1st Project in

DIGITAL SIGNAL PROCESSING

by Dimitrios Yfantidis (3938)

Department of Informatics,

Faculty of Sciences,

Aristotle University of Thessaloniki

April 2024

Task 1

Difficult: Using MATLAB's `signalAnalyzer()` tool, several extra viewpoints of the EEG's frequency domain can be extracted. The original signal is listed below:

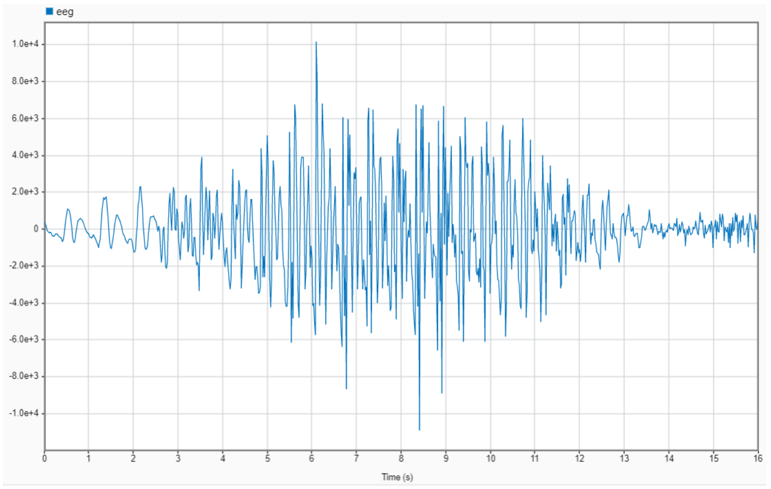


Figure 1: The EEG signal in time domain

By accessing the `signalAnalyzer` GUI, certain representations of the frequency spectrum can be derived such as the following graph and heatmap:

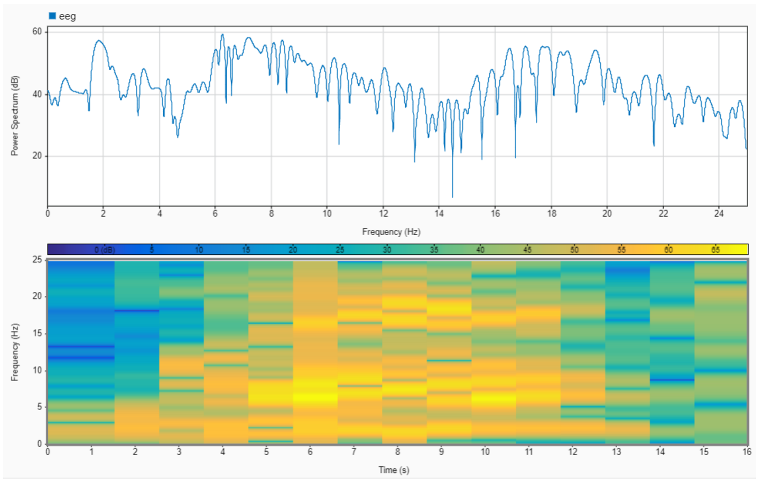


Figure 2: The EEG's power/frequency graph (top) & frequency/time heatmap (bottom)

Task 4

Difficult: Within script[5] an ECG signal is analyzed. After executing it, three 2-dimensional plots of the signal in time domain are displayed. The first one (top) is the original ECG signal, while the second (middle) and third (bottom) correspond to the processed version of the signal using an FIR and IIR filter respectively. The processing aims to remove the frequencies near 60Hz from the ECG signal.

