Independent Component Analysis
Assignment 4 - Project
FastICA and AMUSE in the context of ECG

Tom Andersen
`elt15tan@student.lu.se`

February 10, 2021

1 Introduction

In this project attempts were made to separate the ECG of a pregnant mother and her baby using ICA.

The measurements consisted of 8 observations (time series) which included the baby's and the mother's ECG together with some unknown noise sources.

Two different methods were considered: AMUSE and FastICA, which both are algorithms used to separate independent sources in a linear mix. PCA was also used as a pre-processing step before any ICA was done.

2 Method

The data was given in a 9x2500 matrix in MATLAB. The first row represented the time stamps for the corresponding data which was located in row 2–9. As a pre-processing step before PCA the mean was removed from the data. The time dimension (row 1) was also removed and stored.

```
X = load("data/foetal_ecg.dat").';
meanX = mean(X,2);
X = X - meanX;

time = X(1,:);
X = X(2:end,:);
```

2.1 PCA

The given data consisted of 8 observations of the mixed ECGs. PCA using eigenvector decomposition was used to reduce the abundant information of the 8 dimensional data to data of lower dimension.

PCA was done in MATLAB using the built-in function `eigs(·)` together with simple orthogonal projection.

```
Rxx_hat = cov(X.');
[PC, eig_values] = eigs(Rxx_hat, eig_count);
Z = (X.'*PC).';
```

`X` represent the 8-dimensional observations, and `Z` is the data after PCA. Different values of `eig_count` was tested, however only in the range between 2 and 5.

2.2 ICA using AMUSE

The AMUSE algorithm was implemented in MATLAB code as below.

```
Ctao1 = (Z*delayseq(Z',1))./length(Z);
Ctao1_ = 0.5.*(Ctao1 + Ctao1.');
[W1, D1] = eig(Ctao1_);
W1 = W1';
Y1 = W1*Z;
```

`Z` represent the 8-dimensional observations after PCA, and `W1` the separation matrix which was used to separate the sources.

2.3 ICA using FastICA

The FastICA algorithm was implemented in MATLAB code using the deflationary version. The `fastICA(·)` function was used to separate the sources in `Z`, the pre-processed observations.

```
Y1 = fastICA(Z)
```

For the sake of fluency the implementation of the `fastICA(·)` function is omitted and instead appended to the appendix, see section 6.1.

2.4 Calculating heart rate directly from observations

The observations was clean enough for heart rate to be observed without any processing. This was done by measuring the time between 10 of the main spikes (10 R-to-R intervals) in the ECG—the main spike (R-wave) of the so called QRS complex—for both the mother's and the baby's ECG. From this the heart rate could be calculated.

This information was mostly used as a verification after the ICA was done.

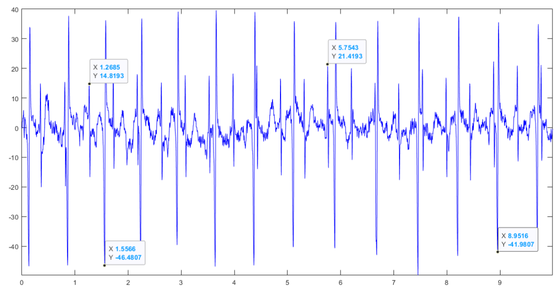

Figure 1: Observation of index 1 plotted alone. The approximate frequency of the heart rate can be calculated.

3 Results

As the reader might discover, only some of the results are of interest. However, for the sake of consistency all the results are presented below.

