Music and Co.
COM-500 Statistical signal and data processing through applications : Mini-project report

Samuel Dubuis
IC - EPFL
s.dubuis@epfl.ch

Benoit Knuchel
IC - EPFL
benoit.knuchel@epfl.ch

Abstract

In this mini-project are discussed parametric line spectra estimation methods which takes noise into account. The first method that we will analyze is the MUSIC algorithm. While simple and elegant, it is quite limited. Thus we will discover other methods such as ESPRIT and SAMV, which we will implement and compare.

I. Introduction

In statistical signal processing, the goal of frequency estimation is the process of estimating the complex frequency components of a signal in the presence of noise given assumptions about the number of components [2]. Depending on prior assumptions about the components, we can use different algorithms that are more or less powerful and efficient.

The first that will be discussed in this report is the MUSIC algorithm. One of the first algorithm implemented that would take into account knowledge of components of the signal. Based on an eigenspace method, MUSIC is very effective in the presence of noise but still requires few samples to provide satisfactory results.

However, MUSIC has a few disadvantages that we will recreate and explain. Thus we will also experiment on the ESPRIT algorithm, which was later created and uses the least squares method with the Moore-Penrose inverse.

Finally, we will introduce and explain one of the latest discovered algorithm for frequency estimation, the SAMV algorithm. It is a parameter-free resolution of the linear inverse problem in spectral estimation. We will experiment with it and we should see that it is the most powerful and efficient algorithm of the three to estimate frequencies of a noisy signal.

II. Procedure

To experiment with each algorithm we created some fake data and we were also given some real data. All computations in this report are made in the software MATLAB. The fake data can be mathematically recreated as such

$$X_1[n] = \sin(2\pi f_1 \frac{n}{f_s}) + \sin(2\pi f_2 \frac{n}{f_s}) + \sin(2\pi f_3 \frac{n}{f_s}), \text{ with } \begin{matrix} f_1 = 900 \\ f_2 = 500 \\ f_3 = 200 \\ f_s = 2000 \end{matrix} \text{ and } n \in N = (f_s * 2) \quad (1)$$

$$X_2[n] = \sin(2\pi f_1 \frac{n}{f_s}) + \sin(2\pi f_2 \frac{n}{f_s}) + \sin(2\pi f_3 \frac{n}{f_s}), \text{ with } \begin{matrix} f_1 = 900 \\ f_2 = 500 \\ f_3 = 450 \\ f_s = 2000 \end{matrix} \text{ and } n \in N = (f_s * 2) \quad (2)$$

$$Y_{1,2}[n] = X_{1,2}[n] + W[n] \tag{3}$$

with $W[n]$ defined as an Additive White Gaussian Noise (AWGN) of $SNR = 0.1$. At higher SNR, the algorithms make only few to no mistakes. We use the second signal $X_2[n]$ to have two close frequencies, $f2 = 500hz$ and $f3 = 450hz$.

Fig. 1 and Fig. 3 show the signals $X_1[n]$ and $X_2[n]$ over the first 100 samples, while Fig. 2 and Fig. 4 represent the noisy signals $Y_1[n]$ and $Y_2[n]$. We can already see that the noise has a considerable impact on the signal.