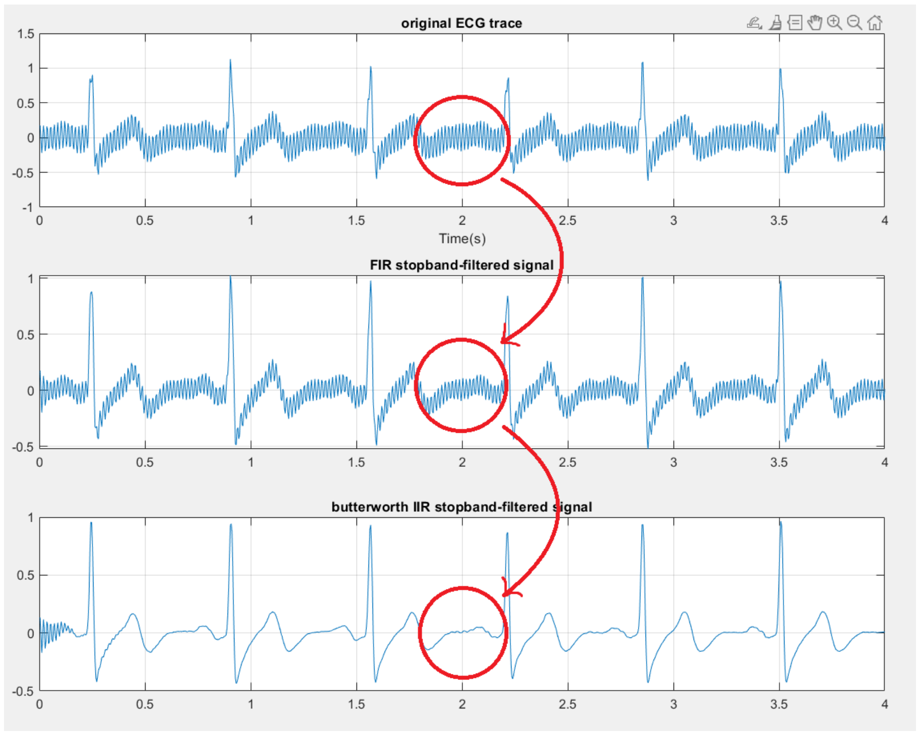


Figure 3: The original ECG signal followed by the filtered ECG signal using an FIR of order 36 and an IIR of order 3.

Although the frequency spectrum is not directly visible, it is clear that, given a time frame $[t1 + kT,\ t2 + kT],\ k \in \mathbb{Z}$, the IIR-processed signal is much smoother relative to the FIR-processed signal which exhibits greater similarity (visually speaking) to the original one.

This showcases the IIR-filter's effectiveness over the FIR one's, given that the FIR is of order 36 while the IIR is only of order 3; $\times 12$ lower.

By changing the FIR filter's order from 36 to 250 we can achieve better results in roughly the same time for the given signal.

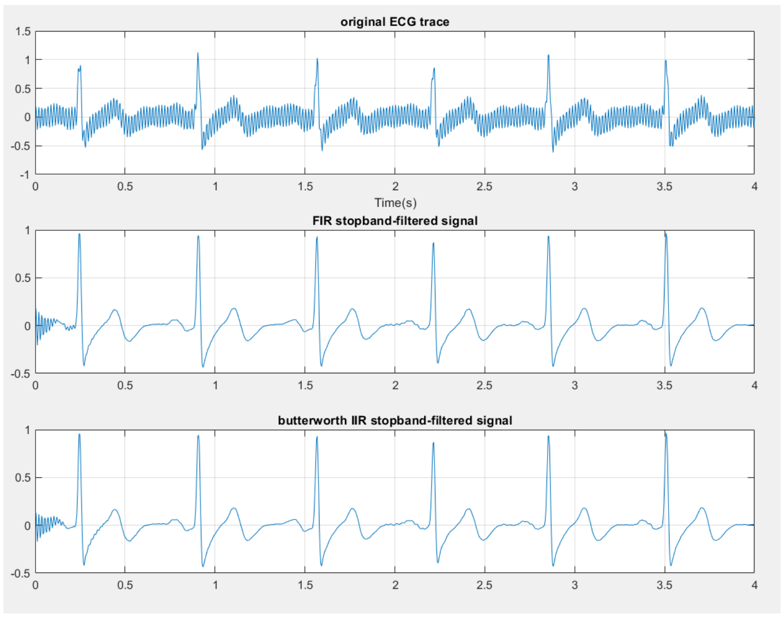


Figure 4: The original ECG signal followed by the filtered ECG signal using an FIR of order 250 and an IIR of order 3.

