The Smart Kalman Filter: A Deep Learning-Based Approach for Time-Varying Channel Estimation

Antoine Siebert[1,2], Guillaume Ferré[1], Bertrand Le Gal[1], Aurélien Fourny[2]
[1]Univ. Bordeaux, CNRS, Bordeaux INP, IMS, UMR 5218, F-33400, Talence, France
[2]THALES, France
Email: forename.name@{ims-bordeaux.fr[1], thalesgroup.com[2]}

***Abstract*—In digital wireless communications, the received signal can be strongly altered by the environment and may contain Inter-Symbol Interference (ISI). To remove or reduce the ISI i.e. equalize, the impulse response of the propagation channel can be estimated. The Kalman Filter (KF) is an inescapable estimation algorithm in linear systems because of its optimality in terms of Minimum Mean Square Error (MMSE) under certain assumptions. However, in real conditions, implementations of KF are often difficult because of the necessity of hand-tuning parameters. In this paper, we present the Smart Kalman Filter (SKF), a hybrid architecture that combines a KF and the power of neural networks to extract relevant features from data to benefit from an adaptive KF that is automatically well tuned over time. We demonstrate in this paper that the proposed SKF is up to 5dB better at low Signal-to-Noise Ratio (SNR) and 3dB better at high SNR than Least Square (LS) algorithm in a time-varying channel estimation context with abrupt Doppler frequency variations.**

***Index Terms*—Channel Estimation, Artificial Intelligence, Kalman filter, Neural Network**

I. Introduction

During a wireless digital communication, the environment plays a major role. Indeed, due to the physical effects such as reflection, diffraction, or scattering, the receiver gets at each time an observation composed of multiple signals with different power delay profiles. This is called multipath and an illustration is given in Fig. 1. At any time, the observation sampled at the symbol time can be written as:

$$\begin{aligned} z(m) &= (\mathbf{h} * \mathbf{s})(m) + v(m) \\ &= \sum_{p=0}^{L-1} h(p)s(m-p) + v(m) \end{aligned} \tag{1}$$

with L the number of paths, $\mathbf{s}$ the transmitted symbols, $\mathbf{h}$ the Channel Impulse Response (CIR), and v an additive noise.

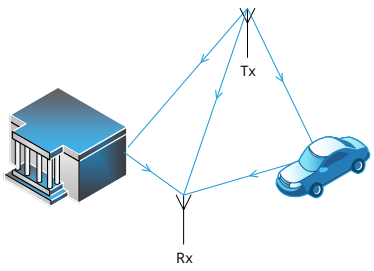

Fig. 1. Illustration of an equivalent discret channel composed of $L = 3$ paths

The multipath creates Inter-Symbol Interference (ISI), and in order to correctly estimate each symbol, it has to be reduced or removed. This process is called equalization. To equalize the received signal, the propagation channel, which represents the environment, has to be estimated. The propagation channel can be seen as a filter entirely characterized by its impulse response $\mathbf{h}$. Pilot symbol sequences are typically used to estimate the Channel Impulse Response (CIR), i.e. symbols known to both the transmitter and the receiver. Different algorithms can be used to estimate the coefficients of the CIR, such as the Least Squares (LS) algorithm [1], Least Mean Squares (LMS), Recursive Least Squares (RLS) [2] or others [3], [4]. The effect of the channel being seen as a mathematical convolution which is a linear operation, it is natural to turn to the Kalman Filter (KF) [5]. The KF is an inescapable estimation algorithm for tracking parameters of linear systems, and it is optimal in term of Minimum Mean Square Error (MMSE) under certain assumptions. A lot of applications are based on it, such as in ballistic for target tracking [6]. Although channel estimation with KF has already been proposed [7], [8], this algorithm was not used in real systems because of the need to hand-tune parameters. A KF that is well set up in one case will not necessarily be in another. To get around this issue, Interacting Multiple Model (IMM) was developed [9]. The idea is to combine multiple KF with different parameters to take advantage of the performance of the KF in multiple cases, in exchange for a higher computational complexity. But, the issue of hand-tuned parameters still remains.

The physical layer has not escaped to the advent of deep learning of the last few years in term of modulation recognition [10], coding/decoding [11], etc. Also, neural networks have been used to estimate the CIR [12], [13]. Particularly, Recurrent Neural Networks (RNN) have been shown great efficiency to extract relevant features from data and understand complex patterns. Hence, the combination of a KF with the deep learning to create a hybrid algorithm could make the use of KF viable in real-world systems as neural networks adjust the parameters over time.

