted ones are also strongly influenced by the noise.

These pairs were selected to effectively capture blinking and heartbeat, as well as for the approximate symmetry they exhibit, which allows simple mathematical manipulation (see Fig. 2). Due to the overall field characteristics through the human head, field lines will enter one side and exit the other. Because of the symmetry between the channels, the magnitudes of the signals can be added to create a set of reference data vectors which contain the same number of samples as the other channels. This is actually accomplished by subtraction of the pairs (0-23, 1-22, 2-21). This is important for the matrix manipulations performed during PCA. The PCA algorithm itself must then be modified to generate these synthetic reference channels and employ them in place of the

external channels. When executed, the algorithm will treat the noise-heavy peripheral channels as reference, and remove biological noise as though it were external.

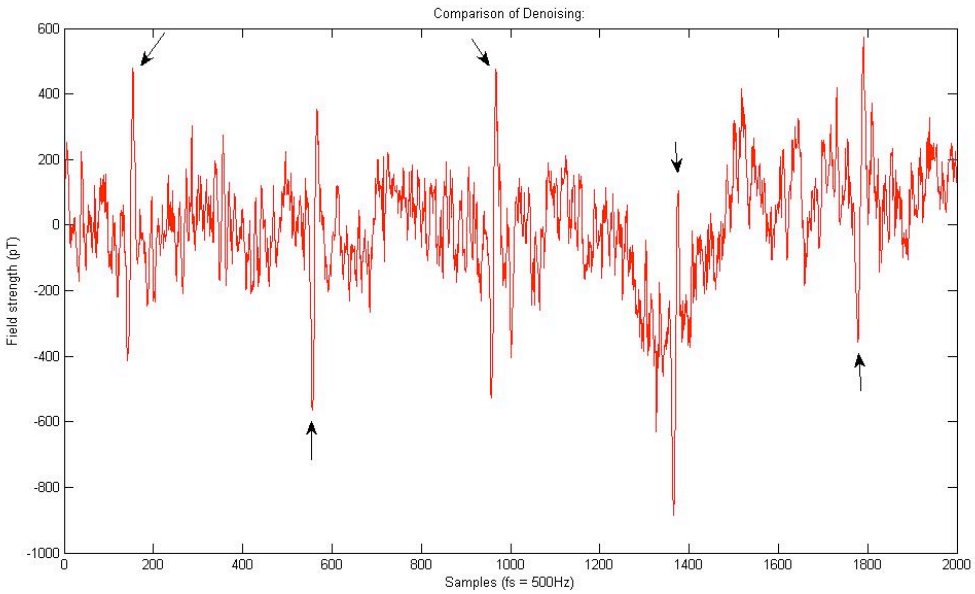

Fig. 2.a: This is a sample of the test MEG file used to develop the modified PCA denoising algorithm. The plot shows four seconds of reference data from channel 0, after PCA was applied once to remove environmental noise. The arrows label the strong peaks typical of heartbeat and eye blinks in peripheral channels. Channel 23 (paired with 0) demonstrates a very similar pattern, reversed in amplitude.