Now the FIR filtered signal and the IIR signal look identical. This can be validated with `signalAnalyzer()`, where we can observe that the FIR and IIR filtered signals' frequency spectrums are approximately equal, illustrating a similar cutoff around 60Hz.

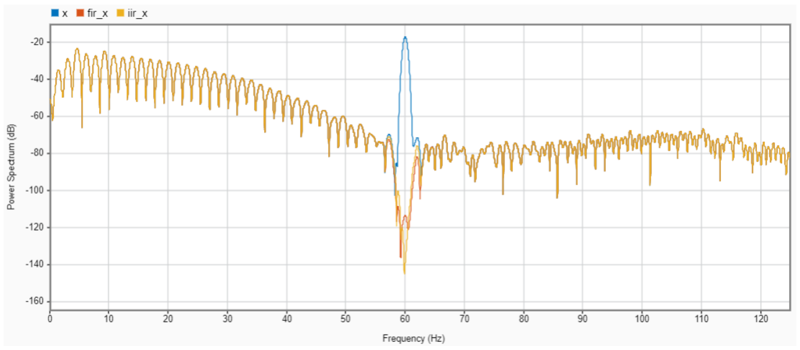
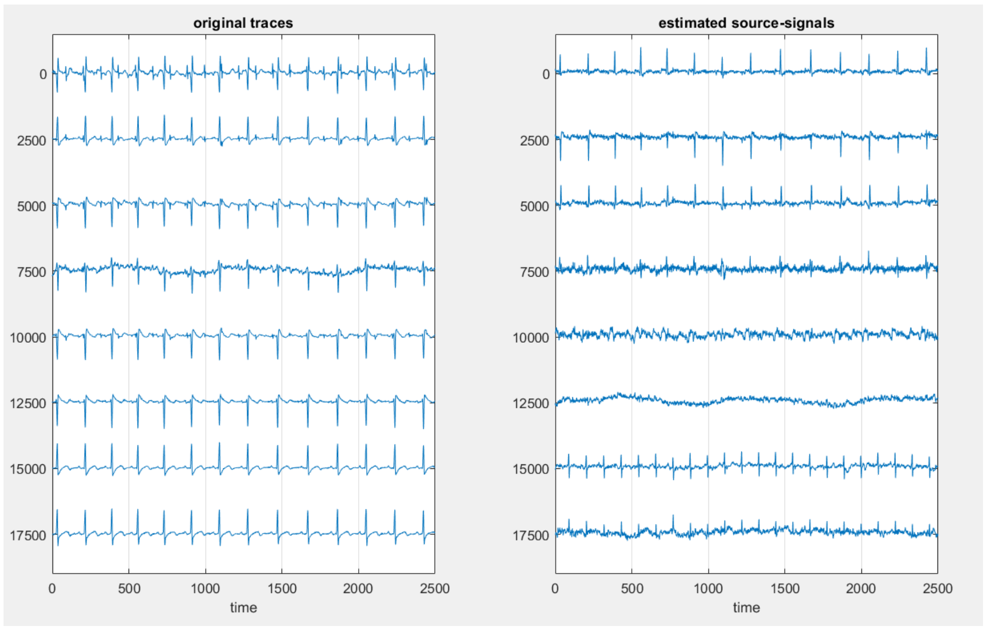


Figure 5: The frequency spectrum of each signal.

Task 6

Difficult: Given the fact that the JADE algorithm is deterministic then the resulting unmixing matrix will be always the same for a given input matrix, `ECGdata` in our case. The unmixing matrix is a square matrix with dimension equal to the number of recorded signals, i.e. for n recorded signals $\mathbf{x_i}(t)$ the algorithm will produce an $n \times n$ matrix, W, that will be used to determine the source signals $\mathbf{s_i}(t)$ using the equation: $\mathbf{S} = \mathbf{WX}$



On the other hand, the FastICA algorithm is not entirely deterministic as it involves **weight initialization** in a randomized way, which is common in statistical learning. As a result, running the FastICA algorithm may yield approximately equal but not equal results.