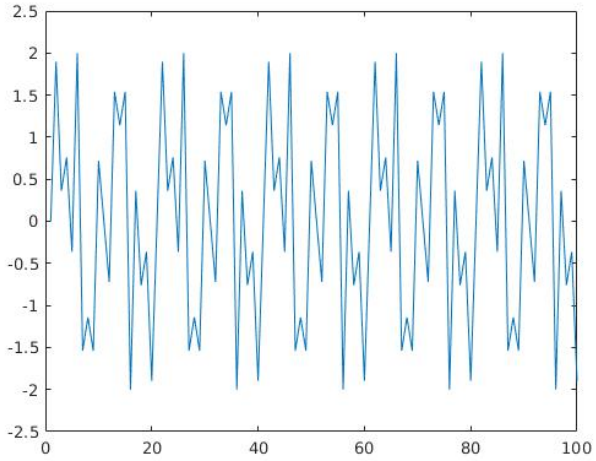

Fig. 1: Plot of $X_1[n]$ for 100 samples

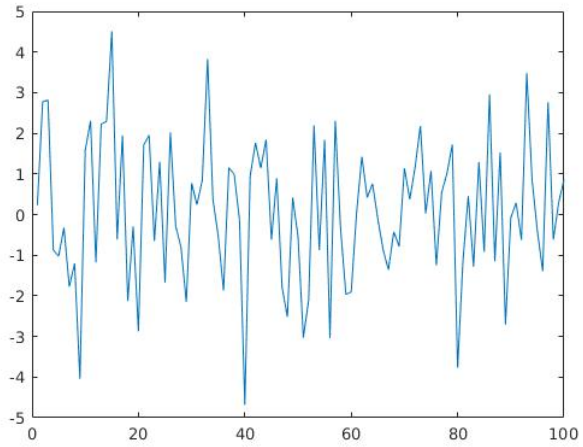

Fig. 2: Plot of $Y_1[n]$ for 100 samples

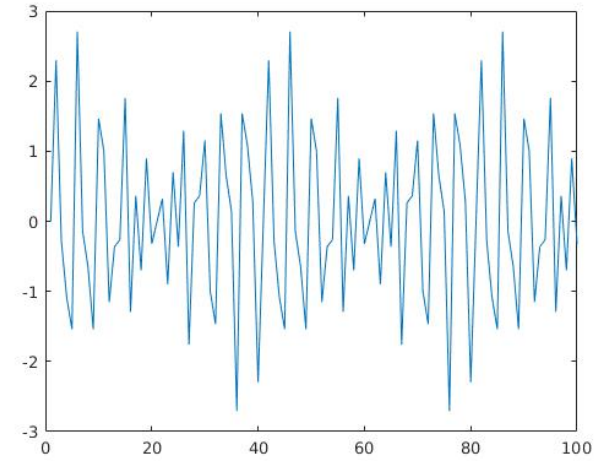

Fig. 3: Plot of $X_2[n]$ for 100 samples

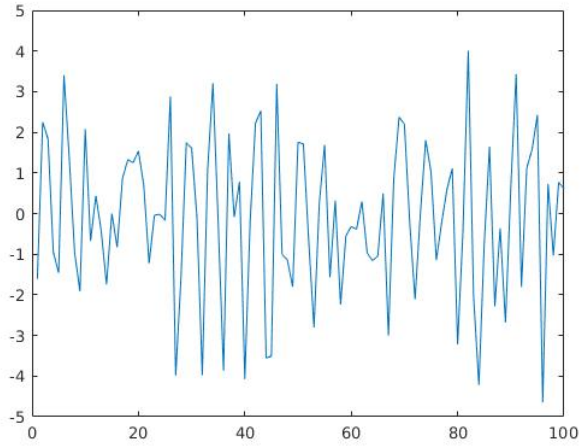

Fig. 4: Plot of $Y_2[n]$ for 100 samples

We also test the selected algorithms on real data that we were given : a bass line that is either clean or noisy with four different notes. We present those results at the end of the 'Results' section.

A. MUSIC

The MUSIC algorithm, short for MUltiple SIgnal Classification formulates the estimation problem using the correlation matrix of a sequence of observations of the noisy process $Y[n]$. Following [3] and [5],

we can explain the MUSIC algorithm more precisely. First it exploits the independence of the harmonic signal $X[n]$ and the noise $W[n]$ to express the correlation matrix in a particular form. Then it uses the properties of the eigendecomposition of the correlation matrix to estimate the amplitudes and the positions of the spectral lines.

Now we will describe mathematically how it is done.
We know that our harmonic signal, corrupted by noise is defined as such

$$Y[n] = X[n] + W[n] = \sum_{k=1}^{K} \alpha_k e^{j(\omega_k n + \Theta_k)} + W[n], n \in \mathbb{N} \tag{4}$$

where $W[n]$ is the white noise independent of $X[n]$, centered and with variance σ_W^2.

