Homework 3: Nonlinear Channel Equalization

Kaifa Lu

Solution:

Step 1: **Generate the input *x* and desired response *y* of an adaptive filter**. Simulate the channel dynamics and generate a dataset of 5,000 data pairs using the following equations implemented in Python:

1) Desired response y of an adaptive filter:

$$y_n \sim N(0,1)$$

2) Output q of linear and nonlinear channels:

$$t_n = -0.8y_n + 0.7y_{n-1}$$

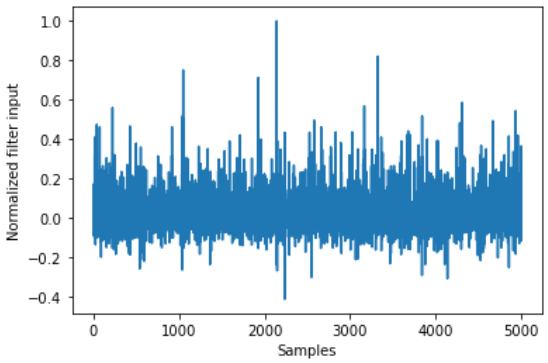
$$q_n = t_n + 0.25t_n^2 + 0.11t_n^3$$

3) Additive White Gaussian Noise (AWGN) $noise$ with 15 dB

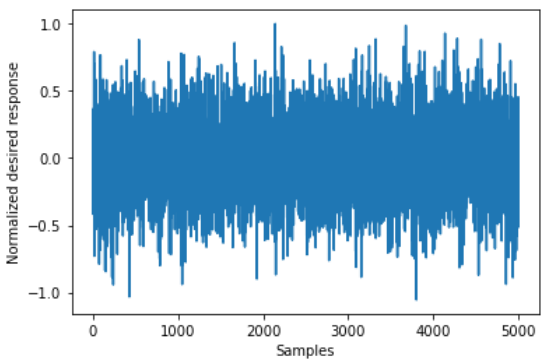
4) Input x of an adaptive filter:

$$x_n = q_n + noise$$

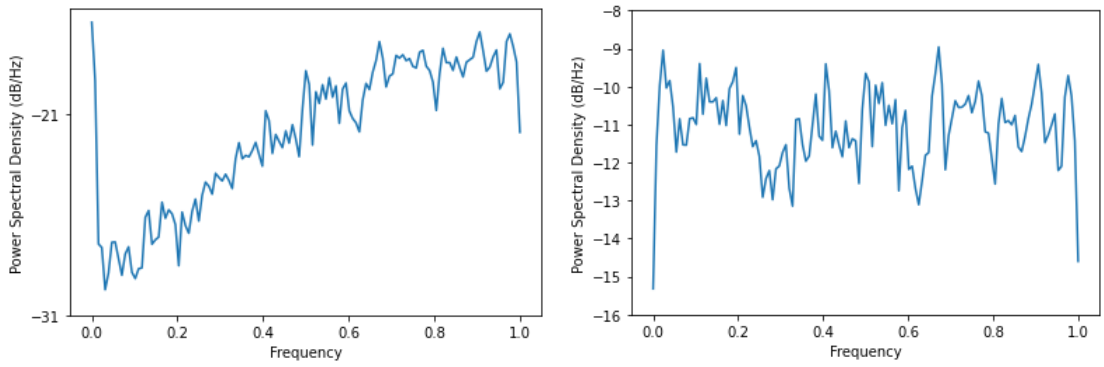
Here, we can obtain the input x and desired response y of an adaptive filter with 5000 pairs of normalized (x_n, y_{n-2}) to train the adaptive filter for nonlinear channel equalization, as shown in Fig. 1.



(1) Input samples of filter



(2) Desired responses of filter



(3) Power spectrum density of inputs (4) Power spectrum density of desired responses

Fig. 1 The input and desired response of an adaptive filter

Step 2: **Select four adaptive algorithms**. Here, we separately select least mean square (LMS), kernel LMS (KLMS), KLMS with novelty criterion (NC-KLMS) and quantized KLMS (QKLMS) as adaptive algorithms of the filters, and the pseudocodes of their implementation process are presented in Table 1.