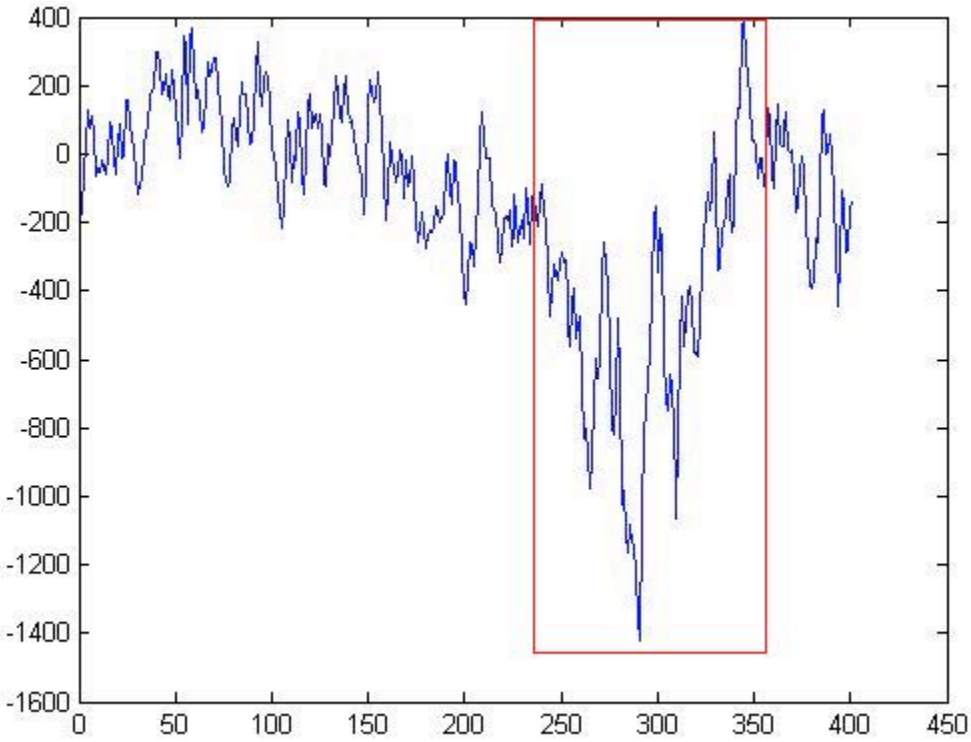

Fig. 2.b: This plot shows .8 seconds from a PCA-denoised version of the test file (biological noise present, environment filtered). The axes are samples (X-axis) and field amplitude (Y-axis) as above. The red box outlines an eye blink. The disruption of data lasts approximately .2 seconds. This pattern must be effectively processed so that signal can be recovered.

Synthesizing Additional Reference Channels

The effectiveness of PCA scales with the number of reference channels. Because the reference channel synthesis method is not hardware limited to three channels, more can be synthesized if desired. For example, the subtracted pairs (0-23), (1-22), (2-21) can be used with addition pairs, for a total of six reference channels (New channels being (0+23), (1+22), (2 +21)). This exploits the increased power of PCA without requiring

designation of more data channels as reference, allowing for a more aggressive noise reduction scheme than the three-channel denoising.

Because this method effectively exceeds the hardware limitations the PCA was originally intended to observe, a few careful modifications are required over the three-channel denoising discussed above. The algorithm must be adjusted to include the additional channels in its reference block. Also important are modifications to the output file generation routine, so that all the correct channels are written into the output.

Results:

Synthesis of Three Reference Channels

When the modified PCA is applied to remove biological noise, the reduction of heartbeat and eye blink pulses is substantial. The initial PCA pass removes approximately 80% of the power within the signal. The subsequent pass to remove biological noise eliminates approximately 23% of the remaining power, and with it, much of the heartbeat and eye blink pulses. The results of the biological denoising are encouraging, as the final result generally tracks the first denoise very closely, except where a peak due to noise has been removed or reduced (see Fig. 3, below). These characteristics are most notable in the peripheral channels.

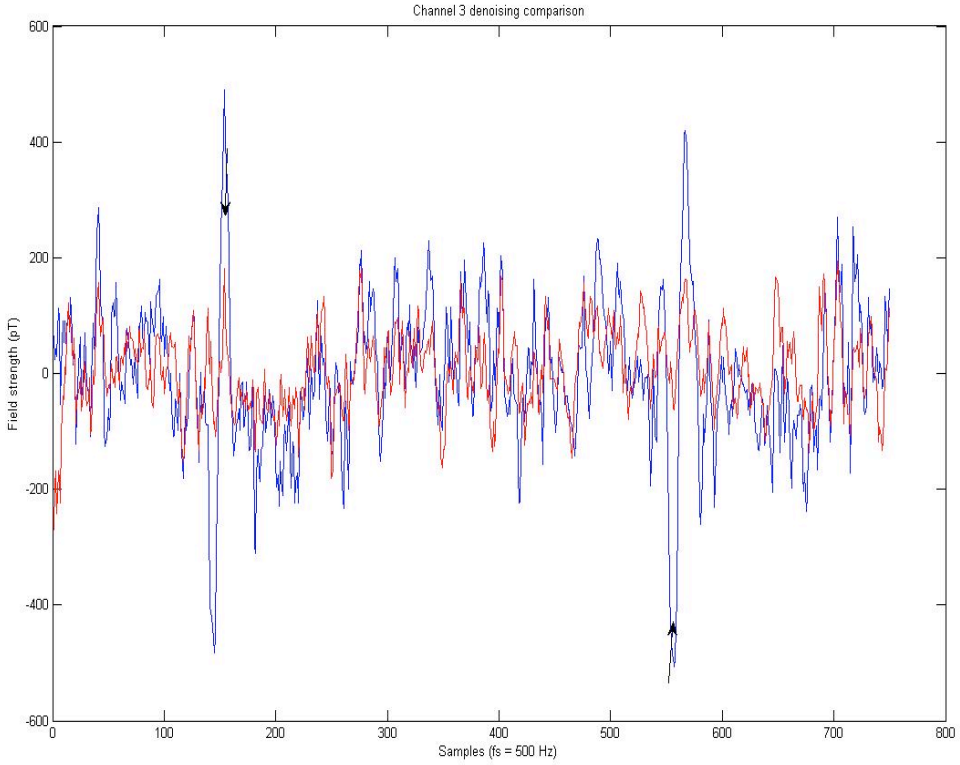

Fig. 3: 1.5 seconds of denoised data from channel 3 (in the peripheral region). The blue plot represents PCA. The red shows the subsequent application of modified PCA. The character of the signals is the same, but noise spikes at heartbeats and other biological noise sources are reduced, and in some cases, removed altogether.