These claims can be examined with the following MATLAB code segments:

FastICA Test:

```
[sources1] = fastica (ECGdata,'numOfIC', 3);
[sources2] = fastica (ECGdata,'numOfIC', 3);
[sources3] = fastica (ECGdata,'numOfIC', 3);
disp(isequal(sources1, sources2))
disp(isequal(sources2, sources3))
```

The code segment above prints “0” twice in the terminal, proving that different runs of the FastICA algorithm do not yield the same results. Now, let’s test the JADE algorithm.

JADE Test:

```
B1=jadeR(ECGdata); % deriving the unmixing-matrix
Sources1=B*ECGdata; % estimating the ICs (source-signal)

% Repeat 3 times
B2=jadeR(ECGdata);
Sources2=B2*ECGdata;
B3=jadeR(ECGdata);
Sources3=B3*ECGdata;
B4=jadeR(ECGdata);
Sources4=B4*ECGdata;

disp(isequal(Sources1, Sources2));
disp(isequal(Sources2, Sources3));
disp(isequal(Sources3, Sources4));
```

The code segment above prints “1” three times, reinforcing that “Joint Approximation Diagonalization of Eigen-matrices” is a deterministic procedure.

Reason for using ICA:

The reason for performing ICA is to derive accurate electrocardiographic data from the foetus. This is a complex procedure as the ECG will record other data along with the foetus’ heart rate such as the mother’s heart rate and even other noise signals. Thus using ICA is necessary in such problems as multiple source signals, $\mathbf{s_{foetus}}(t)$ and $\mathbf{s_{mother}}(t)$, can only be determined by multiple sensor readings $\mathbf{x_1}(t)$, $\mathbf{x_2}(t)$, ..., $\mathbf{x_n}(t)$ using this technique.

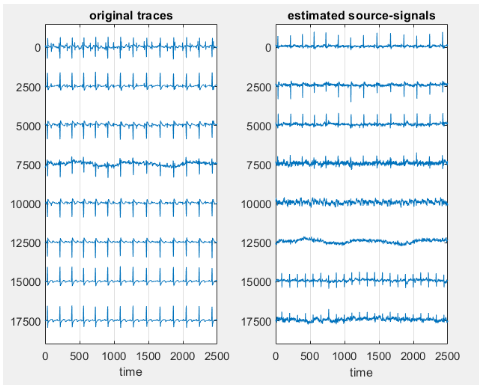
Using less sensors:

We can assign the $\mathbf{X}$ matrix to only its first four rows using the MATLAB command

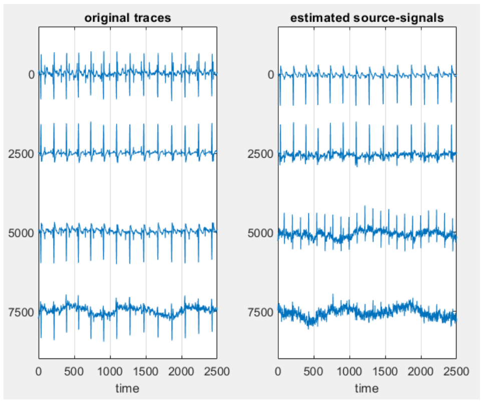
```
ECGdata = ECGdata(1:4, :);
```

This way, instead of 8 sensor readings, $\mathbf{x_1}(t)$, $\mathbf{x_2}(t)$, ..., $\mathbf{x_8}(t)$ we only have 4 sensor readings: $\mathbf{x_i}(t),\ i = 1..4$. The independent components will still be evaluated but this time with less input data. Thus an output of lower resolution (lower accuracy) will be derived.

Obviously, the JADE algorithm will return a 4×4 unmixing matrix instead of 8×8 because, as was previously mentioned, the output is a square matrix with dimension equal to the number of recorded signals. Thus, 4 source signals will be determined using $\mathbf{S} = \mathbf{WX}$.