3.1 Heart rate and observations

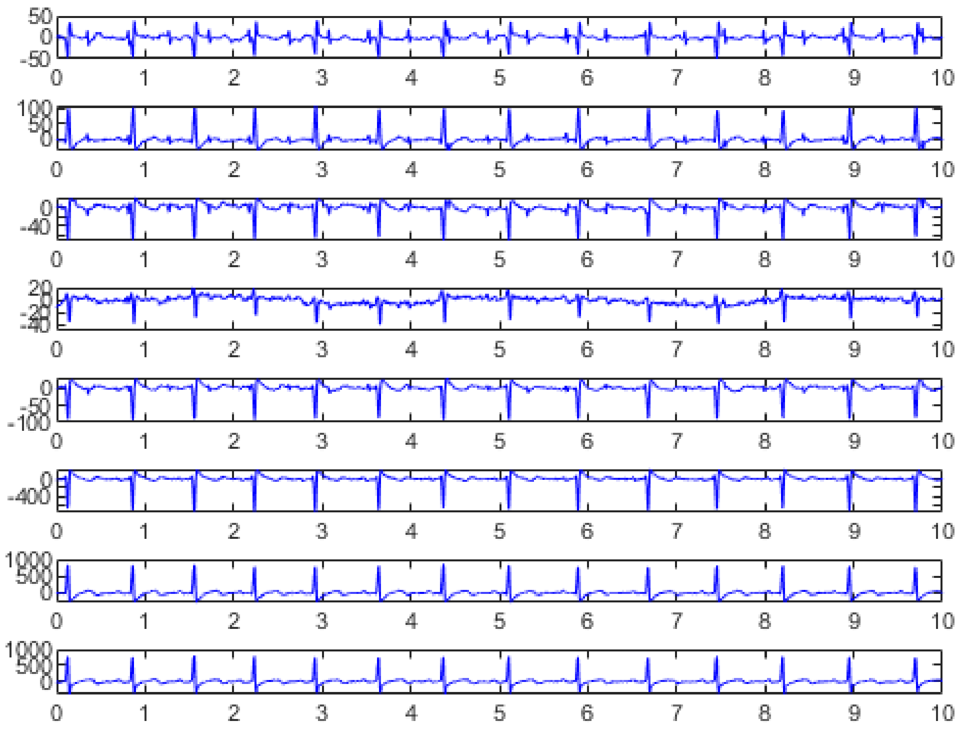

Figure 2: Raw data. The 8 observations plotted together.

Assumed source	$10 \cdot T_{R-R}$	R-spike frequency $\frac{1}{T_{R-R}}$	Heart rate (bpm)
Mother	7.395 s	1.35 Hz	81
Baby	4.4858 s	2.23 Hz	134

Table 1: Rate of heart beat calculated by hand using the time of 10 R-to-R intervals.

3.2 Observations after PCA

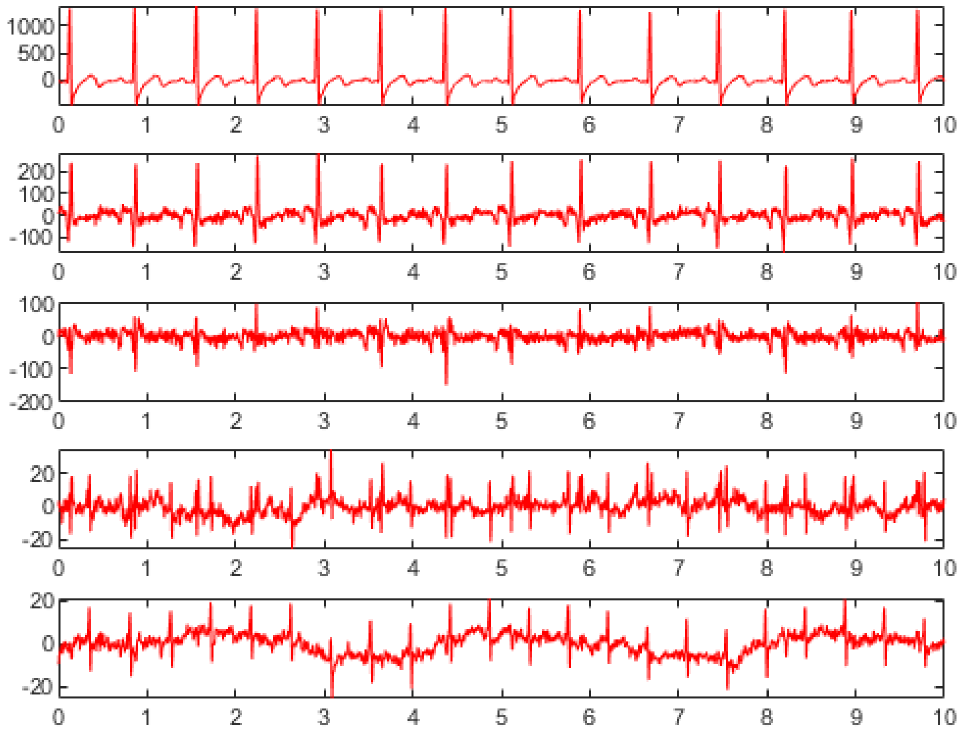

Figure 3: Observations after PCA using eigenvectors corresponding to the 5 largest eigenvalues.