Not all the results of this approach are ideal, however. Upon close examination, the results of the biological denoising occasionally show what appears to be a small shift of a few samples between peaks of the two signals (fig. 4). Considering the space between the two sensors in each pair, and the fact that the sensor network on the head is not perfectly symmetrical, it does make sense that the same information may show up at a slightly different time in different channels.

One other factor that is cause for some concern is the potential for loss of signal through the denoising process. In tests, the percentage of power removed by the second denoise was approximately 23% of what was present beforehand. This suggests the possibility of some signal loss, possibly due to the shifting behavior of the modified PCA with only three channels. This can be addressed by the addition of the counterbalancing addition channels.

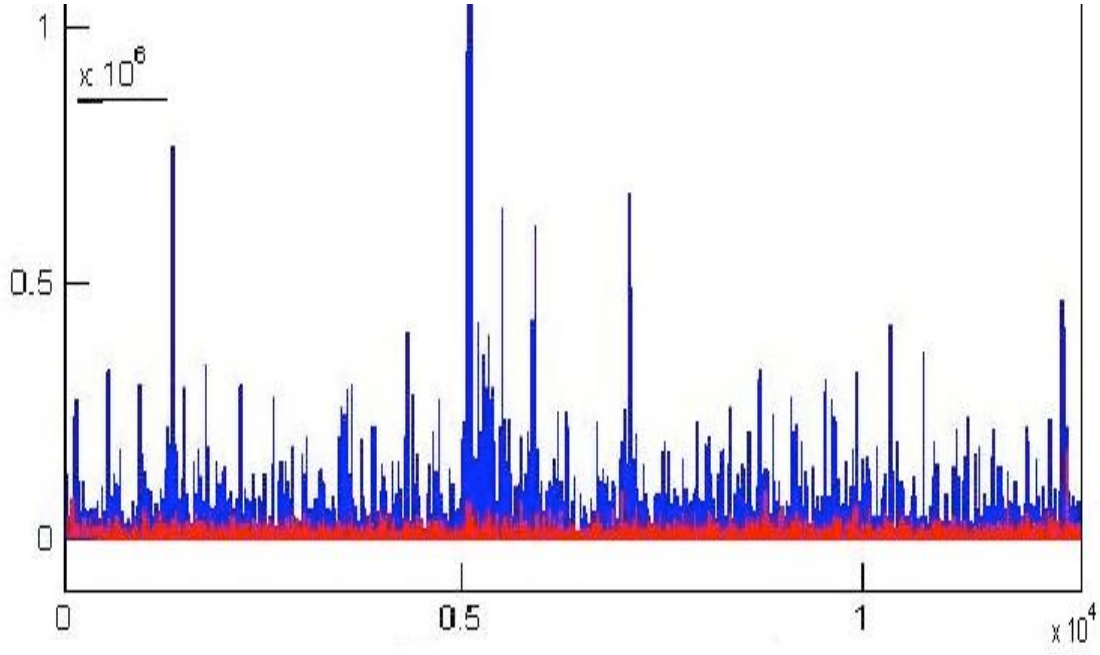

Fig. 4.a: This is an amplitude-squared plot of channel 3 in the test file. The amplitude squared axis is truncated to better contrast the environmental denoise (blue) and the data after biological noise is removed (shown in red). Maximum environmental amplitude is actually approximately 2.5 x10^6 near sample 5000.