We can first define two matrices

$$\mathbf{E}^{NK} = [\mathbf{e}^{N1}(\omega_1) \dots \mathbf{e}^{N1}(\omega_K)] = \begin{bmatrix} 1 & 1 & \dots & 1 \\ e^{-j\omega_1} & e^{-j\omega_2} & \dots & e^{-j\omega_K} \\ \vdots & \vdots & & \vdots \\ e^{-j(N-1)\omega_1} & e^{-j(N-1)\omega_2} & \dots & e^{-j(N-1)\omega_K} \end{bmatrix} \tag{5}$$

$$\mathbf{A}^{KK} = \begin{bmatrix} \alpha_1^2 & 0 & \dots & 0 \\ 0 & \alpha_2^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \alpha_K^2 \end{bmatrix} \tag{6}$$

and because we are given a noisy signal $Y[n]$ observed at N time instants $Y[1], \dots, Y[N]$, we can rewrite the noisy harmonic signal as

$$\mathbf{Y}^{N1}[n] = \mathbf{E}^{NK}\mathbf{X}^{K1}[n] + \mathbf{W}^{N1}[n] \tag{7}$$

with

$$\mathbf{Y}^{N1}[n] = \begin{bmatrix} Y[n] \\ \vdots \\ Y[n-(N-1)] \end{bmatrix}, \mathbf{X}^{K1}[n] = \begin{bmatrix} \alpha_1 e^{j(\omega_1 n + \theta_1)} \\ \vdots \\ \alpha_K e^{j(\omega_K n + \theta_K)} \end{bmatrix}, \mathbf{W}^{N1}[n] = \begin{bmatrix} W[n] \\ \vdots \\ W[n-(N-1)] \end{bmatrix}$$

The correlation matrix $R_{\mathbf{Y}}^{NN}$ of $\mathbf{Y}^{N1}[n]$ is

$$R_{\mathbf{Y}}^{NN} = E[\mathbf{Y}^{N1}[n]\mathbf{Y}^{N1}[n]^H] \tag{8}$$

and thus

$$\mathbf{R}_{\mathbf{Y}}^{NN} = \mathbf{E}^{NK}\mathbf{A}^{KK}\mathbf{E}^{NK^H} + \sigma_W^2\mathbf{I}^{NN} \tag{9}$$

Then, we do the eigendecomposition of $\mathbf{R}_{\mathbf{Y}}^{NN}$, getting the eigenvalues $\lambda_n^R = \lambda_n^{EAE} + \sigma_W^2$ and their corresponding eigenvectors $\mathbf{G}^{NN}$. Note that $\lambda_n^{EAE} = 0$ for $n = K+1, ..., N$, so by choosing $\mathbf{G}^{N(N-k)}$ as the eigenvectors of λ_{K+1}^R to λ_N^R we get the following equality :

$$\mathbf{e}^{N1}(\omega)^H\mathbf{G}^{N(N-k)}\mathbf{G}^{N(N-k)^H}\mathbf{e}^{N1}(\omega) = 0 \tag{10}$$

The algorithm to extract the frequencies goes as follows :

First compute the empirical covariance matrix:

$$\hat{\mathbf{R}}_{\mathbf{Y}}^{NN} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{Y}^{N1}[n]\mathbf{Y}^{N1H}[n] \tag{11}$$

Then do the eigendecomposition of the empirical covariance $\hat{\mathbf{R}}_{\mathbf{Y}}^{NN}$ and end up with $\hat{\mathbf{G}}^{N(N-k)}$, the eigenvectors corresponding to the noise.

Finally, we can extract the frequencies by finding the peaks of :

$$\frac{1}{\mathbf{a}(\omega)^H \hat{\mathbf{G}}^{N(N-k)} \hat{\mathbf{G}}^{N(N-k)^H} \mathbf{a}(\omega)} \tag{12}$$

B. ESPRIT

The second algorithm we implemented is ESPRIT, short for Estimation of Signal Parameters via Rotational Invariant Techniques, based on [6]. We followed [7] to implement it. We focus on the case of multiple frequencies as it is the one that interests us in our mini-project.

We define two vectors :

$$A_1 = \begin{bmatrix} I_{m-1} & 0 \end{bmatrix} A, \tag{13}$$

and

$$A_2 = \begin{bmatrix} 0 & I_{m-1} \end{bmatrix} A \tag{14}$$

both of size $(m-1) \times n$, where I_{m-1} is the identity matrix.