Table 1 Pseudocodes of implementing the LMS, KLMS, NC-KLMS and QKLMS

Algorithm 1 LMS Algorithm

Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), y_{n-D})$ where $L = 5, D = 2$, and the number N=10,000

Output: the weights of filters $\boldsymbol{w} = [w_1, w_2, w_3, w_4, w_5]^T$
Initialization: step size $\eta = 0.05$, $\boldsymbol{w}_0 = [w_1, w_2, w_3, w_4, w_5]^T \sim N(0,0.1)$
While $\{(\boldsymbol{x}_i, d_i), i = 1, \cdots, N\}$ is available **do**
$e_i = d_i - \boldsymbol{w}^T \boldsymbol{x}_i$
$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} + \eta e_i \boldsymbol{x}_i$
end while

Algorithm 2 KLMS Algorithm

Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), y_{n-D})$ where $L = 5, D = 2$, and the number N=10,000

Output: the expansion $w = \sum_k^K \alpha_k \kappa(\cdot, \boldsymbol{x}_k)$ where $\alpha_k = \eta e_k$
Initialization: step size $\eta = 0.1$, Gaussian kernel size $\sigma = 0.4$, center dictionary $\boldsymbol{C} = \boldsymbol{C}_1\{\boldsymbol{x}_1\}$ and coefficient vector $\boldsymbol{\alpha} = [\eta d_1]$
While $\{(\boldsymbol{x}_i, d_i), i = 1, \cdots, N\}$ is available **do**
$e_i = d_i - \sum_{j=1}^{K_{i-1}} \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\})$
$\boldsymbol{C}_i = \{\boldsymbol{C}_{i-1}, \boldsymbol{x}_i\}$, $\boldsymbol{\alpha}_i = [\boldsymbol{\alpha}_{i-1}, \eta e_i]$
end while

Algorithm 3 NC-KLMS Algorithm

Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), y_{n-D})$ where $L = 5, D = 2$, and the number N=10,000

Output: the expansion $w = \sum_k^K \alpha_k \kappa(\cdot, \boldsymbol{x}_k)$ where $\alpha_k = \eta e_k$
Initialization: step size $\eta = 0.1$, Gaussian kernel size $\sigma = 0.4$, distance threshold $\delta_1 = 0.04$, prediction error threshold $\delta_2 = 0.04$, center dictionary $\boldsymbol{C} = \boldsymbol{C}_1\{\boldsymbol{x}_1\}$ and coefficient vector $\boldsymbol{\alpha} = [\eta d_1]$
While $\{(\boldsymbol{x}_i, d_i), i = 1, \cdots, N\}$ is available **do**

$e_i = d_i - \sum_{j=1}^{K_{i-1}} \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\})$

$dist(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}\}) = \min_{1 \le j \le K_{i-1}} \|\boldsymbol{x}_i - \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\}\|_2$

$j^* = \arg\min_{1 \le j \le K_{i-1}} \|\boldsymbol{x}_i - \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\}\|_2$

if $dist(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}\}) \le \delta_1$ **then**
$\boldsymbol{C}_i = \boldsymbol{C}_{i-1}, \ \boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i-1}$
elseif $|e_i| \le \delta_2$ then
$\boldsymbol{C}_i = \boldsymbol{C}_{i-1}, \ \boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i-1}$
else
$\boldsymbol{C}_i = \{\boldsymbol{C}_{i-1}, \boldsymbol{x}_i\}, \ \boldsymbol{\alpha}_i = [\boldsymbol{\alpha}_{i-1}, \eta e_i]$
end if
end while

Algorithm 4 QKLMS Algorithm

Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), y_{n-D})$ where $L = 5, D = 2$, and the number N=10,000