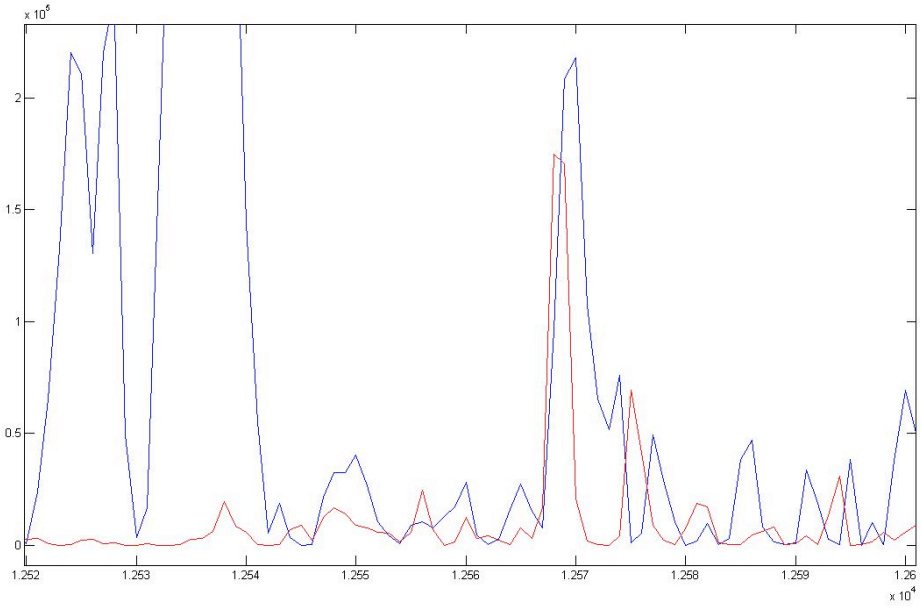

Fig. 4.b: This shows a magnification of the denoise/biological denoise plot. The effectiveness of the second denoise can be seen in most peaks, along with some of its less-effective behavior in and around the central peak. Also, the shifting problem discussed above can be clearly seen in the central peak and a few others.

Denoising with Additional Synthesized Channels

Denoising that uses six reference channels synthesized from the channels already selected yields an algorithm that is extremely aggressive and effective at removing noise. This algorithm removes 34.5% of the power remaining after environmental denoising, opposed to 23% for three channels. The three subtraction-based reference channels combined with three addition-based channels from the same pairs reduces the time shift seen in the three-channel case. The comparison of performance between three- and six-channel algorithms can be seen by plotting the signal that remains, below (Fig. 5). The more aggressive denoising brings signal range down to levels expected of magnetic fields in the brain, with minimal distortion.

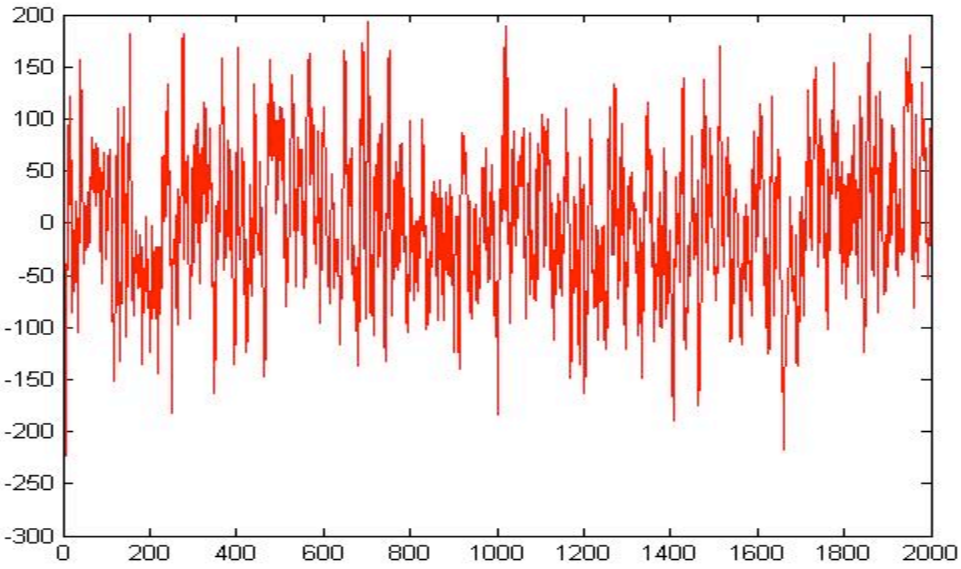

Fig. 5.a: Signal plot for three-channel biological denoising; the plot shows four seconds of the denoised file. This is a fairly clean signal, covering an amplitude range of only about 350 pT. Fig. 3 contains the first 750 samples of this plot, and is therefore useful for putting these plots in context.