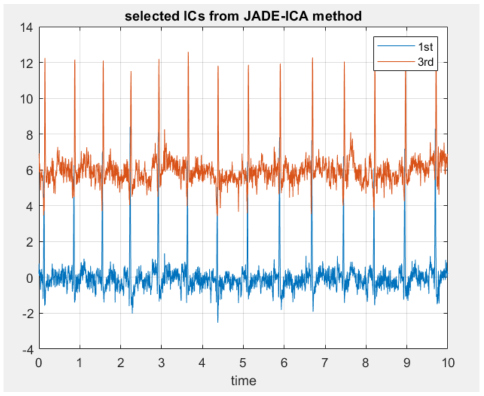


Before

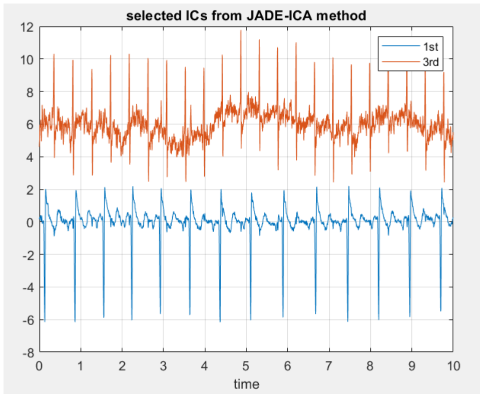


After

When selecting the 1st and 3rd estimated source signals, i.e. $\mathbf{s_1}(t)$ and $\mathbf{s_3}(t)$, several differences can be observed. The signal $\mathbf{s_1}(t)$ seems smoother at the time intervals between spikes. This could be attributed to the fact that less recorded signals offer less information and thus the estimated signals are not as rich in information as they were initially. Also, the mean value of the second signal, $\mathbf{s_3}(t)$, varies greater between the spikes in relation to its initial estimation.

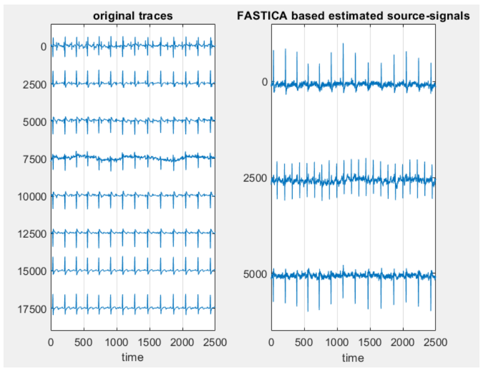


Before

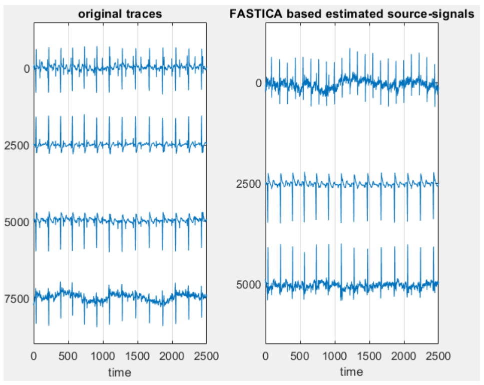


After

Lastly, the previous observation can be applied to the final graph. The FastICA algorithm is producing 3 estimated source signals, as it was tasked to do. Though, this time it has less information (recorded signals) to work with and the resulting estimations may not be as representative of the source signals as before.