Output: the expansion $w = \sum_k^K \alpha_k \kappa(\cdot, \boldsymbol{x}_k)$ where $\alpha_k = \eta e_k$
Initialization: step size $\eta = 0.1$, Gaussian kernel size $\sigma = 0.4$, quantization threshold $\delta = 0.1$, center dictionary $\boldsymbol{C} = \boldsymbol{C}_1\{\boldsymbol{x}(1)\}$ and coefficient vector $\boldsymbol{\alpha} = [\eta d(1)]$
While $\{(\boldsymbol{x}_i, d_i), i = 1, \cdots, N\}$ is available **do**

$e_i = d_i - \sum_{j=1}^{K_{i-1}} \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\})$

$dist(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}\}) = \min_{1 \le j \le K_{i-1}} \|\boldsymbol{x}_i - \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\}\|_2$

$j^* = \arg\min_{1 \le j \le K_{i-1}} \|\boldsymbol{x}_i - \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\}\|_2$

if $dist(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}\}) \le \delta$ **then**
$\boldsymbol{C}_i = \boldsymbol{C}_{i-1}, \ \boldsymbol{\alpha}_{i,j^*} = \boldsymbol{\alpha}_{i-1,j^*} + \eta e_i$
else
$\boldsymbol{C}_i = \{\boldsymbol{C}_{i-1}, \boldsymbol{x}_i\}, \ \boldsymbol{\alpha}_i = [\boldsymbol{\alpha}_{i-1}, \eta e_i]$
end if
end while

As shown in Table 1, when implementing the four algorithms, the first step is to define the necessary parameters for the filters, and we summarize the parameters of the four algorithms in Table 2.

Table 2 Summary of parameter settings for the filters

Algorithm	Linear	Nonlinear (Gaussian kernel)		
	LMS	KLMS	NC-KLMS	QKLMS
Time delay	D = 2			
Filter order	L = 5			
Step size	0.05	0.1	0.1	0.1
Kernel size	N/A	(0.1, 0.4, 0.8, 1, 5, 10, 20)		
Delta1	N/A	N/A	0.04	0.1
Delta2	N/A	N/A	0.04	N/A

As shown in Table 2, the most important parameter for kernel methods is the kernel size that has significant impacts on the performance of nonlinear filters. Hence, we vary the kernel size of the QKLMS algorithm from 0.1, 0.4, 0.8, 1, 5, 10 to 20, to find the best kernel size for the kernel methods. As presented in Table 1, we execute the QKLMS algorithm for the set of examples:

$$\left((x_n, x_{n-1}, \cdots, x_{n-L+1},), y_{n-D}\right)$$

Where: L refers to the order of the adaptive filter and D denotes the time delay. Here, we set $L=5$ and $D=2$. We present the results corresponding to the QKLMS algorithm with different kernel size (0.1, 0.4, 0.8, 1, 5, 10, 20) in Fig. 2.