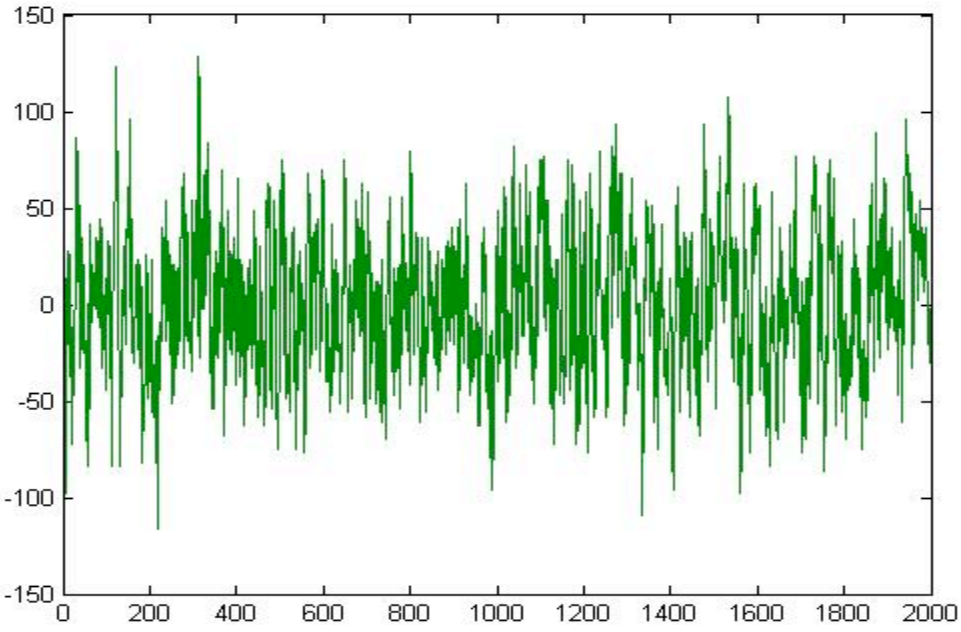

Fig. 5.b: Signal plot for six-channel biological denoising; the plot shows four seconds of the denoised file. The signal is thoroughly denoised, covering an amplitude range of 150 pT or less.

Conclusion:

Modifying the PCA algorithm and generating artificial reference channels based on biological noise-heavy channels is a powerful method. It is highly successful at removing biological noise after environmental noise has been eliminated. Heartbeat and eye blinks can be almost completely removed from MEG data by this method. It is an entirely software-based approach, meaning that no new hardware is needed, and exploits the power of PCA to accomplish a goal where classical measurement is difficult at best. The ability to synthesize and add other reference channels without hardware limitations also makes modification to the algorithm less difficult. The denoising can be readily tailored to fit multiple levels of denoising rigor, as desired. In particular, the effects of adding the pair-addition channels to balance the effects of the subtraction are excellent, as this resolves or reduces many of the problems that arise with this method, such as time shifting between the original and resulting files. As more aggression is demanded of the algorithm, more noise is removed, at the cost of a small amount of signal that will be removed as well, due to random correlation with the reference data.

Works Cited:

Ahmar, Nayef. "Da Vinci's Encephalogram: In Search of Significant Brain Signals". 2005. Graduate thesis. Pages 35-49.

Ahmar, Nayef and Jonathan Z. Simon. "MEG Adaptive Noise Suppression using Fast LMS". Neural Engineering, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on. March 16-19, 2005 Pages:29 – 32

De Cheveigné, Alain and Jonathan Z. Simon. "Magnetoencephalography Denoising based on Time-Shift PCA". September, 2006. Submitted for publication.

Appendix: MATLAB code used for the modified PCA algorithms:

These MATLAB functions are both heavily based on the PCA algorithm of Drs. Simon and de Cheveigné.

