Project Assignment 2: Report

Johannes Lemonde 960911-T357 and Lucas Streit 970606-T232

I. Introduction

The main concern of this project is to decode a signal which was distorted by an unknown time-invariant finite impulse response filter and which includes also some white noise. Such filters are commonly used by communication channels; our role here is to take a glimpse into the maths behind the receptor – in other words, to elaborate the algorithm able to decode this distorted signal back to the original signal.

We are provided with such a distorted signal consisting of a large sequence of real numbers, knowing that the original binary signal – before it was distorted – starts with a given binary sequence, which we also know.

The knowledge of this sequence at the beginning of the signal is needed by the receptor to compute a set of constant parameters, comparing the original and distorted signals – through a system of linear equations to solve –, which would then be used to determine the further bits of the original signal.

To make this project fancier, we are dived into a situation where the original signal is a key to decipher a picture. We are given an encrypted picture and the ciphering/deciphering algorithms, and the better the recovered key, the more accurate the deciphered picture.

II. Formalisation of the problem

The original binary signal – the key – is composed by a sequence $b(k) \in \{-1, 1\},\ \ k \in [1, N] \subset \mathbb{N}$, where N is the length of the key.[1] The values $b(1),\ \ldots\ , b(M)$ – referred to as the training sequence – are known to us, with $M = 32$. Before it was transmitted, the key has been subjected to an unknown filter h of order 3 and to the white noise $n(k)$ such as

$$r(k) = \sum_{l=0}^{3} h(l)\ b(k-l)\ + n(k), \quad k \in [1, N] \subset \mathbb{N}. \tag{1}$$

These values of r are known to us, and we are asked to reconstruct the sequence b.

[1] Actually, the key is originally composed by a sequence $s(k) \in \{0, 1\}$, but we map it with the sequence $b(k) \in \{-1, 1\}$.

III. Resolution

In order to do so, we apply an equaliser (an other filter) of order L such as

$$\hat{b}_r(k) = \sum_{l=0}^{L} w(l)\ r(k-l) \approx b(k), \tag{2}$$

where the coefficients $w(0),\ \ldots\ , w(L+1)$ must be initialised using the training sequence. To do so, we define a matrix $\boldsymbol{R}$, a vector $\boldsymbol{w}$ and a vector $\boldsymbol{b}$ so that $\boldsymbol{Rw} = \boldsymbol{b}$ represent the equations in (2). Since $r(k)$ does not have values for k being null or negative, and since we have this subtraction of l in the argument of r in (2), we can only use (2) for $k = L+1,\ \ldots\ , M$, which thus gives $\boldsymbol{Rw} = \boldsymbol{b} \Leftrightarrow$

$$\begin{pmatrix} r(L+1) & r(L) & \cdots & r(1) \\ r(L+2) & r(L+1) & \cdots & r(2) \\ \vdots & \vdots & \ldots & \vdots \\ r(k) & r(k+1) & \cdots & r(k-L) \\ \vdots & \vdots & \ldots & \vdots \\ r(M) & r(M-1) & \cdots & r(M-L) \end{pmatrix} \cdot \begin{pmatrix} w(0) \\ w(1) \\ \vdots \\ w(L) \end{pmatrix} = \begin{pmatrix} b(L+1) \\ b(L+2) \\ \vdots \\ b(k) \\ \vdots \\ b(M) \end{pmatrix} \tag{3}$$

Notice that $\boldsymbol{R}$ is not square for any value of L, so the system might have much more equations than unknowns. Thus, the most straightforward way to solve the system for $\boldsymbol{w}$ is to resolve it in the approximation of the least squares: $\boldsymbol{w} \approx (\boldsymbol{R}^T\boldsymbol{R})^{-1}\ \boldsymbol{R}^T\boldsymbol{b}$. Matlab does this automatically for non-square matrices with the command `w = R\b`.

Notice also that this least-squares solution for the coefficients $\boldsymbol{w}$ is per definition the solution which minimises the mean square error defined by $MSE = \mathbb{E}\{(b(k) - \hat{b}_r(k))^2\},\ k \in [L+1, M] \subset \mathbb{N}$. The causal finite impulse response (FIR) Wiener filter – which is an other filter – would also minimise the mean square error, but works using the autocorrelation and the cross-correlation functions of the signals, which we do not have at our disposal. An approach to use this filter anyway could be to compute estimates of these functions from the training sequence. But even if the Wiener filter is in the general case an optimal filter, using it with estimates (affected by the noise) makes it no much better than the least-squares approach we used. Indeed, the matrix $(\boldsymbol{R}^T\boldsymbol{R})$ in the least-squares