We define now the rotation matrix $\mathbf{D}$

$$\mathbf{D} = \begin{bmatrix} e^{-i\omega_1} & & 0 \\ & \ddots & \\ 0 & & e^{-i\omega_n} \end{bmatrix} \tag{15}$$

and we can observe that $A_2 = A_1\mathbf{D}$

Now, in reality, we define again two vectors S_1 and S_2 :

$$S_1 = \begin{bmatrix} I_{n-1} & 0 \end{bmatrix} S, \tag{16}$$

and

$$S_2 = \begin{bmatrix} 0 & I_{n-1} \end{bmatrix} S \tag{17}$$

The goal now is to find the matrix ϕ such that $S_2 = S_1\phi$.

Using the Moore-Penrose inverse [4] for ϕ :

$$\phi = (S_1^* S_1)^{-1} S_1^* S_2 \tag{18}$$

And finally the angular frequencies ω_n are then given by :

$$\omega_n = -\arg(\lambda_n) \tag{19}$$

with λ being the eigenvalues of ϕ.

C. SAMV algorithm

The final algorithm is SAMV, meaning iterative Sparse Asymptotic Minimum Variance. Following [1] (we implemented the SAMV-1 version), we can define this iterative algorithm.

The idea is to generate a matrix $\mathbf{A}$ for the range of potential frequencies $[\omega_0, \dots \omega_K]$ for a very large K, usually unknown, so that the possibilities are many, and to update the power vectors $\hat{p}_k$ for each of the potential frequency ω_k, based on the formula we will describe below. When the algorithm converges, the matrix $\mathbf{P}$ will become sparse (with very few non-zero elements) and only the relevant powers $\hat{p}_k$ will be non-zero.

Thus we define the algorithm as such : The matrix $\mathbf{A}$ is given by $\mathbf{A} = [\mathbf{a}(\omega_1) \dots \mathbf{a}(\omega_K)]$. Then we initialize the diagonals value of $\mathbf{P}$

$$\text{diag}(\mathbf{P}^{(0)}) = [\hat{p}_1^{(0)} \dots \hat{p}_k^{(0)}]^T = \frac{1}{N}\|\mathbf{A}^H\mathbf{y}\| \tag{20}$$

and

$$\hat{\sigma}^{(0)} = \frac{1}{N}\sum_{n=1}^{N}\|Y[n]\|^2 \tag{21}$$

The covariance matrix of $Y[n]$ that conveys information about $\hat{p}$ is :

$$\mathbf{R} = \mathbf{A}\mathbf{P}\mathbf{A}^H + \sigma\mathbf{I} \tag{22}$$

Then at each iteration i we update the values of the $\hat{p}_k$ vectors and of the $\hat{\sigma}$ with the help of a matrix $\mathbf{R}$ (to ease the reading, we define $\mathbf{a}(\omega_k) = a_k$) :

$$\mathbf{R}^{(i)} = \mathbf{A}\mathbf{P}^{(i)}\mathbf{A}^H + \sigma^{(i)}\mathbf{I}$$

$$\hat{p}_k^{(i+1)} = \frac{a_k^H\mathbf{R}^{-1(i)}\mathbf{R}_N\mathbf{R}^{-1(i)}a_k}{(a_k^H\mathbf{R}^{-1(i)}a_k)^2} \tag{23}$$

$$\hat{\sigma}(0) = \frac{\text{Tr}(\mathbf{R}^{-2(i)}\mathbf{R}_N)}{\text{Tr}(\mathbf{R}^{-2(i)})} \tag{24}$$

with $\text{Tr}(\cdot)$ being the trace function, i.e. the sum of the diagonal elements of the matrix.

III. Results

A. MUSIC

Since we implemented the MUSIC algorithm we first want to compare the results of our implementation against the MATLAB function pmusic. Note that pmusic returns an estimate of the spectrum and not directly the spectral lines. So, in order to find the frequencies, we have to find the peaks of the returned spectrum.

We ran a 100 times the same experiments and plotted the results for $Y_1[n]$ (respectively $Y_2[n]$) on Fig. 5 (resp. Fig. 7) for our implementation of MUSIC and on Fig. 6 (resp. Fig. 8) for the function pmusic.

For the signal $Y_1[n]$, pmusic gives an almost perfect estimation of the frequencies even though the SNR is equal to 0.1. Our implementation of MUSIC is doing well since in approximately 70% of the cases it gets the frequencies correctly and in the other ones he gets two out of three. Running multiple instances of our implementation and averaging the results would get us the same results as pmusic.