Three-channel denoising:

```
function denoise_mod3(file1, file2, ntaps, noisy)
% denoise - suppress noise correlated with reference channels from MEG data
%
%   denoise(file1, file2, [ntaps], [noisy]) denoises data from file1 and puts it into file2.
%   file2 is squashed.
%
%   ntaps: number of taps to apply to ref channels
%   noisy: list of noisy channel numbers to set to zero
%
%  Alain de Cheveigne
%  CNRS / ENS / Universite Paris 5
%  and Shamma lab, ISR, University of Maryland
%  November 2005.


%Last modified 11/27/06 by Bob Prior
%Changes:
% Modified ref definition.
% PCA section should behave as though it has 3 ref channels.


%chunksize=50000; % samples, size of processing chunks
chunksize=10000; % samples, size of processing chunks
boost=10000;   % factor to boost reference channels

verbose = 1;
showplots = 0;

if nargin<4; noisy=[]; end
if nargin<3; ntaps=200; end
if nargin<2; help denoise; return; end
if strcmp(file1,file2); error('files should be distinct'); end
if exist ([pwd, filesep, file2], 'file'); delete (file2); end

info=sqdread(file1, 'info');
nsamples=get(info, 'SamplesAvailable');
channelcount=get(info, 'ChannelCount');
ndata=157; % number of data channels
nref=3;    % number of noise reference channels
if mod(ntaps,2)
   ntaps=ntaps+1; % ensure ntaps even
   disp(['warning: setting ntaps=',num2str(ntaps)]);
end

% The idea is to rotate the data such that the first dimensions are colinear
% with the subspace spanned by the reference channels and all filtered
```

```
% versions of the ref channels.

% This is done by (a) time-shifting the reference channels over a range of shifts
% (positive and negative), (b) boosting the amplitude of these shifted reference
% channels, (c) performing a PCA with these and the data channels, and (d)
% discarding the first nrefs*ntaps PCA components.
% The remaining components are orthogonal to this subspace, ie they are stripped
% of environmental noise common to all sensors.  They are then rotated back
% to sensor space.


% first pass: covariance matrix
nchunks=ceil(nsamples/chunksize);
R=zeros(ndata+nref*ntaps);
if verbose; tic; disp('Scan file to calculate covariance matrix...'); end
for k=0:nchunks-1
   from=k*chunksize+1;
   to=min(from+chunksize-1,nsamples);
   x=xsqdread(file1, [from-ntaps/2 to+ntaps/2], (1:160)-1, nsamples);

   ref=x(:,1:3)-x(:,24:-1:22);    % reference channels
   data=x(:,1:157);               % data channels
   for c=noisy; data(:,c)=0; end  % set noisy channels to zero

   n=to-from+1;
   shifts=ones(n,ntaps);
   shifts=cumsum(shifts)+cumsum(shifts,2)-1; % shift indices
   yy=zeros(n,nref*ntaps);
   %yy=[];
   for k=0:nref-1
      y=ref(:,k+1);
     % yy=[yy,y(shifts)];
      yy(:,k*ntaps+1: k*ntaps+ntaps)=y(shifts);
   end
   data=data(ntaps/2+1:ntaps/2+n,:);
   z=[data,boost*yy];                  % append boosted ref channels
   z=z - ones(size(z,1),1)*mean(z);    % remove mean
   R=R+z'*z;                           % accumulate into covariance matrix
   if verbose; fprintf(1,'.'); end
end
R=R/nsamples;
if verbose; disp(' done'); toc; end

% PCA matrix
[V, S] = eig(R) ;
[eigenvalues, idx] = sort(diag(S)') ;
idx = fliplr(idx) ;
eigenvectors = V(:,idx);
clear R x y yy ref z data;

% clip matrix to discard noise components
eigenvectors=eigenvectors(:,nref*ntaps+1:end);

% second pass: rotate to principal components
e1=0; e2=0;
if verbose; disp('Rescan file to rotate to principal components...'); end
```

```
for k=0:nchunks-1
   from=k*chunksize+1;
   to=min(from+chunksize-1,nsamples);
   x=xsqdread(file1, [from-ntaps/2 to+ntaps/2], (1:192)-1, nsamples);
  %Generate biological noise signals from noisy peripheral channels.
   %x(:,161) = x(:,1) - x(:,24); %Indeces start at 1.
   %x(:,162) = x(:,2) - x(:,23);
   %x(:,163) = x(:,3) - x(:,22);
   %x(:,164) = x(:,4) - x(:,21);
   %x(:,165) = x(:,5) - x(:,20);

   ref=x(:,1:3)-x(:,24:-1:22);         % reference channels
   junk=x(:,161:end);        % additional channels
   data=x(:,1:157);             % data
   for c=noisy; data(:,c)=0; end  % set noisy channels to zero

   n=to-from+1;
   shifts=ones(n,ntaps);
   shifts=cumsum(shifts)+cumsum(shifts,2)-1; % shift indices
   yy=zeros(n,nref*ntaps);
   %yy=[];
   for k=0:nref-1
      y=ref(:,k+1);
     % yy=[yy,y(shifts)];
      yy(:,k*ntaps+1: k*ntaps+ntaps)=y(shifts);
   end
   clear y;
   e1=e1+sum(sum(data.^2));     % variance before
   data=data(ntaps/2+1:ntaps/2+n,:);
   z=[data,boost*yy];                      % append boosted ref channels
   z=z - ones(size(z,1),1)*mean(z);      % remove mean
   y=(z*eigenvectors*eigenvectors');     % rotate to pcs and back
   y=y(:,1:157);                         % clip to remove boosted channels

   e2=e2+sum(sum(y.^2));     % variance after

   if showplots;
      z=z(:,1:157);
      subplot 231
      draw_topo_fast((sum(z.^2).^.5));
      title('before');
      subplot 234;
      draw_topo_fast(sum(y.^2).^.5);
      title('after');
      subplot 432;
      plot([z(1:600,1), y(1:600,1), y(1:600,1)]); title('sensor 1 (blue: before, red: after)'); xlabel('samples');
      subplot 435;
      plot([z(1:600,2), y(1:600,2), y(1:600,2)]); title('sensor 2'); xlabel('samples');
      subplot 438;
      plot([z(1:600,3), y(1:600,3), y(1:600,3)]); title('sensor 3'); xlabel('samples');
      subplot (4,3,11);
      plot([z(1:600,4), y(1:600,4), y(1:600,4)]); title('sensor 4'); xlabel('samples');
      set(gcf,  'name', 'denoise')

      subplot (4,3,3), pwelch(z(:,1),[],[],[],500)
      subplot (4,3,6), pwelch(y(:,1),[],[],[],500)
```

```
      drawnow;
   end

   ref=ref(ntaps/2+1:ntaps/2+n,:);
   junk=junk(ntaps/2+1:ntaps/2+n,:);
   y=[y,ref,junk];
   sqdwrite(file1, file2, 'Action', 'Append', 'Data', y);
   if verbose; fprintf(1,'.');  end
end

if verbose
   disp(' done'); toc;
   disp(['Discarded ', num2str((e1-e2)/e1*100), ' % of data channel variance.']);
end

% extended read function returns zeros if 'samples' out of bounds
function x=xsqdread(file, samples, channels, nsamples)
from=samples(1);
to=samples(2);
x=sqdread(file, 'samples', [max(from,1) min(to,nsamples)], 'channels', channels);
if from<1; x=[zeros(1-from, size(channels,2)); x]; end
if to>nsamples; x=[x; zeros(to-nsamples, size(channels,2))]; end
```