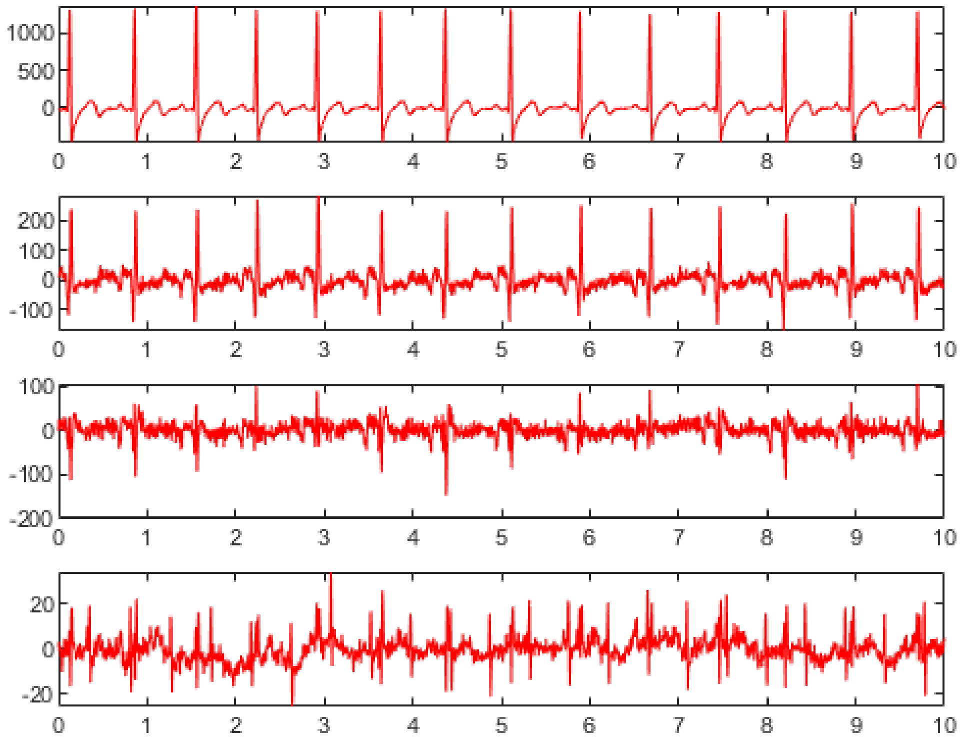

Figure 4: Observations after PCA using eigenvectors corresponding to the 4 largest eigenvalues.

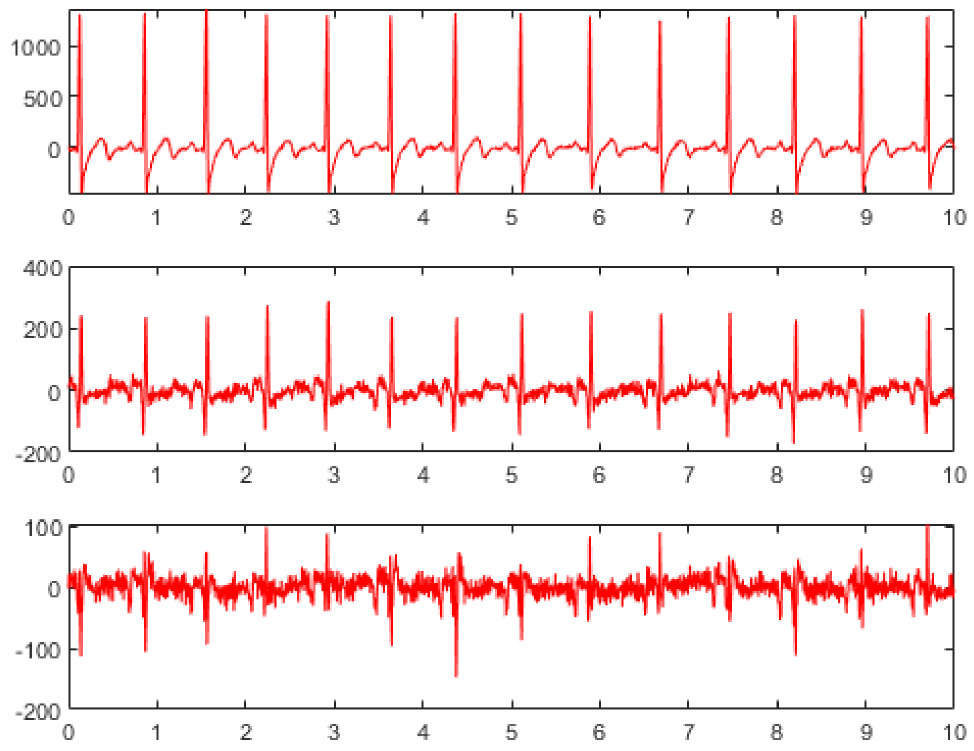

Figure 5: Observations after PCA using eigenvectors corresponding to the 3 largest eigenvalues.