It gets more complicated when two frequencies are close to each other as can tell Fig. 7 and Fig. 8. The algorithm pmusic is definitely more stable : it gets $f_1 = 900hz$ every time correctly but is having some troubles defining precisely the two frequencies $f2 = 500hz$ and $f3 = 450hz$. It is still doing much better than our implementation that does not even let a hint about $f3$.

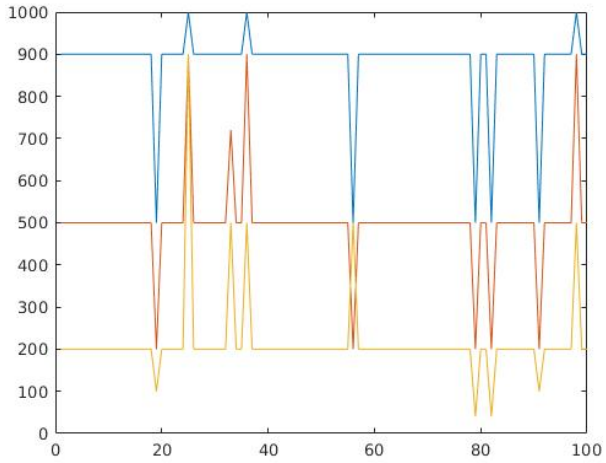

Fig. 5: Results for 100 experiments of $Y_1[n]$ using our implementation

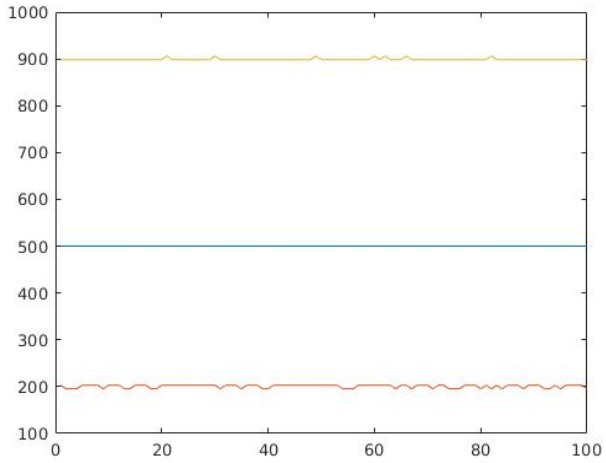

Fig. 6: Results for 100 experiments of $Y_1[n]$ using pmusic

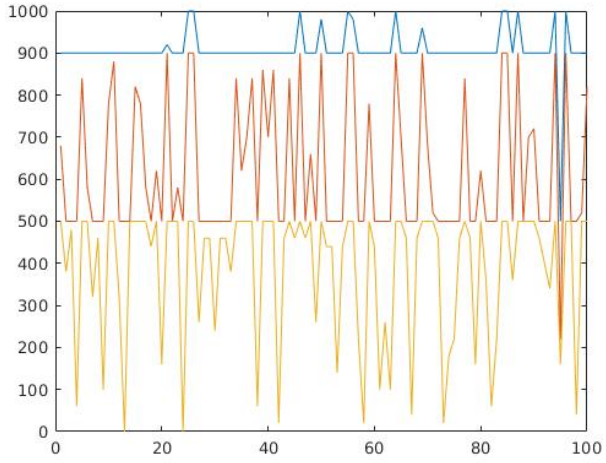

Fig. 7: Results for 100 experiments of $Y_2[n]$ using our implementation

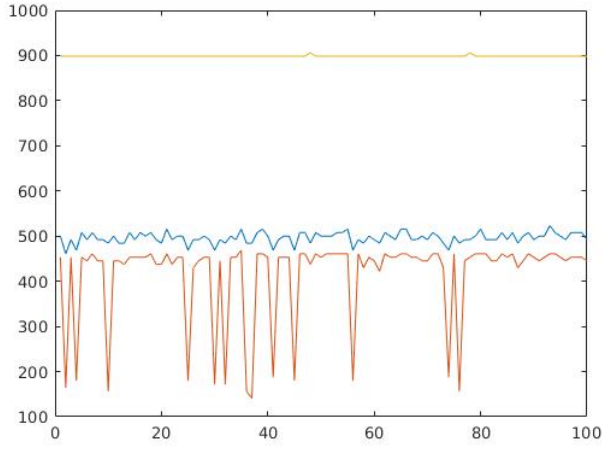

Fig. 8: Results for 100 experiments of $Y_2[n]$ using pmusic

B. ESPRIT

ESPRIT has been declared as providing slightly more accurate frequency estimates than MUSIC, and this at lower computational costs. Also, as seen in equation 18, ESPRIT has no problems separating "signal roots" from the "noisy roots". This is a specificity shared by MUSIC though, but ESPRIT seems to outperform MUSIC in terms of statistical accuracy.