Six-Channel Addition/Subtraction Denoising:

```
function denoise_mod4(file1, file2, ntaps, noisy)
% denoise - suppress noise correlated with reference channels from MEG data
%
%   denoise(file1, file2, [ntaps], [noisy]) denoises data from file1 and puts it into file2.
%   file2 is squashed.
%
%   ntaps: number of taps to apply to ref channels
%   noisy: list of noisy channel numbers to set to zero
%
%  Alain de Cheveigne
%  CNRS / ENS / Universite Paris 5
%  and Shamma lab, ISR, University of Maryland
%  November 2005.


%Last modified 12/15/06 by Bob Prior
%Changes:
% Modified ref definition, data output code.
% PCA section should behave as though it has 6 ref channels.


%chunksize=50000; % samples, size of processing chunks
chunksize=10000; % samples, size of processing chunks
boost=10000;   % factor to boost reference channels

verbose = 1;
showplots = 0;

if nargin<4; noisy=[]; end
if nargin<3; ntaps=200; end
if nargin<2; help denoise; return; end
if strcmp(file1,file2); error('files should be distinct'); end
if exist ([pwd, filesep, file2], 'file'); delete (file2); end

info=sqdread(file1, 'info');
nsamples=get(info, 'SamplesAvailable');
channelcount=get(info, 'ChannelCount');
ndata=157; % number of data channels
nref=6;    % number of noise reference channels
if mod(ntaps,2)
   ntaps=ntaps+1; % ensure ntaps even
   disp(['warning: setting ntaps=',num2str(ntaps)]);
end

% The idea is to rotate the data such that the first dimensions are colinear
% with the subspace spanned by the reference channels and all filtered
% versions of the ref channels.

% This is done by (a) time-shifting the reference channels over a range of shifts
% (positive and negative), (b) boosting the amplitude of these shifted reference
% channels, (c) performing a PCA with these and the data channels, and (d)
```

```
% discarding the first nrefs*ntaps PCA components.
% The remaining components are orthogonal to this subspace, ie they are stripped
% of environmental noise common to all sensors.  They are then rotated back
% to sensor space.


% first pass: covariance matrix
nchunks=ceil(nsamples/chunksize);
R=zeros(ndata+nref*ntaps);
if verbose; tic; disp('Scan file to calculate covariance matrix...'); end
for k=0:nchunks-1
   from=k*chunksize+1;
   to=min(from+chunksize-1,nsamples);
   x=xsqdread(file1, [from-ntaps/2 to+ntaps/2], (1:160)-1, nsamples);

   ref=[x(:,1:3)-x(:,24:-1:22) x(:,1:3) + x(:,24:-1:22)];    % reference channels
   data=x(:,1:157);              % data channels
   for c=noisy; data(:,c)=0; end  % set noisy channels to zero

   n=to-from+1;
   shifts=ones(n,ntaps);
   shifts=cumsum(shifts)+cumsum(shifts,2)-1; % shift indices
   yy=zeros(n,nref*ntaps);
   %yy=[];
   for k=0:nref-1
      y=ref(:,k+1);
     % yy=[yy,y(shifts)];
      yy(:,k*ntaps+1: k*ntaps+ntaps)=y(shifts);
   end
   data=data(ntaps/2+1:ntaps/2+n,:);
   z=[data,boost*yy];                  % append boosted ref channels
   z=z - ones(size(z,1),1)*mean(z);    % remove mean
   R=R+z'*z;                           % accumulate into covariance matrix
   if verbose; fprintf(1,'.'); end
end
R=R/nsamples;
if verbose; disp(' done'); toc; end

% PCA matrix
[V, S] = eig(R) ;