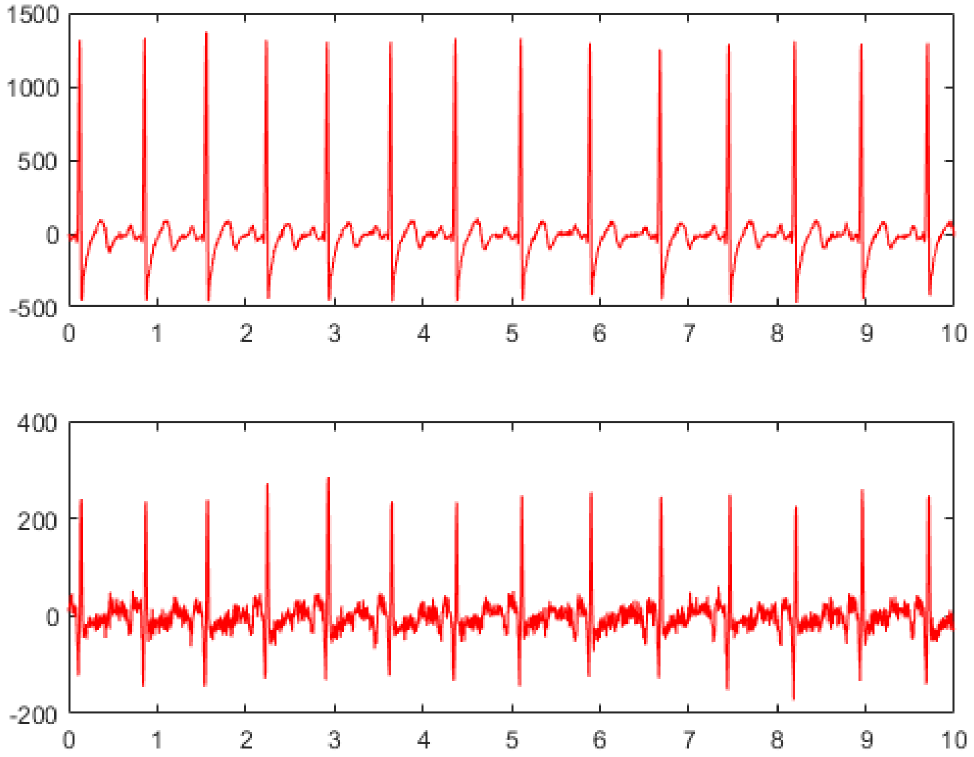

Figure 6: Observations after PCA using eigenvectors corresponding to the 2 largest eigenvalues.

3.3 Separation using AMUSE

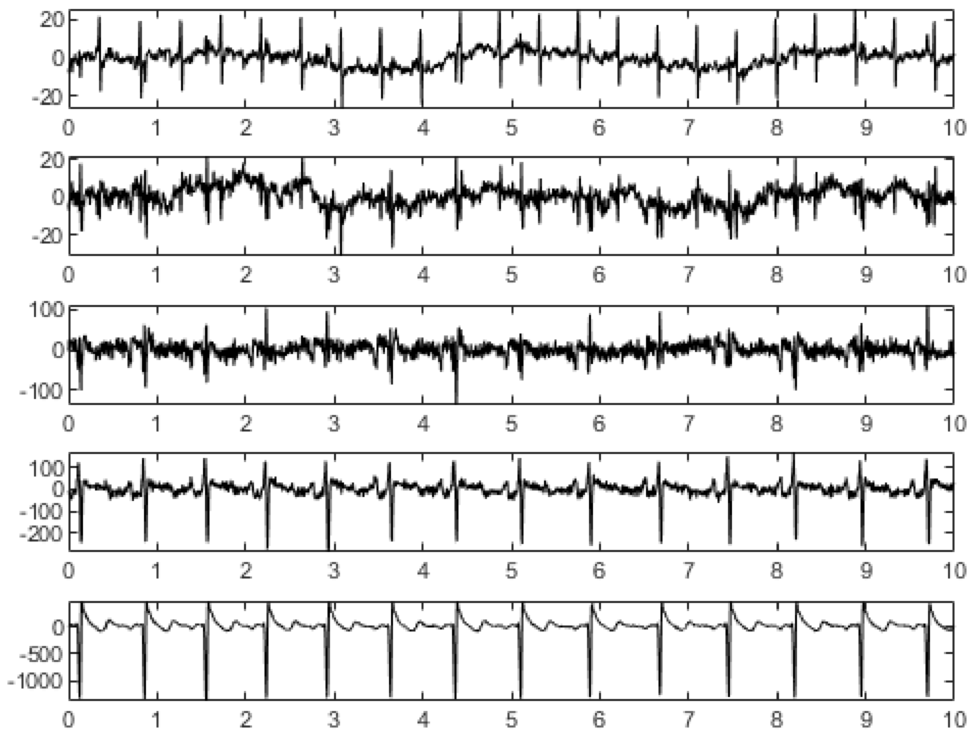

Figure 7: Separated sources using AMUSE algorithm after PCA using eigenvectors corresponding to the 5 largest eigenvalues.