approach is itself an approximate of the matrix of autocorrelation which would be used in the Wiener filter, and the same applies for $(\boldsymbol{R}^T\boldsymbol{b})$ which is an approximate of the vector of cross-correlation. As a reminder, the Wiener filter would have been used like this: $\boldsymbol{w} \approx \boldsymbol{T}_{acf}^{-1}\boldsymbol{v}_{cross}$. Unfortunately, neither approach gives a perfect, optimal solution, due to the noise $n(k)$ present in the signal $r(k)$. A training sequence of arbitrarily short, finite length $M = 32$ does indeed not permit the effects of the noise to converge towards zero.

Once these coefficients $\boldsymbol{w}$ are computed, we can freely use the equation (2) to get the estimated values $\hat{b}_r(k)$ of the whole signal (although for $k \leq M$, and especially for $k \leq L$ where the equation cannot be used, we use the known training sequence directly, instead), but we still need to convert them to a binary sequence. In order to do so, we apply a sign detector on the real-valued estimator of b, which gives: $\hat{b}(k) = sign(\hat{b}_r(k)) = \begin{cases} +1, & \hat{b}_r(k) > 0 \\ -1, & \hat{b}_r(k) \leq 0 \end{cases}$.

IV. Optimisation of the equaliser's order L

We have now everything needed to reconstruct the key, apart from L. Indeed, the accuracy of the recovery of the key depends on the choice made for the order of the equaliser.

On the first hand, having a really low order L for the equaliser would be insufficient to cancel out the white noise (see figure 1). Indeed, the expected value of the white noise tends towards zero if we consider a great amount of its realisations, so increasing L would result in a better cancellation of the noise. Whereas the effects of the unknown filter h in (1) are much easier to restore, even with a quite low order equaliser.

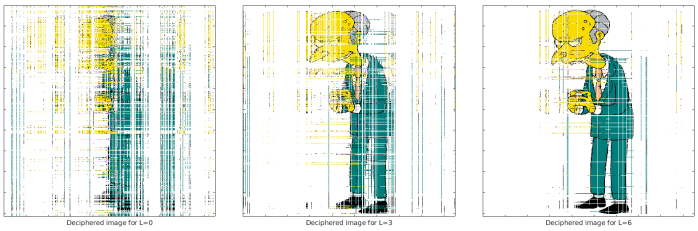

Fig. 1. The deciphered image shows *Mr. Burns*, a character out of the American animated sitcom *The Simpsons*. Here, it was deciphered with an equaliser order L of respectively 0 (no equaliser), 3 and 6.

On the other hand, increasing the order L results in decreasing the amount of linear equations and thus the amount of values of the training sequence we are going to be able to use. Indeed, the amount of equations is equal to $M-L$ while the amount of unknowns is $L+1$.

For a great order $L > \frac{M-1}{2} = 15.5$, the system of equations is under-determined and this results in very few values $b(k)$ determining a large amount of coefficients $w(k)$ as if one tried to project a vector into a greater dimension as its own: an infinity of solutions exist. Even if the training sequence $b(k)$ for $k = L+1, \ \ldots \ , M$ can be recovered perfectly with $\hat{b}_r(k)$, the extrapolation on further values of the key does not work. In this case, the key contains to many errors to decipher the image accurately (see third image in figure 2).

For $L < \frac{M-1}{2} = 15.5$, the system of equations is over-determined and its solution is in the sense of the least-squares. Considering the fact that the equations around the training sequence are not necessarily perfectly representative of the following of the key, due for example to aberrant noise values misleading the calculation of the coefficients $w(k)$, it seems to be an advantage to have (some) degrees of over-determination.

Empirically, the accuracy of the key recovery is the best for $L = 12$ or $L = 13$, quite good for $L \in [7, 14] \subset \mathbb{N}$, while the image is only just recognisable for $L = 15$.

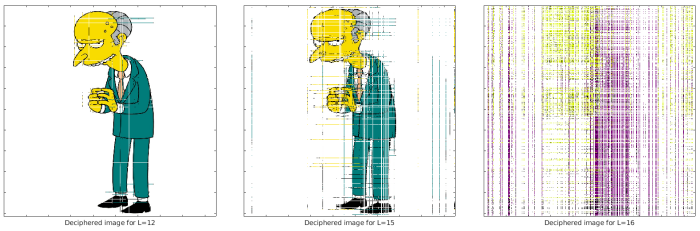

Fig. 2. Here, the image was deciphered with an equaliser order L of respectively 12, 15 and 16.