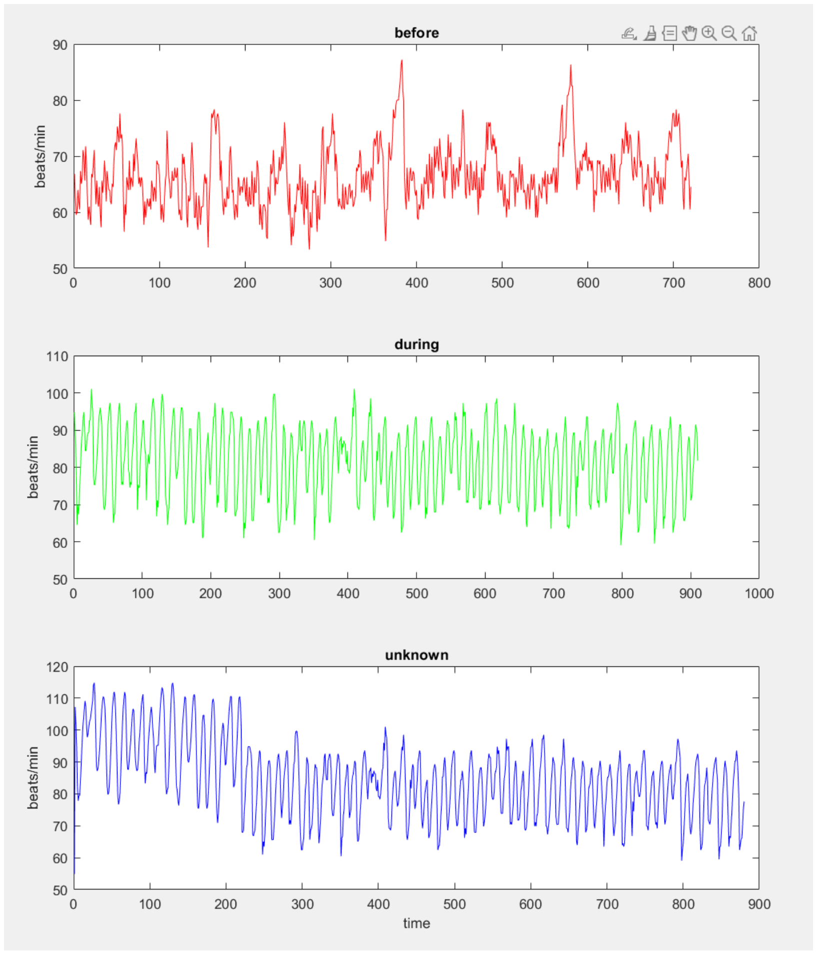
Before



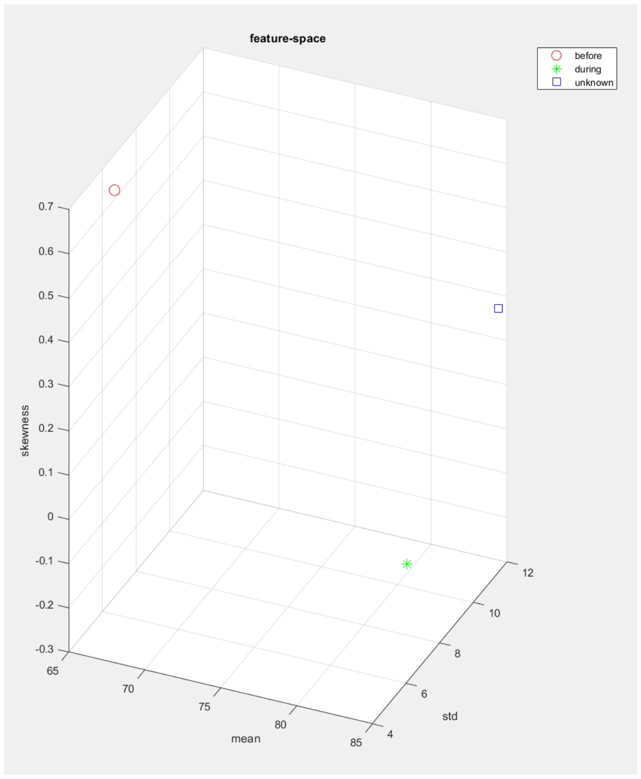
After

Task 7

Difficult: After running script[8], three 2-dimensional plots and one 3-dimensional plot are generated. The 2D plots show two different signals of a persons heart rate before and during meditation, as well as a third signal showing the person's heart rate during an unknown state.