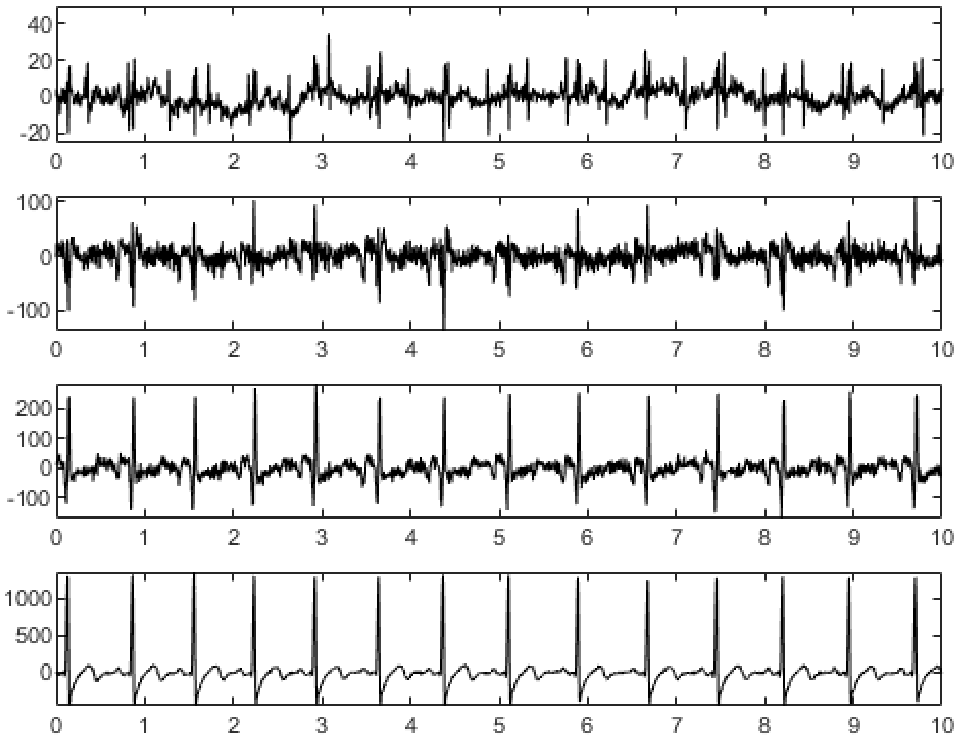

Figure 8: Separated sources using AMUSE algorithm after PCA using eigenvectors corresponding to the 4 largest eigenvalues.

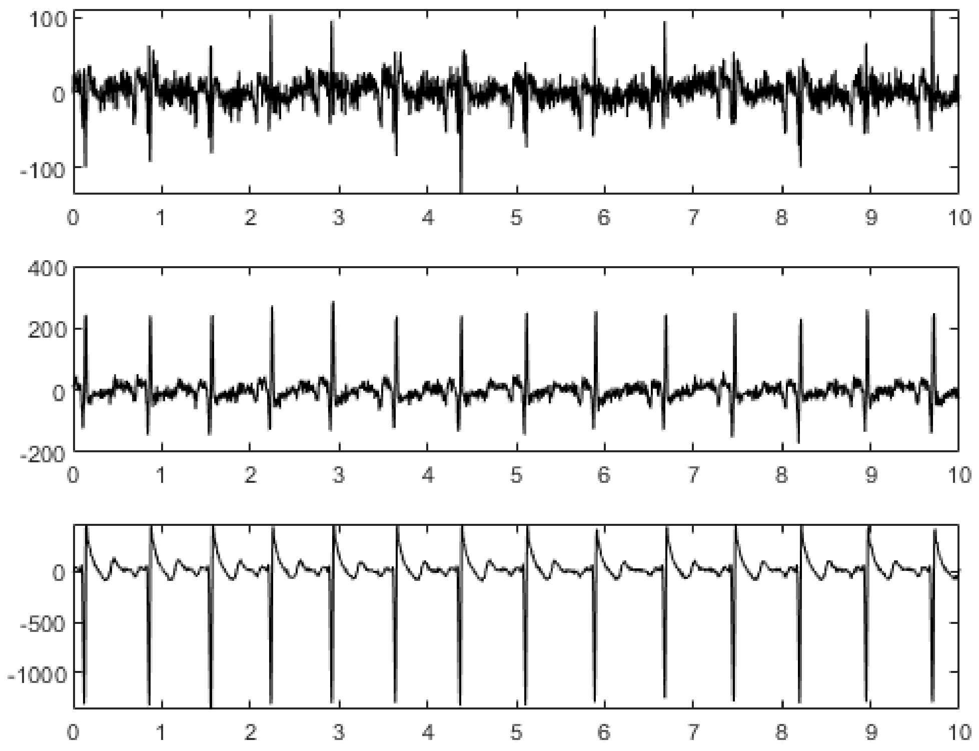

Figure 9: Separated sources using AMUSE algorithm after PCA using eigenvectors corresponding to the 3 largest eigenvalues.