As shown by Fig. 9 and Fig. 10 ESPRIT is doing much better than MUSIC : we clearly see the three frequencies even when two are close to each other. By testing a bit more with the closeness of frequencies

we can find that the limit for two of them to be recognisable by ESPRIT is approximately $10hz$ for a sampling frequency of $2000hz$.

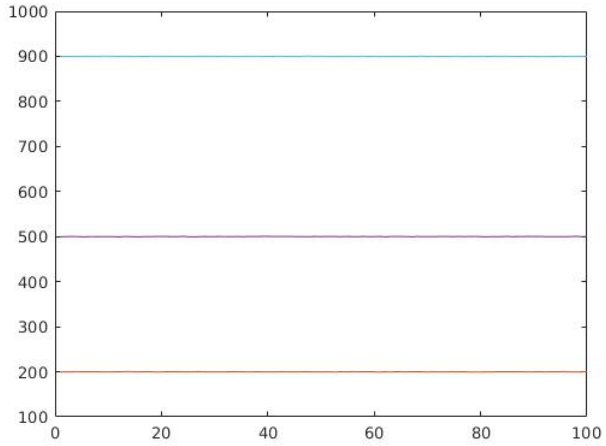

Fig. 9: Results for 100 experiments of $Y_1[n]$ using ESPRIT

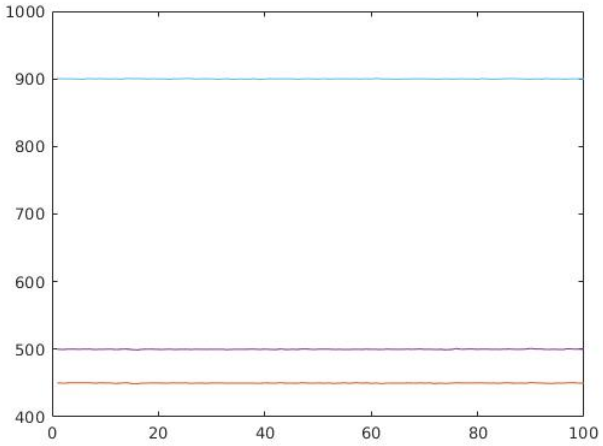

Fig. 10: Results for 100 experiments of $Y_2[n]$ using ESPRIT

C. SAMV

The SAMV algorithm is one of the most recent ones about spectra estimation. It has more applications than this one, in particular it is often used in Direction-Of-Arrival (DOA) estimation problem. Its implementation is complex and takes quite a lot of time to run, but the results are much better and the noise has a lower impact : the algorithm detects the same peaks whether the SNR is of 0.1 or 1. Since it takes more than three minutes to run we did not make a hundred iterations as testing. Instead we ran the algorithm a few times for each experiment..

As shown by Fig. 11 and Fig. 12 the estimated lines are clear except for $f3 = 900hz$. After testing a bit more for frequencies around $fs/2$, it can be seen that the algorithm has some troubles finding them and is wrong half the time about it. Another example is shown on Fig. 13 where the signal is composed of three frequencies with $f1 = 20hz$, $f2 = 500hz$ and $f3 = 980hz$. In this case SAMV missed $f3$, but MUSIC and ESPRIT would have find it.

Since ESPRIT could not recognise frequencies that are only $10hz$ apart, we wanted to test the same with SAMV. Fig. 14 shows the returned line estimation for a signal with $f1 = 497hz$, $f2 = 500hz$ and $f3 = 510hz$ and SNR at 0.1. It gets the three frequencies correctly with an error of $\pm 1hz$, which is where ESPRIT failed.

D. Real Data

We asked MUSIC and ESPRIT to find 24 spectral lines since there are 4 notes and one note can have multiple lines.