The proposed receiver architecture based on this idea, called the Smart Kalman Filter (SKF) is described in the Section II. Then, the different methods used to estimate the KF parameters over time with neural networks are described in Section III. In Section IV, the proposed approach is evaluated on multipath Rayleigh fading channels with abrupt Doppler

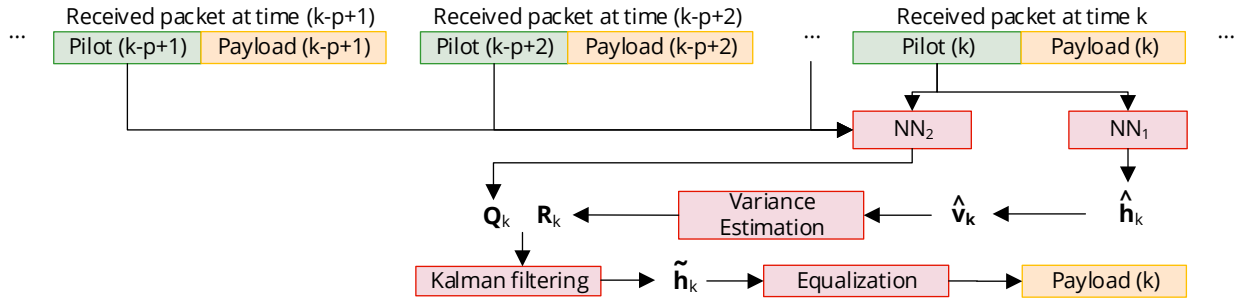

Fig. 2. Principle of the proposed Architecture: The Smart Kalman filter

frequency variations, which corresponds to our military context. For example, it can represent a mobile carrier who evolves in a rapidly changing environment. Then, the performance is compared to IMM and LS estimators. Section V gives conclusion and introduces future works.

II. Presentation of the Proposed Architecture

This section starts with a contextualization of the problem, then the architecture of the SKF is presented and the associated state model is detailed.

A. Notations and contextualization

In this paper we consider the following notations:

- $\mathbf{x}$ denotes a vector or a matrix,
- $\mathbf{x}_k$ denotes the state of the vector $\mathbf{x}$ at the time k,
- $x(n)$ denotes the n^{th} coefficient of the vector $\mathbf{x}$,
- $x(i,j)$ denotes the coefficient at position (i,j) of the matrix $\mathbf{x}$,
- $\mathbf{x}^\mathsf{T}$ denotes the transpose of $\mathbf{x}$,
- L is the length of the CIR,
- N is the length of the estimators of the CIR, $N \geq L$,
- N_p is the number of pilot symbols in each packet.

Let consider a linear system which evolves from the time $k-1$ to k according to the following equation:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{2}$$

with $\mathbf{x}_k$ the state vector, $\mathbf{F}_k$ the state-transition matrix model from time $k-1$ to k, $\mathbf{B}_k$ is the control-input model applied to the control vector $\mathbf{u}_k$, and $\mathbf{w}_k$ the process noise. The KF is known for its capacity to track time-varying process parameters, and can be used to estimate $\mathbf{x}$ over time. The equations of the linear KF are below and divided into two parts. The prediction equations are:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \tag{3}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\mathsf{T} + \mathbf{Q}_k \tag{4}$$

With $\hat{\mathbf{x}}_{k|k-1}$ the predicted state vector at the time k, $\hat{\mathbf{x}}_{k-1|k-1}$ is the updated state vector at the time $k-1$ given observations up to and including at the time $k-1$. $\mathbf{P}_{k|k-1}$ is the estimated covariance matrix. The matrix $\mathbf{Q}_k$ is the covariance matrix of the process noise given by $\mathbf{Q}_k = \mathbb{E}[\mathbf{w}_k \mathbf{w}_k^\mathsf{T}]$. The update equations of the linear KF are:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \tag{5}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k \tag{6}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\mathsf{T} \mathbf{S}_k^{-1} \tag{7}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{8}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_{k|k-1} \tag{9}$$