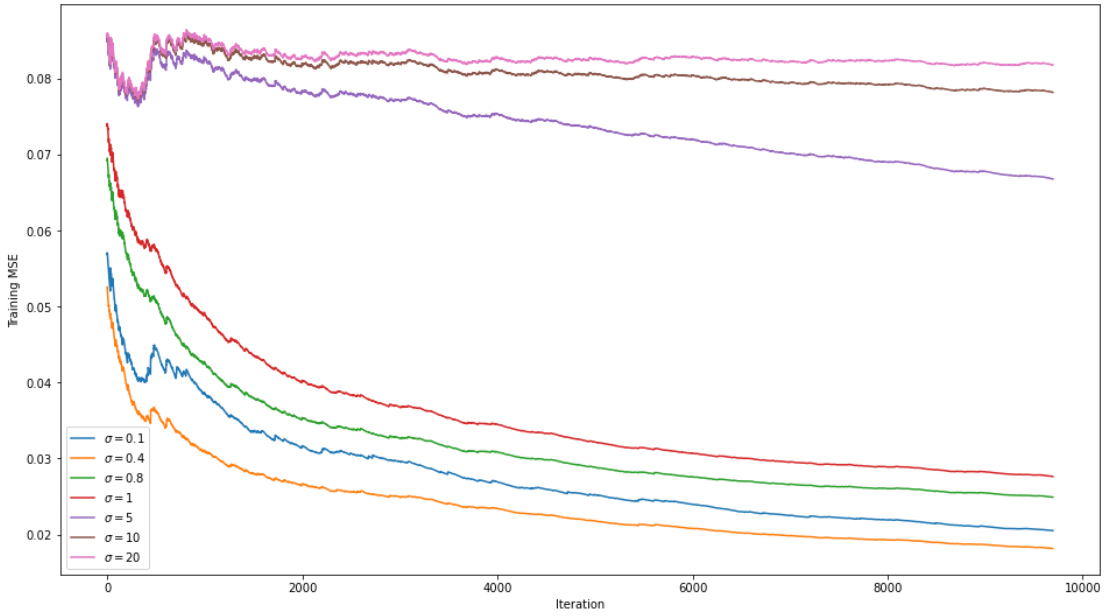
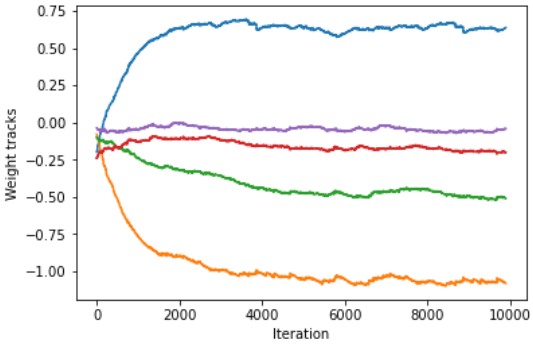


Fig. 2 The performances of the QKLMS algorithm with different kernel sizes

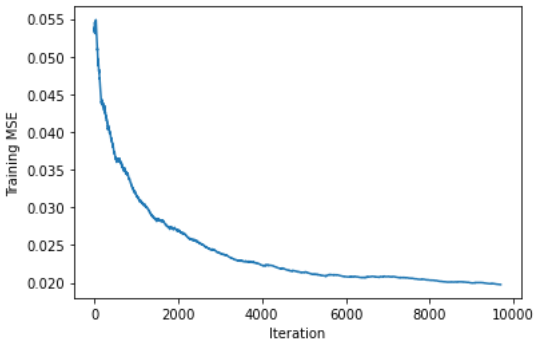
As illustrated in Fig. 2, when the kernel size of the QKLMS algorithm is equal to **0.4**, the nonlinear filter achieves better performances in addressing nonlinear channel equalization problems, with a lower training MSE between the filter outputs and desired responses, than the other six cases with the kernel size from 0.1, 0.8, 1, 5, 10 to 20. Therefore, we can say that the best kernel size for the nonlinear methods is **0.4**, just based on the experimental results in this assignment.

Step 3: **Compare the performances of LMS, KLMS, NC-KLMS, and QKLMS in nonlinear channel equalization**. The performances of linear and nonlinear filters can be reflected by the following metrics: learning curve (training MSE vs iteration) for the linear and nonlinear filters, growth curve (network size vs iteration) for the kernel methods, and weight tracks for the linear method. Specifically, we present the performances of the adaptive filters with four different algorithms as follows:

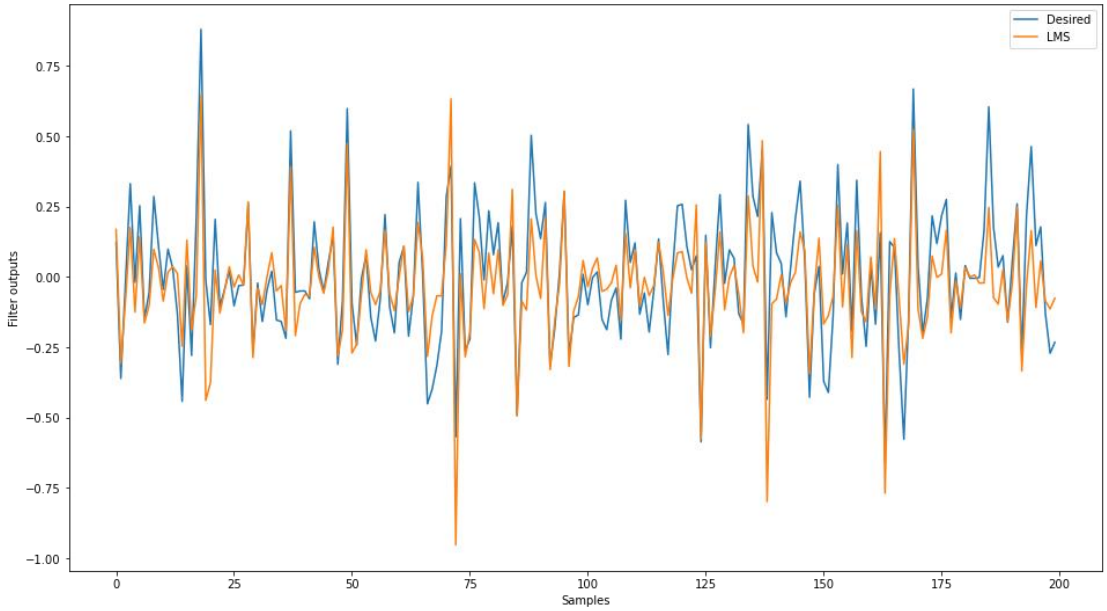
i. Linear Filters for LMS Algorithm



(1) Weight tracks



(2) Learning curve



(3) Fitting curve of filter outputs and desired responses over the first 200 samples

