Homework 4: Frequency Doubling

Kaifa Lu

Solution:

Step 1: **Generate the input *x* and desired response *d* of an adaptive filter**. Simulate a single sinewave at 1KHz sampled at 10KHz and the desired response at 2KHz to generate 2 seconds of data pairs using the following equations implemented in Python:

1) Input x of an adaptive filter:

$$x = sin(2\pi f_x t), t = 0{:}\frac{1}{f_s}{:}2$$

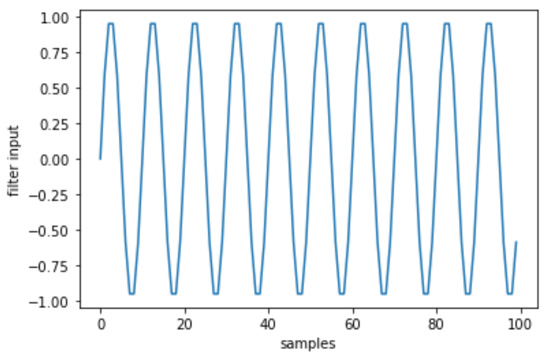
2) Desired response d of an adaptive filter:

$$d = sin(2\pi f_d t), t = 0{:}\frac{1}{f_s}{:}2$$

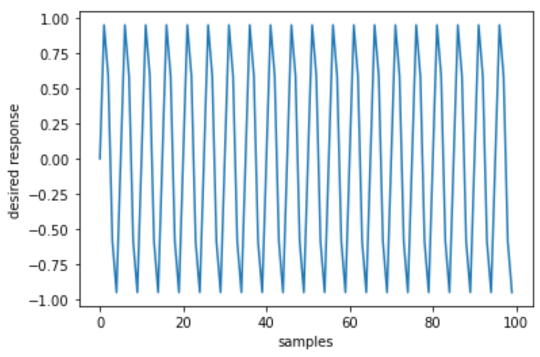
3) Noisy desired response d' of an adaptive filter:

$$d' = d + u, u \sim p(u) = 0.9G(0,0.1) + 0.1G(4,0.1)$$

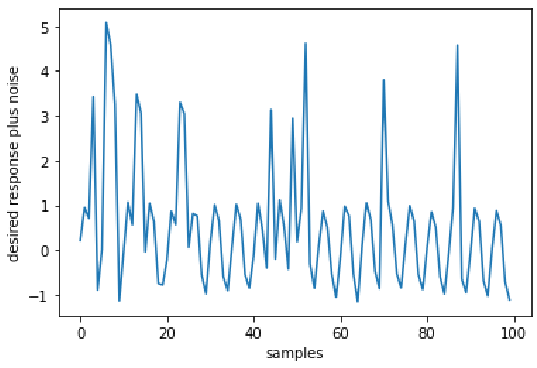
Here, we can obtain the input x, desired response d and noisy desired response d' of an adaptive filter with 2 seconds of data pairs to train the adaptive filter for frequency doubling, as shown in Fig. 1.



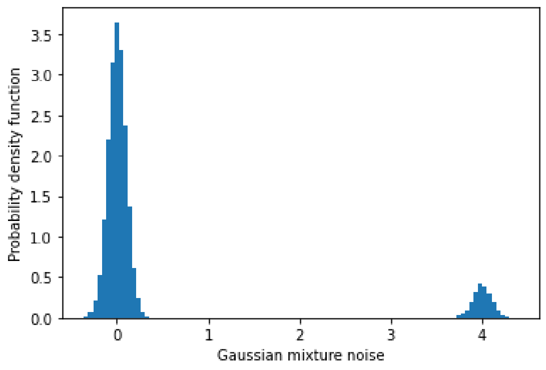
(1) The first 100 samples of filter input



(2) The first 100 desired responses of filter



(3) The first 100 noisy desired responses of filter



(4) PDF of Gaussian mixture noise

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

Step 2: **Select four adaptive algorithms**. Here, we separately select Wiener solution, least mean square (LMS), kernel LMS (KLMS), and kernel minimum error entropy (KMEE) as four algorithms of the adaptive filter, and the pseudocodes of their implementation process are presented in Table 1.