MUSIC returns the following frequencies $[hz]$ (their negative symmetric and the ones higher than $10khz$ are not shown) :

- clean : 0, 441, 882, 1323, 2205, 3528, 7938
- noisy : 0, 441, 882, 1323, 2646, 3087, 3528, 6174, 8820

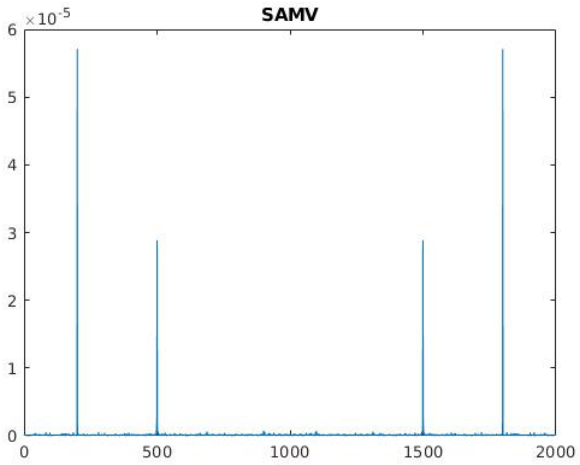

Fig. 11: Result for $Y_1[n]$ using SAMV

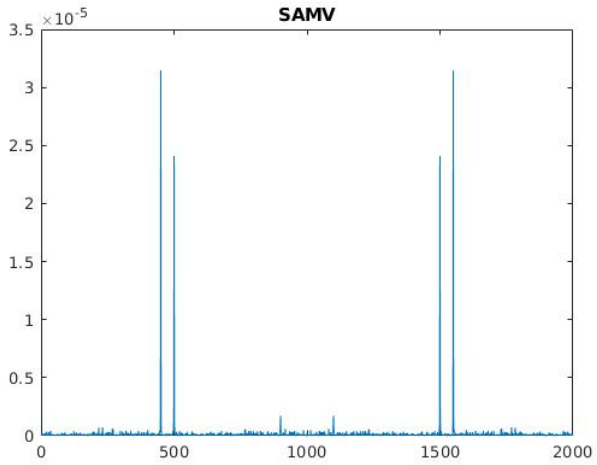

Fig. 12: Result for $Y_2[n]$ using SAMV

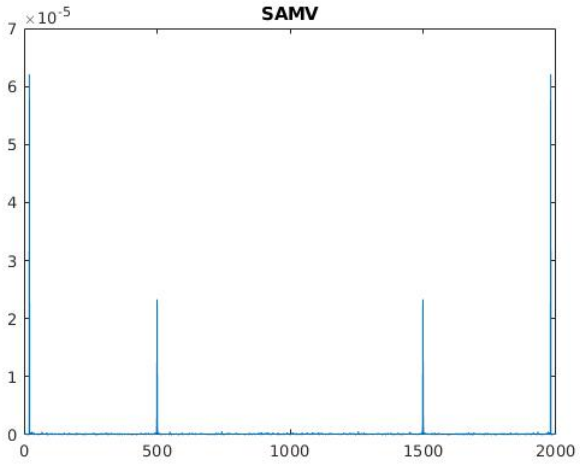

Fig. 13: Result for $f1 = 20hz$, $f2 = 500hz$ and $f3 = 980hz$ using SAMV

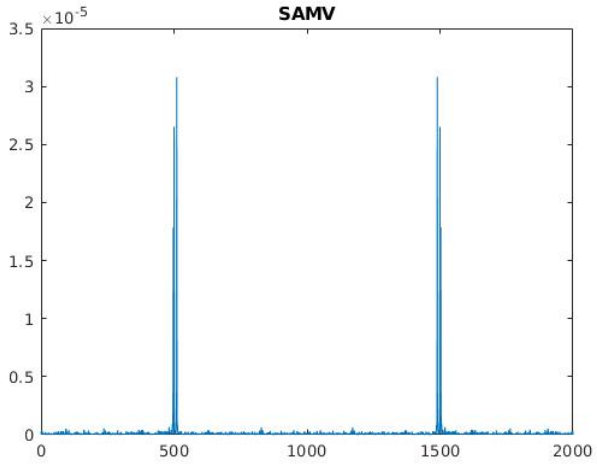

Fig. 14: Result for $f1 = 497hz$, $f2 = 500hz$ and $f3 = 510hz$ using SAMV

ESPRIT returns the following frequencies $[hz]$ (their negative symmetric and the ones higher than $10khz$ are not shown) :