Fig. 3 Performances of the linear adaptive filter with the LMS algorithm

ii. Nonlinear Filters for KLMS, NC-KLMS, and QKLMS Algorithms

1) Fitting curve of filter outputs and desired responses

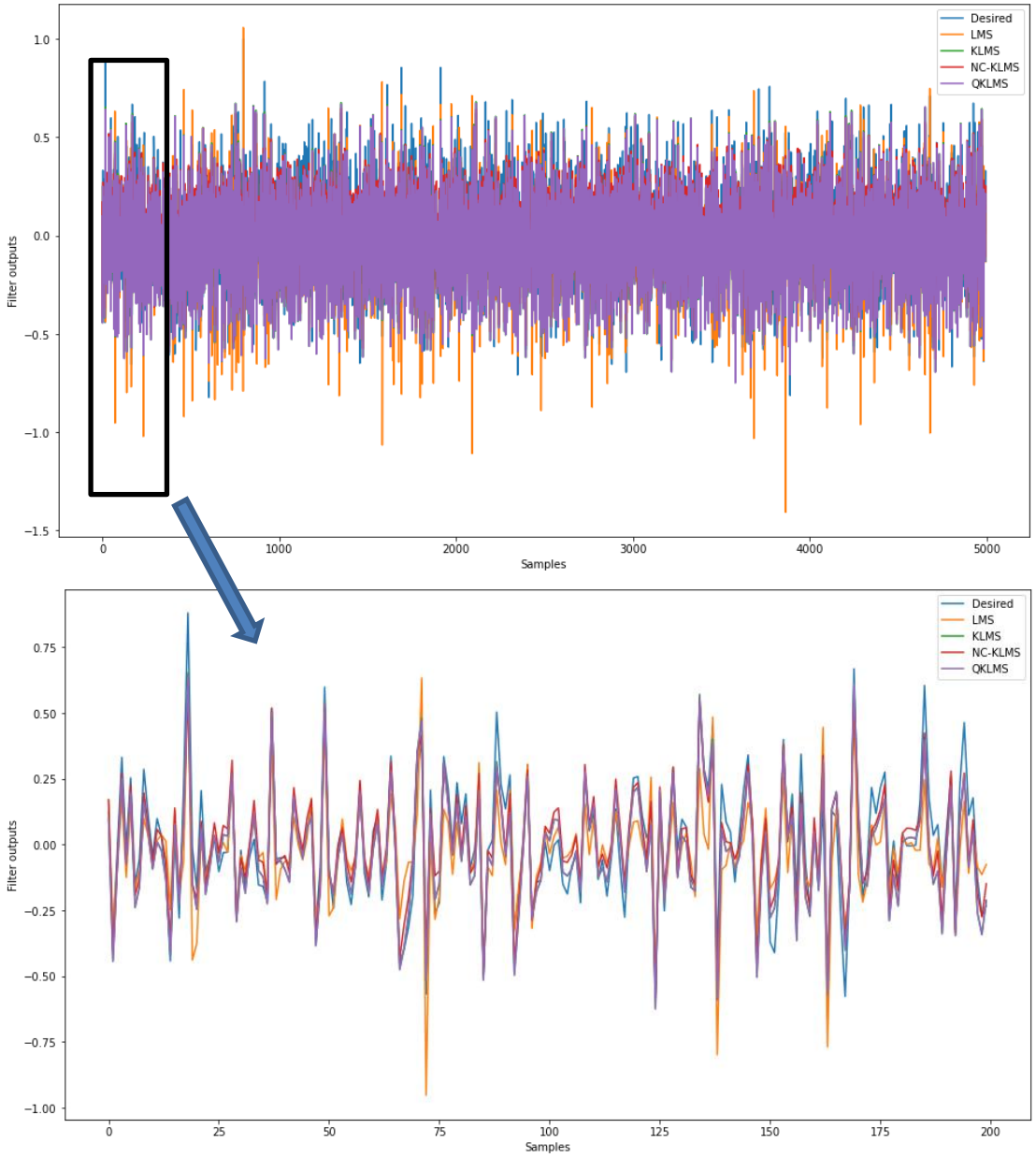


Fig. 4 Fitting curve of filter outputs and desired responses

2) Learning curve of training MSE by iterations

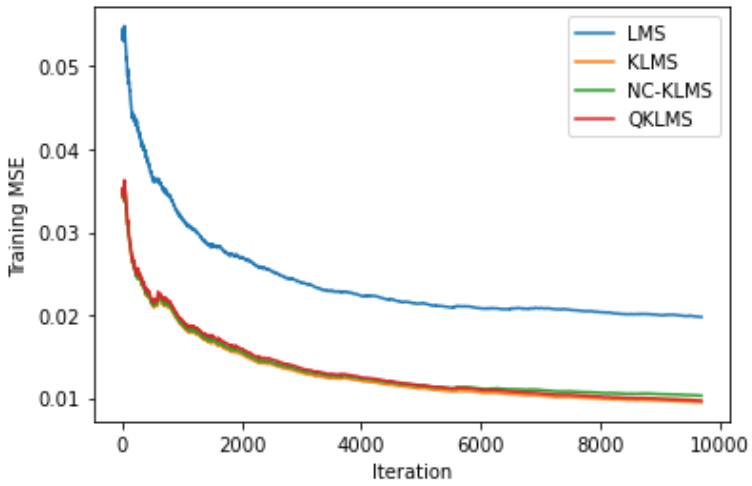


Fig. 5 Learning curve of LMS, KLMS, NC-KLMS, QKLMS algorithms

3) Growth curve of network size

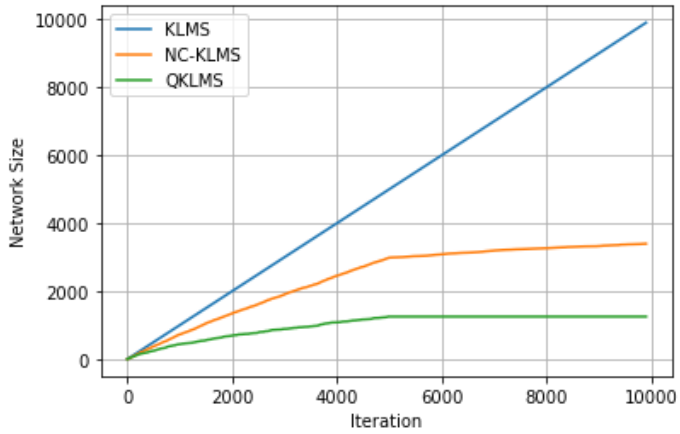


Fig. 6 Growth curve of network size of KLMS, NC-KLMS, QKLMS algorithms

iii. Comparison in the performances of LMS, KLMS, NC-KLMS, and QKLMS

Table 3 Summary of performances for the filters with different adaptive algorithms

Algorithm	Linear	Nonlinear (Gaussian kernel)		
	LMS	KLMS	NC-KLMS	QKLMS
Network size	N/A	9895	3395	1253
Training MSE for N=5,000	0.0198	0.0095	0.0103	0.0097
Testing MSE for the next n=100	0.0210	0.0094	0.0094	0.0095
Running time (s)	3.36	1,320.37	526.73	230.09

Major findings regarding performances of linear and nonlinear filters:

(1) **Performances of linear and nonlinear filters**. For the linear filter with LMS algorithm, the weight tracks indicate that the five weights corresponding to the order (5) of filter gradually converge to approximately [0.64, -1.08, -0.51, -0.20, -0.04], as shown in Fig. 3(1). This result suggests the convergence and stability of the LMS algorithm in addressing channel equalization problems. While in terms of fitting curve (filter outputs vs desired responses) and learning curve (training MSE vs iteration), the nonlinear filters for the kernel methods all outperform the linear filter with the LMS algorithm and have higher consistency smaller training MSE, as illustrated in Fig. 4 and 5. This also implies that kernel methods can well capture the nonlinearity between the filter inputs and desired response due to the presence of nonlinear channel while the LMS algorithm cannot work well. This is also why the kernel methods for KLMS, NC-KLMS and QKLMS algorithms are very helpful in addressing this nonlinear channel equalization task.