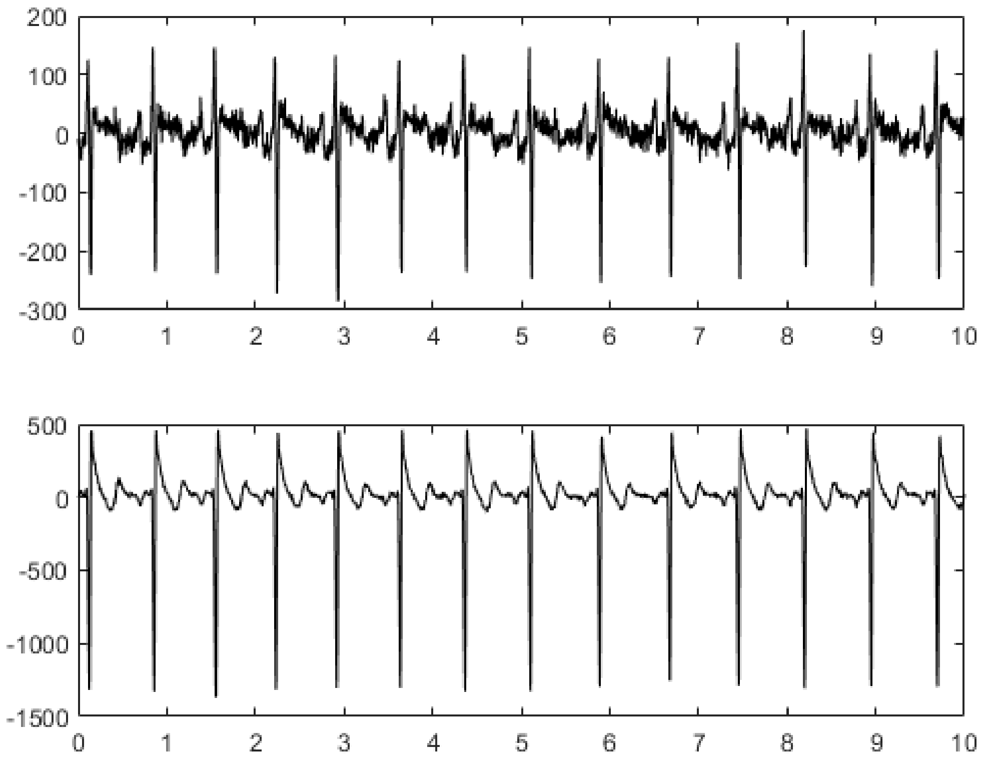

Figure 10: Separated sources using AMUSE algorithm after PCA using eigenvectors corresponding to the 2 largest eigenvalues.

3.4 Separation using FastICA

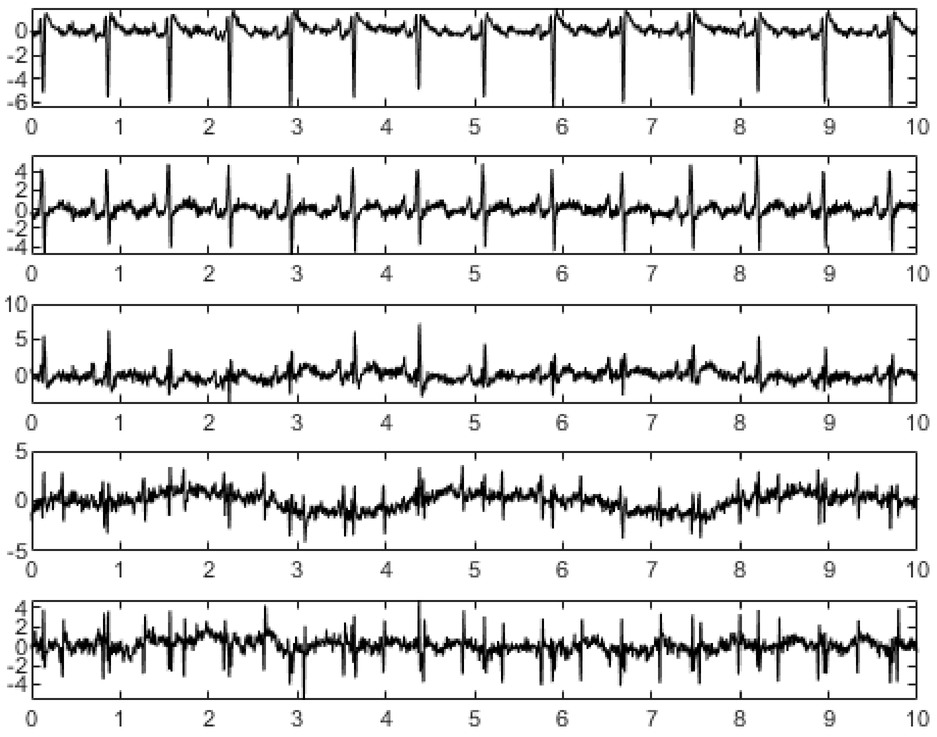

Figure 11: Separated sources using FastICA algorithm after PCA using eigenvectors corresponding to the 5 largest eigenvalues.