[eigenvalues, idx] = sort(diag(S)') ;
idx = fliplr(idx) ;
eigenvectors = V(:,idx);
clear R x y yy ref z data;

% clip matrix to discard noise components
eigenvectors=eigenvectors(:,nref*ntaps+1:end);

% second pass: rotate to principal components
e1=0; e2=0;
if verbose; disp('Rescan file to rotate to principal components...'); end
for k=0:nchunks-1
   from=k*chunksize+1;
   to=min(from+chunksize-1,nsamples);
   x=xsqdread(file1, [from-ntaps/2 to+ntaps/2], (1:192)-1, nsamples);
  %Generate biological noise signals from noisy peripheral channels.
```

```
  %x(:,161) = x(:,1) - x(:,24); %Indeces start at 1.
  %x(:,162) = x(:,2) - x(:,23);
  %x(:,163) = x(:,3) - x(:,22);
  %x(:,164) = x(:,4) - x(:,21);
  %x(:,165) = x(:,5) - x(:,20);

  ref=[x(:,1:3)-x(:,24:-1:22) x(:,1:3) + x(:,24:-1:22)];         % reference channels
%   junk=x(:,161:end);        % additional channels
  junk=x(:,158:end);        % additional channels
  data=x(:,1:157);             % data
  for c=noisy; data(:,c)=0; end  % set noisy channels to zero

  n=to-from+1;
  shifts=ones(n,ntaps);
  shifts=cumsum(shifts)+cumsum(shifts,2)-1; % shift indices
  yy=zeros(n,nref*ntaps);
  %yy=[];
  for k=0:nref-1
     y=ref(:,k+1);
    % yy=[yy,y(shifts)];
     yy(:,k*ntaps+1: k*ntaps+ntaps)=y(shifts);
  end
  clear y;
  e1=e1+sum(sum(data.^2));     % variance before
  data=data(ntaps/2+1:ntaps/2+n,:);
  z=[data,boost*yy];                    % append boosted ref channels
  z=z - ones(size(z,1),1)*mean(z);      % remove mean
  y=(z*eigenvectors*eigenvectors');     % rotate to pcs and back
  y=y(:,1:157);                         % clip to remove boosted channels

  e2=e2+sum(sum(y.^2));     % variance after

  if showplots;
     z=z(:,1:157);
     subplot 231
     draw_topo_fast((sum(z.^2).^.5));
     title('before');
     subplot 234;
     draw_topo_fast(sum(y.^2).^.5);
     title('after');
     subplot 432;
     plot([z(1:600,1), y(1:600,1), y(1:600,1)]); title('sensor 1 (blue: before, red: after)'); xlabel('samples');
     subplot 435;
     plot([z(1:600,2), y(1:600,2), y(1:600,2)]); title('sensor 2'); xlabel('samples');
     subplot 438;
     plot([z(1:600,3), y(1:600,3), y(1:600,3)]); title('sensor 3'); xlabel('samples');
     subplot (4,3,11);
     plot([z(1:600,4), y(1:600,4), y(1:600,4)]); title('sensor 4'); xlabel('samples');
     set(gcf,  'name', 'denoise')

     subplot (4,3,3), pwelch(z(:,1),[],[],[],500)
     subplot (4,3,6), pwelch(y(:,1),[],[],[],500)
     drawnow;
  end

  ref=ref(ntaps/2+1:ntaps/2+n,:);
```

```
   junk=junk(ntaps/2+1:ntaps/2+n,:);
   % y=[y,ref,junk];
   y=[y,junk];
   sqdwrite(file1, file2, 'Action', 'Append', 'Data', y);
   if verbose; fprintf(1,'.');  end
end

if verbose
   disp(' done'); toc;
   disp(['Discarded ', num2str((e1-e2)/e1*100), ' % of data channel variance.']);
end

% extended read function returns zeros if 'samples' out of bounds
function x=xsqdread(file, samples, channels, nsamples)
from=samples(1);
to=samples(2);
x=sqdread(file, 'samples', [max(from,1) min(to,nsamples)], 'channels', channels);
if from<1; x=[zeros(1-from, size(channels,2)); x]; end
if to>nsamples; x=[x; zeros(to-nsamples, size(channels,2))]; end
```