Table 1 Pseudocodes of implementing the Wiener solution, LMS, KLMS and KMEE

Algorithm 1 Wiener solution

```
Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), d_n)$ where $L = 10$, and the number N=20,000
Output: the weights of filters $\boldsymbol{w} = [w_1, w_2, \cdots, w_9, w_{10}]^T$
While $\{i = 0, \cdots, L-1\}$ is available do
    $P_i = E[\boldsymbol{d}(n)\boldsymbol{X}(n-i)]$, $\boldsymbol{d}(n) = [d_L, \cdots, d_n], \boldsymbol{X}(n-i) = [\boldsymbol{x}_{L-i}, \cdots, \boldsymbol{x}_{n-i}]$
    $R_{i,k} = E[\boldsymbol{X}(n-i)\boldsymbol{X}(n-k)], k = 0, \cdots, L-1$
end while
$\boldsymbol{w}_{opt} = \boldsymbol{R}^{-1}\boldsymbol{P}$
```

Algorithm 2 LMS Algorithm

```
Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), d_n)$ where $L = 10$, and the number N=20,000
Output: the weights of filters $\boldsymbol{w} = [w_1, w_2, \cdots, w_9, w_{10}]^T$
Initialization: step size $\eta = 0.1$, $\boldsymbol{w}_0 = [w_1, w_2, \cdots, w_9, w_{10}]^T \sim N(0,0.1)$
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 3 KLMS Algorithm

```
Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), d_n)$ where $L = 10$, and the number N=20,000
```

```
Output: the expansion $y = \sum_{k=1}^{K} \alpha_k \kappa(\cdot, \boldsymbol{x}_k)$ where $\alpha_k = \eta e_k$
Initialization: step size $\eta = 0.01$, 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
    $e_i = d_i - \sum_{j=1}^{i-1} \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}_j\})$
    $\boldsymbol{C}_i = \{\boldsymbol{C}_{i-1}, \boldsymbol{x}_i\}$, $\boldsymbol{\alpha}_i = [\boldsymbol{\alpha}_{i-1}, \eta e_i]$
end while
```

Algorithm 4 KMEE Algorithm