In these equations, $\mathbf{z}_k$ is the observation of the process at the time k, $\mathbf{H}_k$ is the observation model and $\tilde{\mathbf{y}}_k$ is called the innovation. $\mathbf{S}_k$ is the covariance matrix of the innovation, $\mathbf{R}_k$ is the covariance matrix of the measurement noise. $\mathbf{K}_k$ is called the Kalman gain, and $\mathbf{I}$ is the identity matrix. One of the main issues of this filter in real systems is: as $\mathbf{Q}$ has no physical sense, most of the time it has to be hand-tuned and fixed for the use case. The higher the coefficients of the $\mathbf{Q}$ matrix, the larger the variations of the predicted state vector. If the coefficients of $\mathbf{Q}$ are too high and the real parameters vary slightly, state vector will vary greatly and the tracking will not be optimal. On the contrary, if the coefficients of $\mathbf{Q}$ are too small while the process occurs large variations, the convergence of the parameters will be slow and errors may occur. So in this paper, we propose a method based on deep learning to estimate $\mathbf{Q}_k$ and $\mathbf{R}_k$ matrices at each time k to get an adaptive KF.

B. Principle of the architecture

The main objective of the proposed SKF architecture is to estimate the CIR with a KF aided by two neural networks called NN_1 and NN_2 in the rest of the paper. For this purpose, two parameters need to be estimated at each time k: the covariance matrix of the measurement noise $\mathbf{R}_k$, and the covariance matrix of the process noise $\mathbf{Q}_k$. The estimation is made based on the last p pilot symbol sequences received of length N_p. In addition, the transmitted pilot symbol sequence of N_p symbols is the same for all the packets. It works as follows:

- The first neural network NN_1 in Fig. 2 is used to get $\hat{\mathbf{h}}$, a first estimation of the CIR. This pre-estimation of the channel is obtained from the received pilot sequence at

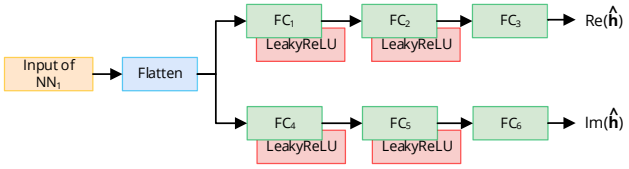

Fig. 3. Architecture of the first neural network (NN_1) that estimates $\mathbf{h}$

time k and is then used to estimate the variance of the additive noise in order to obtain $\mathbf{R}_k$.

- The second neural network NN_2 in Fig. 2 uses the p last received pilot symbol sequences to predict $\mathbf{Q}_k$ in order to get an adaptive time-varying matrix to better track the CIR.

Once the matrices $\mathbf{R}_k$ and $\mathbf{Q}_k$ are estimated at each time, the well setup KF can be used to provide $\tilde{\mathbf{h}}$, the second estimation of the CIR, and the equalization can be carried out.

C. Definition of the state model

Let $\mathbf{s}$ be the transmitted symbol pilot sequence (M-PSK, M-QAM, etc.), and $\mathbf{h}$ the impulse response of the channel. The observation of the received symbol pilot sequence at the time k is given by:

$$\mathbf{z}_k = (\mathbf{s} * \mathbf{h}_k) + \mathbf{v}_k \tag{10}$$

with $\mathbf{v}_k$ an additive noise such as $\mathbf{v}_k \sim \mathcal{CN}(0, \sigma_v^2 \mathbf{I}_{N_p})$, and N_p the number of pilot symbols per packet. As the objective is to estimate the CIR $\mathbf{h}$, then let:

$$\mathbf{x_k} = [h_k(L-1), h_k(L-2), \ldots, h_k(0)]^\mathsf{T} \tag{11}$$

be the state vector corresponding to the L complex coefficients of the CIR at time k. Then, the matrix $\mathbf{H}_k \in \mathcal{M}_{N_p,L}(\mathbb{C})$ is defined as $\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$. More explicitly, $\mathbf{H}_k$ is the toeplitz matrix given by:

$$\mathbf{H}_k = \begin{pmatrix} s(N-L+1) & \ldots & s(N-1) & s(N) \\ \vdots & & \vdots & \vdots \\ s(0) & \ldots & s(L-2) & s(L-1) \\ 0 & \vdots & s(L-3) & s(L-2) \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & s(0) & s(1) \\ 0 & \ldots & 0 & s(0) \end{pmatrix} \tag{12}$$

In the CIR estimation context, for all $k \in \mathbb{N}$, the control-input matrix $\mathbf{B}_k$ is a null matrix, and $\mathbf{P_0}$ is taken equal to the identity matrix. Finally, the model evolution matrix $\mathbf{F}$ is taken equal to identity matrix because the evolution of the CIR is not known. In order to use the KF, it remains to define the matrices $\mathbf{R}_k$ and $\mathbf{Q}_k$.