While trying to generate a new sequence $r(k)$ according to the equation (1), using to do so an arbitrary original key $b(k)$, a filter set arbitrarily to

$$h(l) = \begin{cases} 1 & \text{if } l = 0, \\ 0.7 & \text{if } l \in \{1, 2, 3\}, \\ 0 & \text{else.} \end{cases}$$

and white noise $n(k) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$, we obtain for the mean square error $MSE = \frac{1}{N}\sum_{k=1}^{N}(\hat{b}(k) - b(k))^2$ in function of the equaliser's order L the graph shown in figure 3, which seems to confirm our suspicions. Obviously, we did not have at our disposal the actual function h, nor the actual realisations of the Gaussian white noise, nor even its variance, and according to the many invocations of randomness during the generation of the noise and the key (random ciphering of the image), the shape of this graph does change a lot from an execution to another. However, the amount of errors

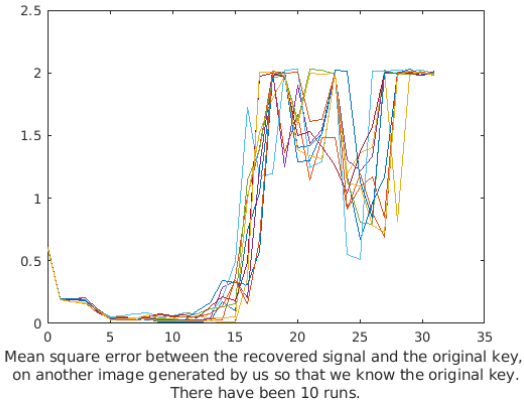

Fig. 3. The MSE increases drastically from the point where the system is under-determined ($L \geq 16$).

seems to be always drastically higher for $L > \frac{M-1}{2}$, while it remains low for an order L between 1 and 15, especially in the upper middle of this range.

V. Random bit error insertions

In this section of the report, we are going to discuss the impact of the insertion of bit errors upon the quality of the image recovery. In the general case, this depends much on the algorithm in use to cipher/decipher the image as some algorithms could make it very hard to recover anything from the fist bit error.

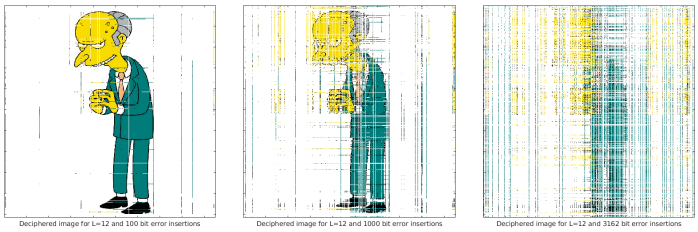

Fig. 4. Here, the image was deciphered with an equaliser order $L = 12$ and 10^2, 10^3 and $10^{3.5}$ bit errors were inserted, respectively.

Nevertheless, in this specific case, the algorithm seems to traduce *linearly* the amount of inserted bit errors as amount of permutations of lines, rows or colours of pixels.[2]

Using this, we can visually suppose that our performance recovering the key with $L = 12$ gave about 10^2 bit errors, and with $L = 15$ it was less than 10^3 errors, for a key length of 19712; in other words, respectively about 99.5% and more than 95% of accuracy were obtained.

Moreover, we find that the limit to recognise the subject of the image lies around these 10^3 bit errors, corresponding to 95% of accuracy.

[2]Images are seen here as a tridimensional matrix where the third dimension is the intensity of red, green and blue of the pixels. The ciphering/deciphering algorithm swaps basically plans within the matrix representing the image. Since the third dimension is only made by three plans while the other two dimensions are made up by as many plans as the image's amount of row or columns, and since the algorithm seems to pick uniformly which permutation it is about to perform, the probability to swap colours is quite low. Here, the matrix' dimensions are 1015x950x3.

VI. Conclusion

With this project, we have been able to understand the basic functioning of some kind of communication channels and have taken a closer look at channel distortion through a finite impulse response filter as well as linear equalisers. Since this equaliser was linear – rather than optimal – it was as expected not possible to recover the key perfectly. But it was still possible to get quite a good result so it was achievable to recognise the subject on the deciphered picture.

References

[1] P. Handel, R. Ottoson, H. Hjalmarsson, *Signal Theory*, KTH, 2012
[2] Wikipedia: https://en.wikipedia.org/wiki/Wiener_filter
[3] Wikipedia: https://en.wikipedia.org/wiki/Linear_least_squares