- clean : 126, 282, 623, 972, 1270, 1631, 2179, 2537, 2821, 3268, 3516, 3750
- noisy : 0, 196, 360, 840, 1119, 1439, 2025, 2407, 2721, 3083, 3426, 3677

We ran into some troubles applying SAMV. Even though we cut the signal length to $280'000$ samples (from almost $2'000'000$), the matrices created by the algorithm are too big for the software MATLAB. To counter that, one could take samples of length ~ 1000 for each note and apply the algorithm to each of the selected samples to get the results. We are confident that it would work since SAMV does not need as much samples as ESPRIT or MUSIC and get better results.

IV. Discussion

After having implemented and compared the results from the three algorithms MUSIC, ESPRIT and SAMV, we can make a few observations. They all seem to have different running times, from fastest to slowest, the order is ESPRIT then our implementation of MUSIC, pmusic, and finally our implemention of SAMV. While MUSIC and ESPRIT take less than a second, SAMV runs for more than three minutes to give back results. Note that we can reduce the running time of SAMV to a few seconds by reducing the number of samples and we still get almost the same results. We might not have been perfectly efficient in our code but we can notice that MUSIC, for the same size of covariance matrix takes longer than ESPRIT for not as good results.

pmusic is also quite slow but outputs a full pseudospectrum estimate which have its utility in certain cases.

Finally SAMV is by far the slowest (using the same number of samples), but it outputs more information. In our case we do not need all of them but it could be useful for other applications, such as DOA. To counter this effect, we can reduce the number of samples and we still obtain much better results than with ESPRIT or MUSIC.

Also, MUSIC and ESPRIT are based on assumptions previously made. If these are correct the output results will give quite good estimations, on the other hand if the assumptions were false, we would have wrong frequencies too.

Data-wise, MUSIC seems to need more than the other two algorithms. This is not an advantage to it as we might not always have enough and we would still like to have performing results.

Without decorrelation preprocessing, SAMV is robust to difficulties such as insufficient snapshot or data, and has superior resolution.

V. Conclusion

To conclude, depending on the need it seems that ESPRIT should become the go-to algorithm for line spectra estimation. It is the quickest and most reliable for instantaneous results. If the need of more precision, or more information, one might lean toward the SAMV algorithm which has more applications.

The MUSIC algorithm was one of the first method for estimation but it seems that it cannot compete with more recent ones nowadays.

In this project we focused on the precision of the presented algorithms. Another interesting approach would be to optimise the relationship between the running time and the precision of the results. For MUSIC and ESPRIT, since the running time depends mainly on the size of the autocorrelation matrix (which we put at 5 times the estimated number of spectral lines) it would be interesting to find the optimal size for the best results with an acceptable running time. For SAMV, the running time greatly depends on the size of the vector that contains the possible frequencies and the number of samples. So, modifying our version of SAMV to reduce the size of this frequency vector and finding an optimal value for the number of samples needed would be interesting.

REFERENCES

[1] Habti Abeida, Qilin Zhang, Jian Li, and Nadjim Merabtine. Iterative sparse asymptotic minimum variance based approaches for array processing. *IEEE Transactions on Signal Processing*, 61(4):933–944, Feb 2013.

[2] M.H. Hayes. *Statistical Digital Signal Processing and Modeling*. Wiley, 1996.

[3] S. M. Kay and S. L. Marple. Spectrum analysis—a modern perspective. *Proceedings of the IEEE*, 69(11):1380–1419, 1981.

[4] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3), 1955.

[5] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. Prentice-Hall, Inc., USA, 1996.

[6] R. Roy, A. Paulraj, and T. Kailath. Estimation Of Signal Parameters Via Rotational Invariance Techniques - ESPRIT. In Jeffrey M. Speiser, editor, *Advanced Algorithms and Architectures for Signal Processing I*, volume 0696, pages 94 – 101. International Society for Optics and Photonics, SPIE, 1986.

[7] P. Stoica and R.L. Moses. *Spectral Analysis of Signals*. Pearson Prentice Hall, 2005.