III. Estimation parameter methods

A. Estimation of $\mathbf{R}_k$

To estimate the matrix $\mathbf{R}_k$ at each time k, a first estimation $\hat{\mathbf{h}}$ of the CIR is computed by NN_1 based on pilot symbol sequence of the packet. The neural network NN_1 is a Multilayer Perceptron (MLP) neural network and its architecture is illustrated in Fig. 3. It takes as input the real and the imaginary parts of the received and the transmitted pilot symbol sequences, i.e. the input size is $2 \times 2 \times N_p$. It gives as output an estimation of the real part and the imaginary part of the CIR, each one of size $1 \times N$ with $N \geq L$. The neural networks first consists of a flattening layer to make the input a vector. After that, there are two blocs of three fully connected layers activated with LeakyReLU function. The LeakyReLU is a function used to get a faster training of the neural network and defined as:

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.01x & \text{else} \end{cases} \tag{13}$$

At time k, the impulse response of the channel $\hat{\mathbf{h}}_k$ is obtained, and then the estimation of the noise

$$\hat{v}(i) = (\mathbf{s} * \hat{\mathbf{h}}_k)(i) - \mathbf{z}_k(i) \tag{14}$$

is computed for $i \in \{0, \ldots, N_p - 1\}$. The variance of the estimated noise can be obtained with the common unbiased estimator

$$\sigma_{\hat{v}}^2 = \frac{1}{N_p - 1} \sum_{k=0}^{N_p-1} \left[(\mathfrak{Re}(\hat{v}(k) - \bar{v}))^2 + (\mathfrak{Im}(\hat{v}(k) - \bar{v}))^2 \right] \tag{15}$$

with $\bar{v}$ the mean of the noise $\hat{v}$, and N_p the length of the pilot. The matrix $\mathbf{R}_k$ is then construct as $\mathbf{R}_k = \sigma_{\hat{v}}^2 \mathbf{I}$ and can be computed at each time if necessary.

B. Estimation of $\mathbf{Q}_k$

The parameter $\mathbf{Q}_k$ is the covariance matrix of the process noise, and it represents the degree of variation of the state vector at each step. By considering the independence between the different coefficients of the CIR, the covariance matrix $\mathbf{Q}_k$ can be written as

$$\mathbf{Q}_k = Diag\left(\sigma_0^2, \ldots, \sigma_{N-1}^2\right) \tag{16}$$

For each $i \in \{0, \ldots, N-1\}, \sigma_i^2$ represents the variance of the i^{th} coefficient. In order to estimate $\mathbf{Q}_k$ over time, a second neural network is used and its architecture is illustrated in Fig. 4. The network consists in a single Gated Recurrent Unit (GRU) [14] layer, followed by three fully connected layers activated with sigmoid function after each layer. This architecture is based on [15] with the Long-Short-Term-Memory (LSTM) neural network replaced by a GRU for computational complexity reduction. At time k, the output of the neural network is the diagonal of $\mathbf{Q}_k$. In order to take into account

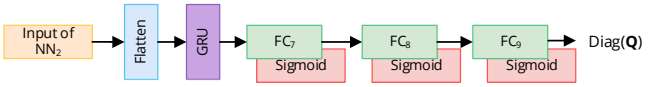

Fig. 4. Architecture of the second neural network NN_2 that estimates $\mathbf{Q}$

the channel evolution, the input of the neural network is a vector composed of the $p \geq 2$ last received pilot sequences, that is a vector of size $2 \times p \times N_p$. With the computation of $\mathbf{R}_k$ and $\mathbf{Q}_k$, the channel can be estimated with the KF at each received packet.

IV. Numerical Results and Discussion

Simulations were conducted to evaluate the performance of the proposed architecture. The simulation parameters are introduced in this section. Then, the dataset used for the different trainings of NN_1 is described and the obtained performance is compared with a CIR estimation made by LS algorithm. Finally, the training of NN_2 is presented and the performance of the proposed Smart Kalman filter is compared with the performance of IMM and LS algorithms on Rayleigh fading multipath channels.