Certain statistical features are extracted from the three signals, namely mean bpm, standard deviation and skewness. These metrics are combined to represent each one of the three signals as a 3D point, as illustrated in the following plot:



It is evident that, while the 3 signals are evenly distributed along the "skewness" axis, the unknown signal (blue) is closer to the signal of the heart's rate during meditation (green) than the one indicating before meditation (red), in terms of standard deviation and mean value. This can be verified by appending and executing this code segment at the end of the script:

```
A = [Features_unknown(:,1:3)', Features_pre(:,1:3)', Features_med(:,1:3)'];
A(:, 2) = abs(A(:, 1) - A(:,2));
A(:, 3) = abs(A(:, 1) - A(:,3));
```

Essentially, in the beginning:

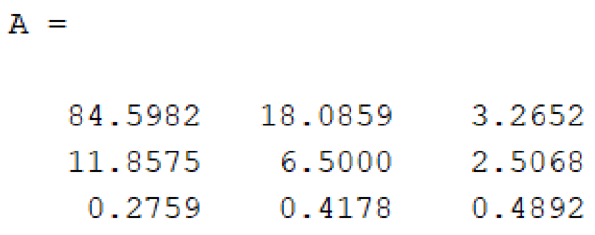
$$A = \begin{pmatrix} mean(\mathbf{x}(t)) & mean(\mathbf{a}(t)) & mean(\mathbf{b}(t)) \\ std(\mathbf{x}(t)) & std(\mathbf{a}(t)) & std(\mathbf{b}(t)) \\ skew(\mathbf{x}(t)) & skew(\mathbf{a}(t)) & skew(\mathbf{b}(t)) \end{pmatrix}$$

where $\mathbf{x}(t)$ is the unknown signal, $\mathbf{a}(t)$ is the "before" signal and $\mathbf{b}(t)$ is the "during" signal. Later, the second column is replaced with the **distance** between signals $\mathbf{x}(t)$ and $\mathbf{a}(t)$ along each axis. Likewise, the third column is replaced with the distances between $\mathbf{x}(t)$ and $\mathbf{b}(t)$.

Thus:

$$A = \begin{pmatrix} mean(\mathbf{x}(t)) & d_{mean}(\mathbf{x}(t), \mathbf{a}(t)) & d_{mean}(\mathbf{x}(t), \mathbf{b}(t)) \\ std(\mathbf{x}(t)) & d_{std}(\mathbf{x}(t), \mathbf{a}(t)) & d_{std}(\mathbf{x}(t), \mathbf{b}(t)) \\ skew(\mathbf{x}(t)) & d_{skew}(\mathbf{x}(t), \mathbf{a}(t)) & d_{skew}(\mathbf{x}(t), \mathbf{b}(t)) \end{pmatrix}$$

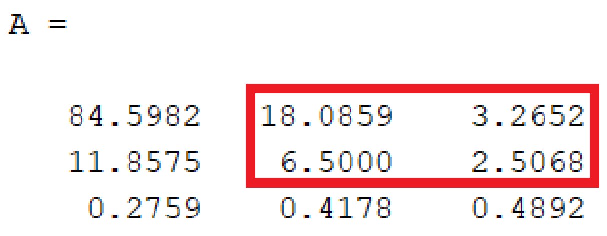
By displaying the matrix, the following result is printed:



```
A =


   84.5982   18.0859    3.2652
   11.8575    6.5000    2.5068
    0.2759    0.4178    0.4892
```

From this, we can make certain useful observations:



```
A =


   84.5982   18.0859    3.2652
   11.8575    6.5000    2.5068
    0.2759    0.4178    0.4892
```

It is evident that using 2 out of 3 characteristics, namely, **mean value** and **standard deviation**, we can confidently classify it as closer to the green signal than the red signal since

$$d_{mean}(\mathbf{x}(t), \mathbf{a}(t)) \gg d_{mean}(\mathbf{x}(t), \mathbf{b}(t))$$

and

$$d_{std}(\mathbf{x}(t), \mathbf{a}(t)) > d_{std}(\mathbf{x}(t), \mathbf{b}(t))$$