```
Input: $((x_n, x_{n-1}, \cdots, x_{n-L+1},), d_n)$ where $L = 10$, and the number N=20,000
Output: the expansion $y = \sum_{k=1}^{K} \sum_{j=1}^{M} \alpha_{k,j} \left(\kappa_d(\cdot, \boldsymbol{x}_{k-j}) - \kappa_d(\cdot, \boldsymbol{x}_k)\right)$
Initialization: step size $\eta = 0.01$, Gaussian kernel size $\sigma_e = 1$, $\sigma_d = 0.4$, batch size $M$ = 200, center dictionary $\boldsymbol{C} = \boldsymbol{C}_1\{\boldsymbol{x}_1\}$ and coefficient vector $\boldsymbol{\alpha}(1) = [\eta d_1]$
While $\{(\boldsymbol{x}_i, d_i), i = 2, \cdots, N\}$ is available do
    $e_i = d_i - \sum_{k=1}^{i-1} \sum_{l=1}^{M} \alpha_{k,l} \left(\kappa_d(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}_{k-j}\}) - \kappa_d(\boldsymbol{x}_i, \boldsymbol{C}_{i-1}\{\boldsymbol{x}_k\})\right)$
    Update the coefficient vector $\boldsymbol{\alpha}_i$ as follows:
    $$\boldsymbol{\alpha}_{i,j} = \begin{cases} \frac{\eta}{M\sigma_e^2} \kappa_e(e_i - e_j)(e_i - e_j), i \le M, 1 \le j < i \\ \frac{\eta}{M\sigma_e^2} \kappa_e(e_i - e_j)(e_i - e_j), i > M, i - M \le j < i \end{cases}$$
    $\boldsymbol{C}_i = \{\boldsymbol{C}_{i-1}, \boldsymbol{x}_i\}$, $\boldsymbol{\alpha}(i) = [\boldsymbol{\alpha}(i-1), \boldsymbol{\alpha}_i]$
end while
```

Step 3 (Q1): **Compare the performances of Wiener solution and LMS in frequency doubling without Gaussian mixture noise**. The performances can be reflected by the fitting curve (filter output vs desired response) for the linear adaptive filter, and weight tracks and learning curve (training MSE vs iteration) for the LMS algorithm as follows:

i. <u>Linear Filters for Wiener Solution</u>

The optimal weights of the adaptive filter with Wiener solution are:

$$\boldsymbol{w} = [2.74, 39.33, 79.25, 102.47, 10.18, -0.92, 18.97, 103.39, 100.02, -4.55]^T$$

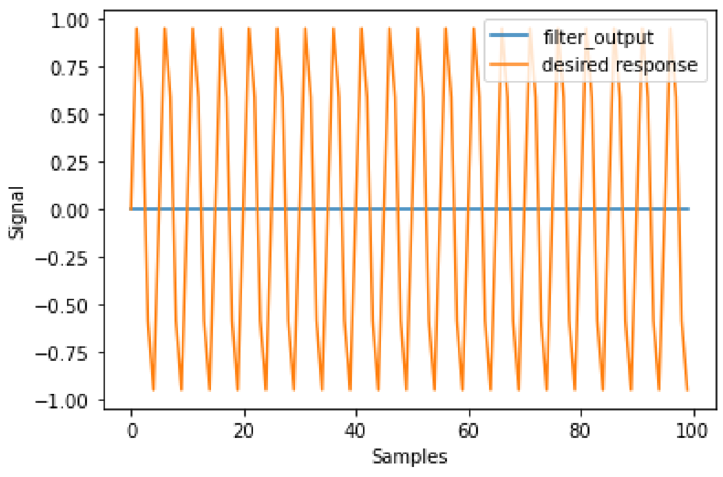
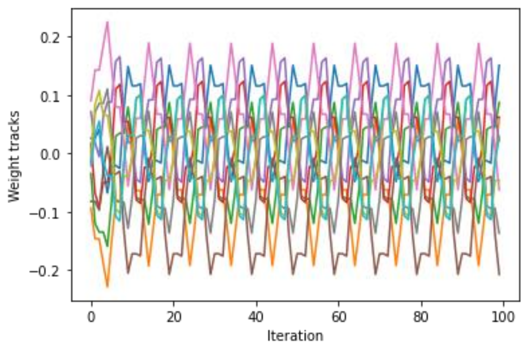


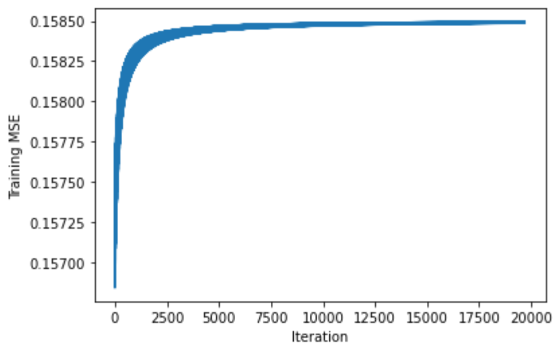
Fig. 2 Fitting curve of desired response and filter output by Wiener solution