A. Simulation parameters

For the following simulations, packets of size 256 are transmitted. Each packet contains a pilot sequence of $N_p = 5$ symbols known by the receiver, which represents less than 2% of the packet. Associated to the pilot sequence, 251 symbols of data are inserted into each packet. The QPSK digital modulation is used. The CIR is constant inside a packet. At the receiver side, a perfect time and frequency synchronizations are assumed. A linear MMSE equalizer of length 15 is used. Concerning the number of pilot symbol sequences used in order to estimate the matrix $\mathbf{Q}_k$ over time, $p = 2$ was fixed. Indeed, some tests were performed with $p \geq 3$ but this did not improve the performance. The IMM used in the last result part is composed of 4 KF with parameters described in Table I.

TABLE I
KF parameters of the IMM

	$\mathbf{R}$	$\mathbf{Q}$
KF1	$10^{-1}\mathbf{I_{N_p}}$	$10^{-1}\mathbf{I_{N_p}}$
KF2	$10^{-1}\mathbf{I_{N_p}}$	$10^{-3}\mathbf{I_{N_p}}$
KF3	$10^{-3}\mathbf{I_{N_p}}$	$10^{-3}\mathbf{I_{N_p}}$
KF4	$10^{-3}\mathbf{I_{N_p}}$	$10^{-1}\mathbf{I_{N_p}}$

The two additional IMM parameters are given by:

$$\mu_0 = [0.25, 0.25, 0.25, 0.25] \tag{17}$$

$$\mathbf{\Pi} = \begin{pmatrix} 0.94 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.94 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.94 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.94 \end{pmatrix} \tag{18}$$

The vector μ is the mixing probability vector which represents the weights of the different state vectors in the final output of the IMM. The matrix $\mathbf{\Pi}$ is the transition probability matrix. This choice of parameters allows defining a balanced IMM for low and high SNR, and for slow and fast channel variations.

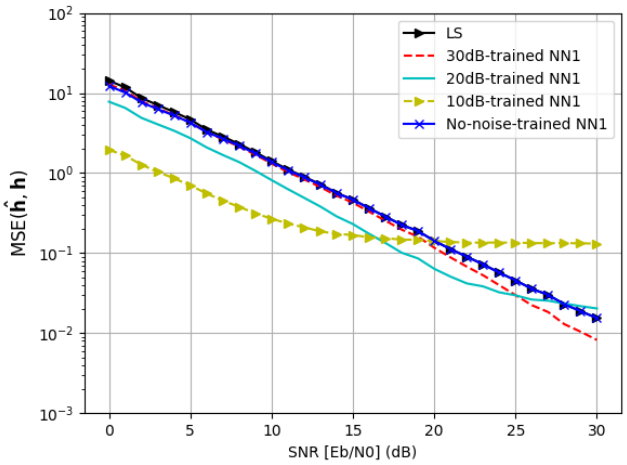

Fig. 5. MSE($\hat{\mathbf{h}}, \mathbf{h}$) vs. SNR for different SNR training levels

B. Results of the estimation of $\mathbf{R}_k$

Firstly, the purpose of NN_1 is to provide at each time $\hat{\mathbf{h}}$, a first estimation of $\mathbf{h}$. To be as general as possible, the training dataset has to be composed of many different CIR, either in channel depth or in coefficients amplitude. The dataset used for the training of NN_1 is composed of 5×10^4 vectors of size $1 \times N$. Each coefficient of the vectors is generated according to a complex Gaussian distribution (μ=0, σ^2=1) representing the CIR.

The length L of the CIR is uniformly distributed between one and five, and then the vector is zero padded to get a size of $N = 5$. The zero padding is used to have a fixed vector length to facilitate the training of the neural network. The dataset is split into two parts, 80% for training and 20% for testing. The size of the batch is 128, Adam optimizer and MSE loss are used. The MSE loss is given by:

$$MSE(\hat{\mathbf{h}}, \mathbf{h}) = \frac{1}{N} \sum_{j=0}^{N-1} (\mathfrak{Re}(\hat{h}(j) - h(j))^2 + \mathfrak{Im}(\hat{h}(j) - h(j))^2) \tag{19}$$