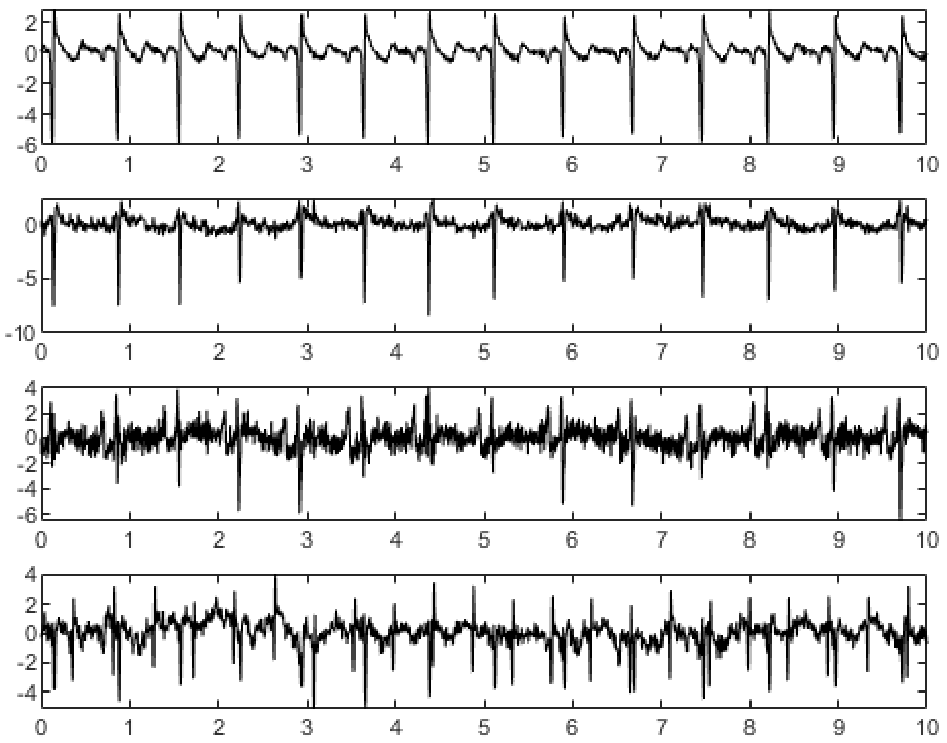

Figure 12: Separated sources using FastICA algorithm after PCA using eigenvectors corresponding to the 4 largest eigenvalues.

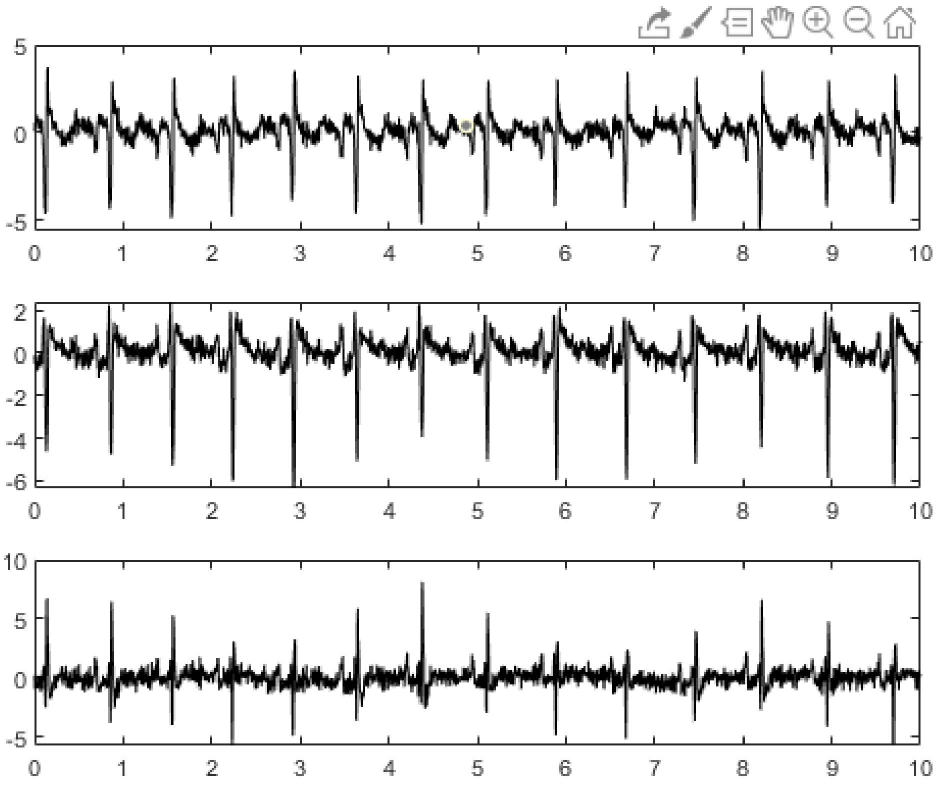

Figure 13: Separated sources using FastICA algorithm after PCA using eigenvectors corresponding to the 3 largest eigenvalues.