As shown in Fig. 2, we can easily find that the Wiener solution fails and exhibits significant discrepancy between filter outputs and desired responses. The mean square error of the Wiener solution is equal to 0.50. The main reason lies in that (1) Wiener solution is used to train the linear filters and cannot well learn the nonlinearity, while doubling frequency of a sinewave is a nonlinear process. (2) Although a total of 20,000 data samples are used in Wiener solution, there is only about 10 effective samples used for the calculation of the optimal weights. This is mainly because the data samples are generated in a periodic manner due to the relationship between the sampling frequency (10KHz) and signal frequency (1KHz). Hence, the Wiener filter cannot capture the entire features of the sinewave, particularly for those not covered and revealed by the existing samples.

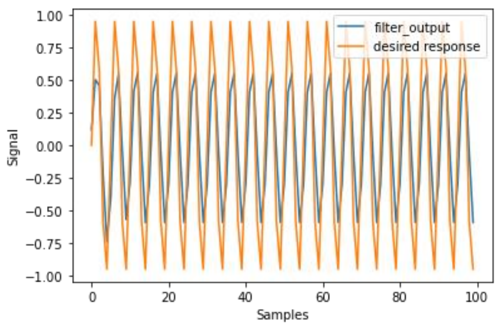
ii. <u>Linear Filters for LMS Algorithm</u>



(1) Weight tracks of the first 100 iterations



(2) Learning curve



(3) Fitting curve of filter outputs and desired responses over the first 100 samples

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

As shown in Fig. 3, the weight tracks of the adaptive filter with the LMS algorithm show periodicity, indicating that simply adding the number of samples cannot further contribute to the weight updates. The learning curve suggests that the training MSE gradually increases by iteration and is eventually maintained at about 0.1585. Also, the mean square error of the adaptive filter with LMS over all samples is equal to 0.1585. These results suggest that the LMS algorithm performs badly in doubling frequency, which can be explained by almost the same reasons as the Wiener solution that (1) the LMS algorithm is used to train the linear filter and cannot well learn the nonlinearity, while doubling frequency of a sinewave is a nonlinear process. (2) Only about 10 samples are effective in calculating the optimal weights of the adaptive filter. Similarly, the adaptive filter with LMS cannot still capture the entire features of the sinewave, particularly for those not covered and revealed by the existing samples.