Concerning the structure of NN_1, the sizes of the FC layers have been chosen according to our experiments and are a compromise between performance and computational complexity. Sizes of FC_1 and FC_4 are 128, FC_2 and FC_5 are 32 and FC_3 and FC_6 are $N = 5$. Different trainings have been realized with different Signal-to-Noise Ratio (SNR) levels with $N_p = 5$. Simulation results for the propose approach are given in Fig. 5 and are compared with the performance of the LS algorithm. These results show that the neural networks trained on a specific SNR level proposes better performance than others around their training SNR level. The obtained LS performance is almost the same as the neural network trained without additive noise, which are equal or lower than the 30dB-trained neural network. At low SNR, the 10dB-trained neural network proposes a gain between 7dB and 8dB over the others neural networks. In return, the MSE of the 10dB-trained neural network seems to stop decreasing from around

15dB. So, except the neural network trained without additive noise, the three others are kept to evaluate the performance of the global architecture.

C. Results of the estimation of $\mathbf{Q}_k$

Regarding the architecture of NN_2, the sizes of the layers of this neural network were chosen empirically, based on our experiments. The size of FC_7 is 32, FC_8 is 16 and FC_9 is 5. The GRU layer is composed of 1 GRU cell with a hidden vector of size 128. The purpose of the NN_2 is to estimate at each time the covariance matrix of the process noise $\mathbf{Q}$. The neural network has to be trained on channels with both small and large variations between two consecutive packets. So this time the dataset can not be composed of completely random channels as it was the case for the training of NN_1. Indeed, channels used have to be temporally correlated. There are several such channels, but the Rayleigh channel correlated with Jakes spectrum is one of the most used in the absence of line of sight between the transmitter and the receiver. There are multiple ways to generate this type of channels, based on inverse Fourier transform, white noise filtering or based on a sum of sinusoids. For its fast implementation, the channels have been generated according to the sinusoidal method:

$$h_k(i) = \frac{1}{\sqrt{N_e}} \sum_{j=0}^{N_e-1} e^{j(\phi_{i,j}+2\pi f_d cos(\theta_{i,j})kT_s)} \qquad (20)$$

with $N_e = 32$ the number of sinusoids, and $\forall i \in [\![0; N-1]\!]$, $\forall j \in [\![0; N_e-1]\!], \phi_{i,j} \sim \mathcal{U}(0, 2\pi)$ and $\theta_{i,j} \sim \mathcal{U}(0, 2\pi)$. f_d is the maximum Doppler frequency, but in the following we will talk about the maximum normalized Doppler frequency $\bar{f}_d = f_d \times T_s$.

The training was made on a dataset of size 2×10^6, split into 80% for training and 20% for testing. Each channel of the dataset contains $L = 5$ multipaths, and $\bar{f}_d$ changes independently and uniformly on each path every 100 packets in $\{10^{-3}, 10^{-2}, 10^{-1}\}$ to create abrupt variations, observable in our military context. These variations may represent, for example a moving carrier in a rapidly changing environment. For the training, a batch size of 200 was used, with Adam optimizer and MSE loss. Different trainings have been realized with different SNR levels, at 10dB, 20dB, 30dB, and the three neural networks have been kept in order to evaluate the complete architecture. An illustration of the real part of $h_k(1)$ for $k \in [\![0; 400]\!]$ is provided in Fig. 6. It shows that at the 200^{th} packet, abrupt variations of the CIR occurred because of the maximum normalized Doppler frequency $\bar{f}_d$ varies from 10^{-1} to 10^{-3}. Same observation at the 300^{th} packet where $\bar{f}_d$ increases from 10^{-3} to 10^{-1}.

The Fig. 7 shows that when the CIR varies significantly in Fig. 6, i.e. between the 1^{st} and the 200^{th} packet, the value of the $Q_k(1,1)$ coefficient is around 10^{-1}. When $\bar{f}_d$ becomes smaller, i.e. between the 200^{th} and the 300^{th} packet, the value of $Q_k(1,1)$ decreases to 10^{-3} before increasing to around 10^{-1}. Therefore, the interest of the NN_2 is illustrated by the adaptation of the value of $\mathbf{Q}_k$ depending of the variations of the CIR.