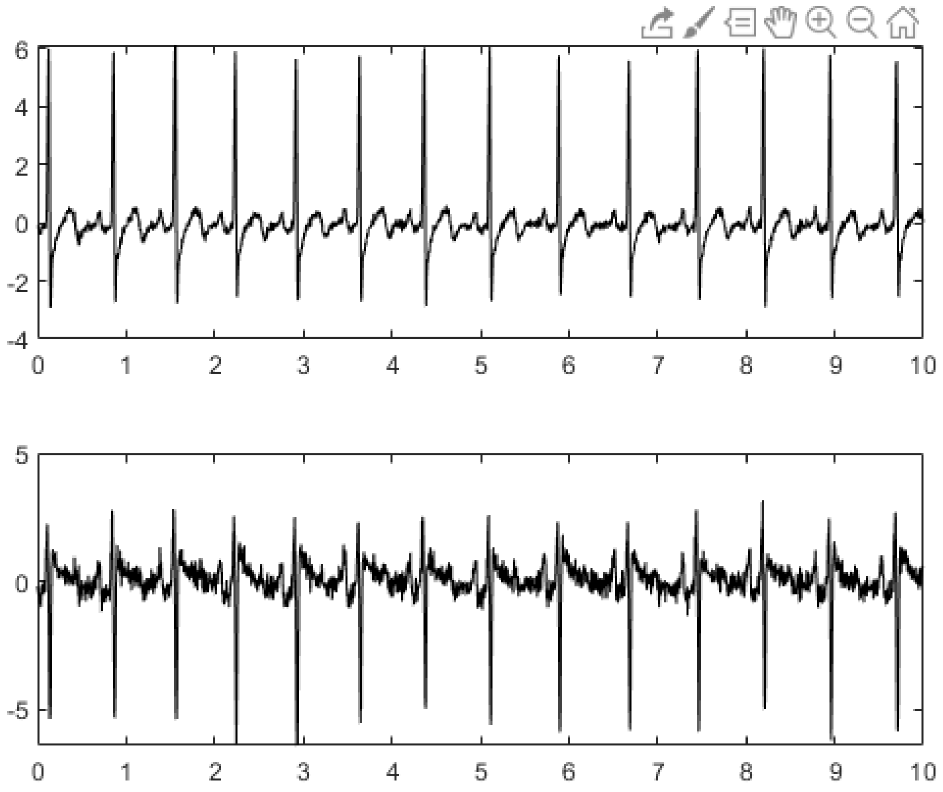

Figure 14: Separated sources using FastICA algorithm after PCA using eigenvectors corresponding to the 2 largest eigenvalues.

4 Discussion

Any good separation of the mother's and the baby's ECG was not possible. The mother's ECG, presumably with lower frequency and larger amplitude, was dominant and easy to obtain. The baby's ECG, however, was not.

The best results was obtained when using AMUSE preceded with PCA using 5 eigenvectors, see figure 7 index 1 and 5. These are also extracted and presented below.

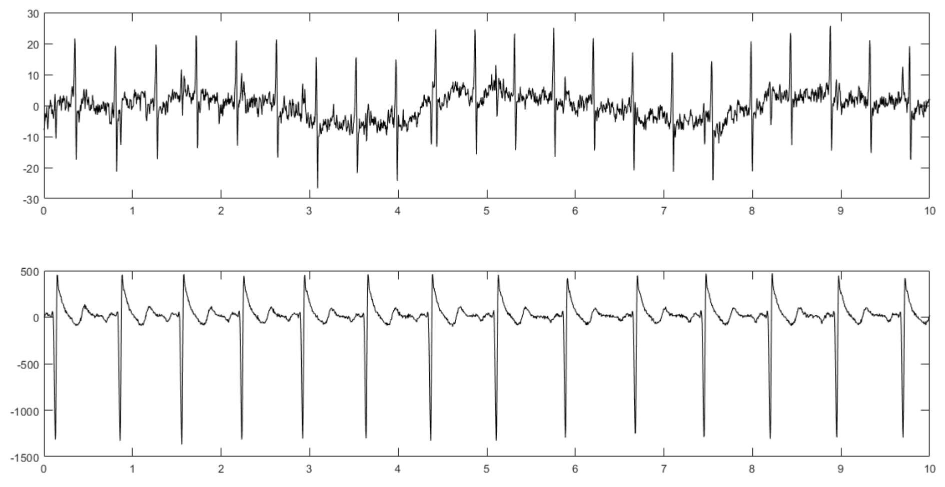

Figure 15: Index 1 and 5 extracted from results using AMUSE preceded with PCA using 5 eigenvectors.

It is unclear why FastICA could not separate the ECG sources as good as AMUSE. One reason could be that in this case time structure contains a lot of information which FastICA does not use.

The low frequency in the baby's ECG could the mother's or the baby's breathing that interfere with the measurements.

5 Conclusion

AMUSE seems to be better in the context of ECG and the number of eigenvectors used in the PCA pre-processing step seems to be crucial. Too few principal components will result in poor separation.

6 Appendix

6.1 FastICA

Implementation of FastICA using the deflationary approach.

```
function [Y, W] = fastICA(X)

[dimW, M] = size(X);
```

```
% Centering data.
meanX = mean(X.')';
X = X - meanX;

%% 2) Whitening of data.
covX = cov(X.');
[E, D] = eig(covX);

% Whitening matrix.
V = D^(-1/2) * E.';

% Whitened data Z.
Z = V*X;

W = eye(dimW);

% Defines convergence.
epsilon = 1E-10;

for p = 1:dimW
    wp = W(:,p);

    while 1==1
        wold = wp;

        a1 = 1;
        temp = tanh(a1 * Z.' * wp);
        wp = (Z * temp - a1 * sum(1 - temp.^2) * wp) ./M;

        % Orthogonalize (using deflation).
        for j = 1:(p-1)
            wj = W(:,j);
            wp = wp - (wp.'*wj)*wj; %./ (wj.'*wj) ;
        end

        % Normalize.
        wp = wp ./ vecnorm(wp);

        W(:,p) = wp;
        %% 8) Check convergence for wp.
        if (vecnorm(wp - wold) < epsilon | vecnorm(wp + wold) < epsilon)
            break; % Break while-loop.
        end
    end % End of while-loop.

end % End of for-loop (p).
```

```
% Obtain sources from Z
Y = W*Z;
end
```