(2) **Compare the growth curve of kernel methods**. Although the kernel methods all demonstrate good performances in nonlinear channel equalization, there still exist some differences among various adaptive algorithms (KLMS, NC-KLMS and QKLMS). The biggest difference among the three algorithms lies in the network size of the nonlinear filters. As presented in Fig. 6 and Table 3, the QKLMS has the least network structure/size equaling to 1253, significantly smaller than the other two algorithms – NC-KLMS (3395) and KLMS (9895),

but they almost achieve the same performances in addressing nonlinear channel equalization problems. This shows a higher efficiency of the QKLMS algorithm than the NC-KLMS and KLMS algorithms, as well as faster convergence speed.

(3) **Compare the running time of the filters**. The superiority of the QKLMS in addressing nonlinear channel equalization problem can be further reflected by the running time of the nonlinear filters. The QKLMS algorithm only requires less than 4 min to train the adaptive filter, while the NC-KLMS and KLMS algorithms need 9 min and 22 min for the training process, respectively. This means that the QKLMS algorithm can significantly reduce the training time of the nonlinear filter, even 2 and 5 times less than the NC-KLMS and KLMS algorithms. Meanwhile, it can also be found that the sparsification method with novelty criterion can help improve the efficiency and decrease the network size of the KLMS algorithm. It is worth mentioning in Table 3 that the training and testing MSE of kernel methods in the ascending order of the KLMS, QKLMS, and NC-KLMS are both smaller than the linear filter with the LMS algorithm. This further suggests that the kernel methods have stronger generalization and better performances in addressing nonlinear channel equalization problems.

In summary, the kernel methods (KLMS, NC-KLMS and QKLMS) exhibit obvious superiority, which is of no surprise as LMS is incapable of handling non-linearities. In addition, the quantization technique, also called QKLMS, achieves better economy and remarkable efficiency than the novelty criterion (NC-KLMS) or without any techniques (KLMS).