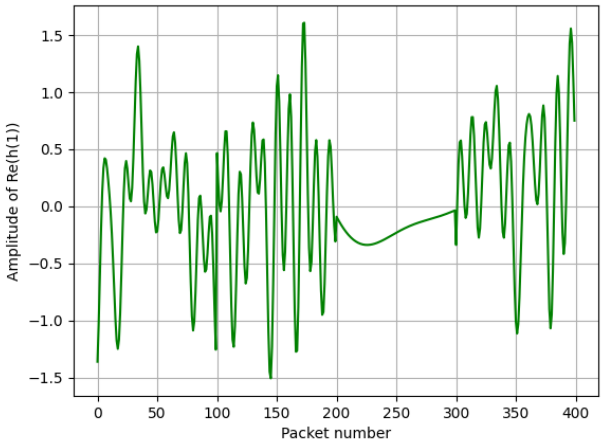

Fig. 6. Example of the evolution of the CIR coefficient $\mathfrak{Re}(h(1))$ over time

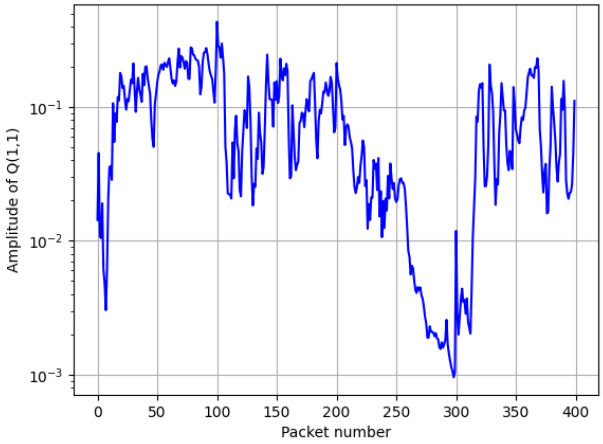

Fig. 7. Example of the evolution of $Q(1,1)$ over time at SNR = 30dB

The proposed architecture has been evaluated in term of Bit Error Rate (BER). The performance comparison is made between different estimation algorithms: the proposed Smart Kalman filter, LS and IMM. The obtained BER are given in Fig. 8. Different BER of the SKF are plotted with different SNR trainings level e.g. "SKF NN_1 20dB - NN_2 10dB" refers to the architecture where the NN_1 used is the neural network trained on a SNR of 20dB and NN_2 is trained on the 10dB dataset. The Fig. 8 shows that SKF performance depends mainly on the NN_2 trained SNR level. More precisely, the Fig. 8 (a) shows that at low SNR, SKF and IMM propose approximately the same performance, but the SKF NN_1 10dB - NN_2 10dB is 3dB better than the IMM around 15dB and offers a gain of 5dB over the LS in the low SNR range. The Fig. 8 (c) shows that a SKF trained at high SNR gives a gain of 3dB over LS and 5dB over the IMM approach at high SNR. Finally, the Fig. 8 (b) illustrates a SKF which proposes

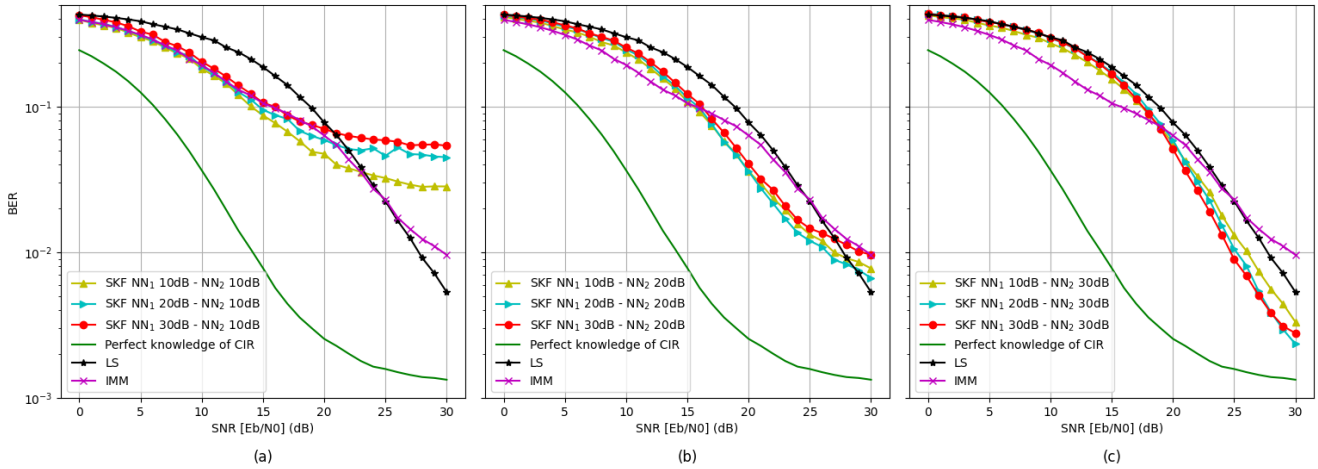

Fig. 8. BER for different CIR estimation algorithms followed by MMSE equalization. The Fig.8 (a) is obtained with NN_2 trained on 10dB dataset, Fig.8 (b) is obtained with NN_2 trained on 20dB dataset, and Fig.8 (c) is obtained with NN_2 trained on 30dB dataset.

a gain of 3dB over both LS and IMM around 20dB. Thus, the proposed architecture, the Smart Kalman filter, offers a gain between 3dB and 5dB over both LS and IMM approaches depending of the neural networks SNR training levels. The two neural networks which estimate matrices $\mathbf{R}_k$ and $\mathbf{Q}_k$ have provided an adaptive KF.

V. Conclusion

In this work, a hybrid architecture based on the Kalman filter coupled with two neural networks has been proposed for the estimation of the channel impulse response: the Smart Kalman filter. The performance was studied on Rayleigh multipath fading channels with abrupt Doppler frequency variations and is compared with the performance of the LS and the IMM algorithms. The study demonstrates the viability of the neural networks to estimate KF parameters. Indeed, on Rayleigh fading multipath channel, the Smart Kalman filter performs between 3dB and 5dB better over both the LS and the IMM algorithms. Moreover, another benefit of this architecture is the fact that the Kalman filter is automatically set, without human intervention, no matter the channel to estimate. In future works, it could be interesting to adapt this architecture for Rayleigh fast fading channel i.e. by considering the CIR varies inside a packet and stop being supposed constant. Finally, a study of the computational complexity could be done, as well as neural network optimizations in order to implement this algorithm on embedded systems.

References

[1] M. B. Sutar and V. S. Patil, "Ls and mmse estimation with different fading channels for ofdm system," vol. 1, pp. 740–745, 2017.

[2] M. Mohammadi, M. Ardabilipour, B. Moussakhani, and Z. Mobini, "Performance comparison of rls and lms channel estimation techniques with optimum training sequences for mimo-ofdm systems," pp. 1–5, 2008.

[3] P. Banelli, R. C. Cannizzaro, and L. Rugini, "Data-aided kalman tracking for channel estimation in doppler-affected ofdm systems," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 3, 2007, pp. III–133–III–136.

[4] C. Ramesh and V. Vaidehi, "Imm based kalman filter for channel estimation in uwb ofdm systems," in *2007 International Conference on Signal Processing, Communications and Networking*, 2007, pp. 320–325.

[5] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[6] K. D. Solomon Raj and I. M. Krishna, "Kalman filter based target tracking for track while scan data processing," pp. 878–883, 2015.

[7] A. Almamori and S. Mohan, "Estimation of channel state information for massive mimo based on received data using kalman filter," pp. 665–669, 2018.

[8] Y. Liao, G. Sun, Z. Cai, X. Shen, and Z. Huang, "Nonlinear kalman filter-based robust channel estimation for high mobility ofdm systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7219–7231, 2021.

[9] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[10] M. L. D. Wong and A. K. Nandi, "Automatic digital modulation recognition using artificial neural network and genetic algorithm," *Signal Process.*, vol. 84, no. 2, pp. 351–365, 2004. [Online]. Available: https://doi.org/10.1016/j.sigpro.2003.10.019

[11] I. Ortuno, M. Ortuno, and J. Delgado, "Error correcting neural networks for channels with gaussian noise," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 4, 1992, pp. 295–300 vol.4.

[12] K. Hiray and K. V. Babu, "A neural network based channel estimation scheme for ofdm system," pp. 0438–0441, 2016.

[13] L. Ge, Y. Guo, Y. Zhang, G. Chen, J. Wang, B. Dai, M. Li, and T. Jiang, "Deep neural network based channel estimation for massive mimo-ofdm systems with imperfect channel state information," *IEEE Systems Journal*, vol. 16, no. 3, pp. 4675–4685, 2022.

[14] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014. [Online]. Available: https://arxiv.org/abs/1409.1259

[15] C. Chen, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, "Dynanet: Neural kalman dynamical model for motion estimation and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5479–5491, 2021.