Step 4: **Determine the optimal hyperparameters of kernel methods for nonlinear adaptive filter and explore the effects of kernel size and step size on filter performances in frequency doubling without Gaussian mixture noise**. The two most important parameters for kernel methods are the kernel size and step size that significantly affect the filter performances. Here, we vary the kernel size of the kernel methods from 0.1, 0.4, 0.8, 1, 5, 10 to 20 and the step size from 0.001, 0.01, 0.05, 0.1, 1, 5 to 10 to find a better kernel size and step size for the kernel methods (e.g., KLMS).

1) Effect of kernel size on filter performances

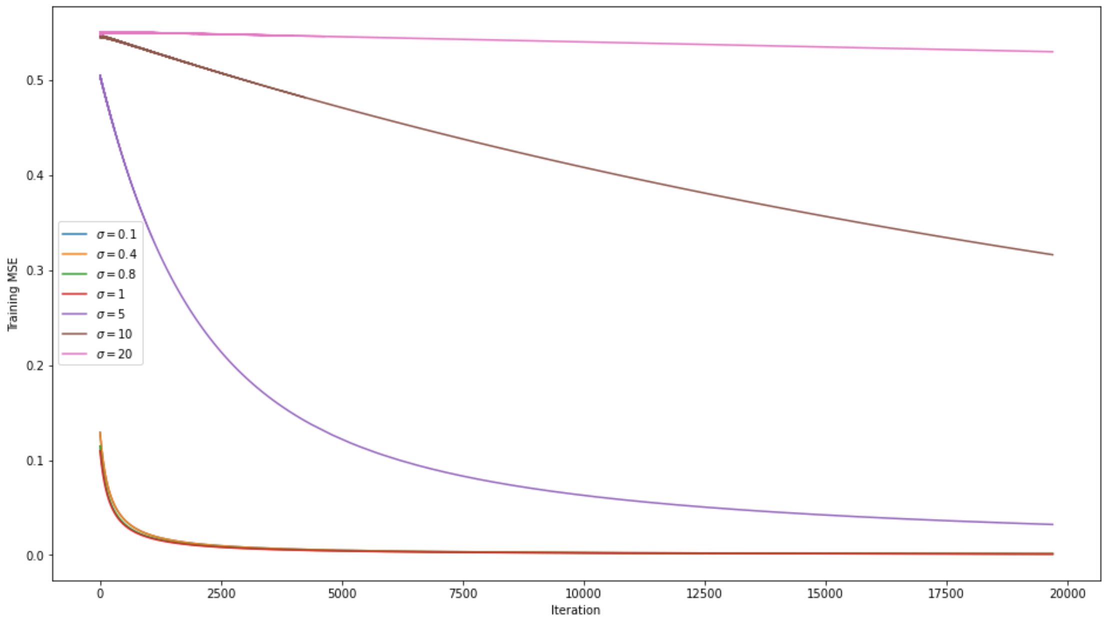


Fig. 4 The filter performances of the kernel methods with different kernel sizes

As illustrated in Fig. 4, when the kernel size of the kernel methods is less than 1, the nonlinear filter achieves better performances in frequency doubling, with lower training MSE between the filter outputs and desired responses, than the cases where the kernel size is higher than 1. Here, we set the best kernel size for the nonlinear methods as **0.4**, just based on the above experimental results.

2) <u>Effect of step size on filter performances</u>

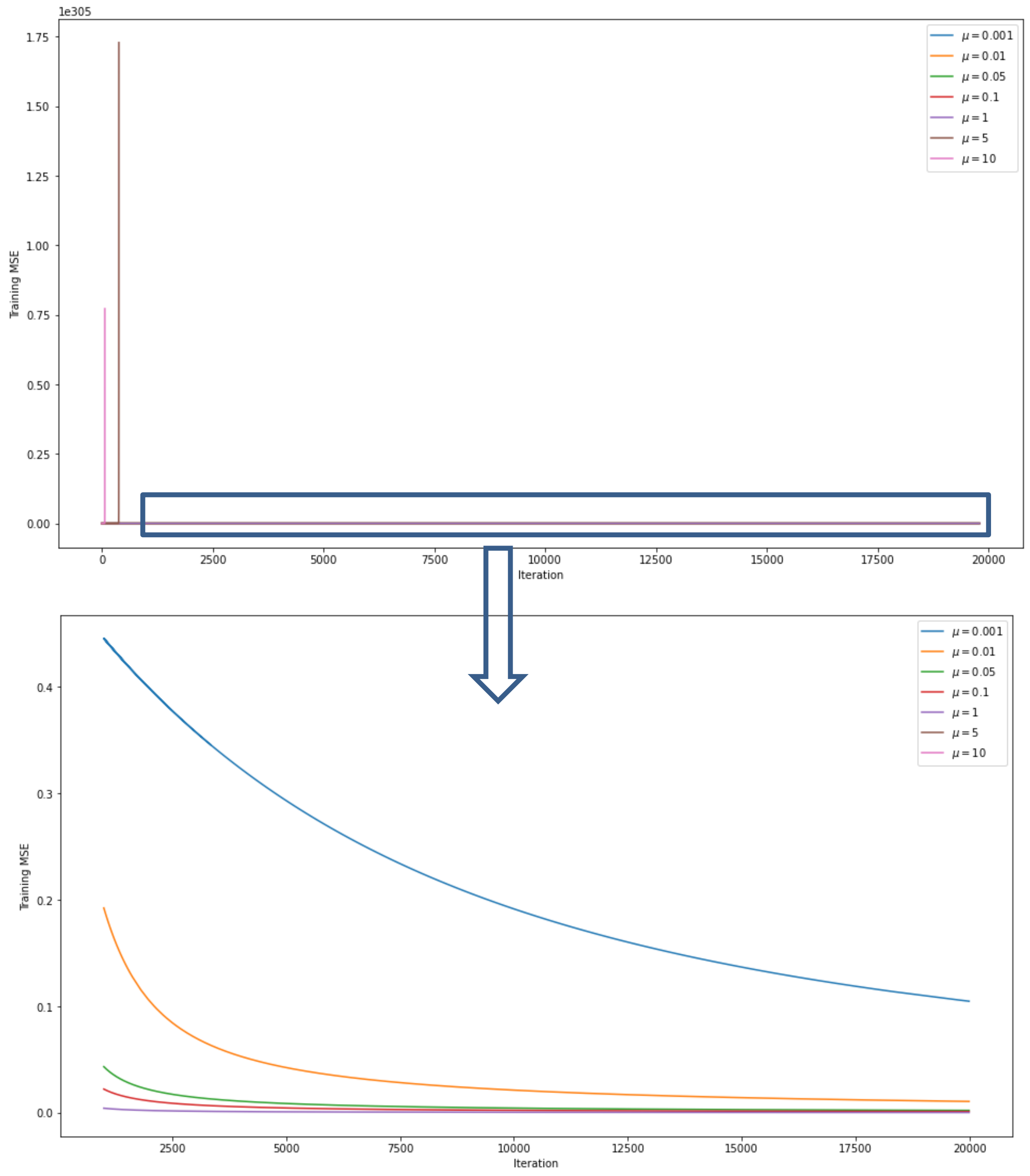


Fig. 5 The performances of the kernel methods with different step sizes

As shown in Fig. 5, when the step size of the kernel methods is less than 1, the nonlinear filter achieves better performances in frequency doubling, with lower training MSE and faster convergence speed, than the cases where the step size is higher than 1. However, when the step size is less than 0.01, the kernel methods converge slowly. Therefore, for convenience, we set the best step size for the kernel methods as **0.01**, just based on the above experimental results.

Step 5 (Q2): **Explore the performances of the nonlinear adaptive filter with KLMS in frequency doubling without Gaussian mixture noise**. The performances can be reflected by fitting curve (filter output vs desired response) and learning curve (training MSE vs iteration) as follows:

1) Fitting curve of filter outputs and desired responses

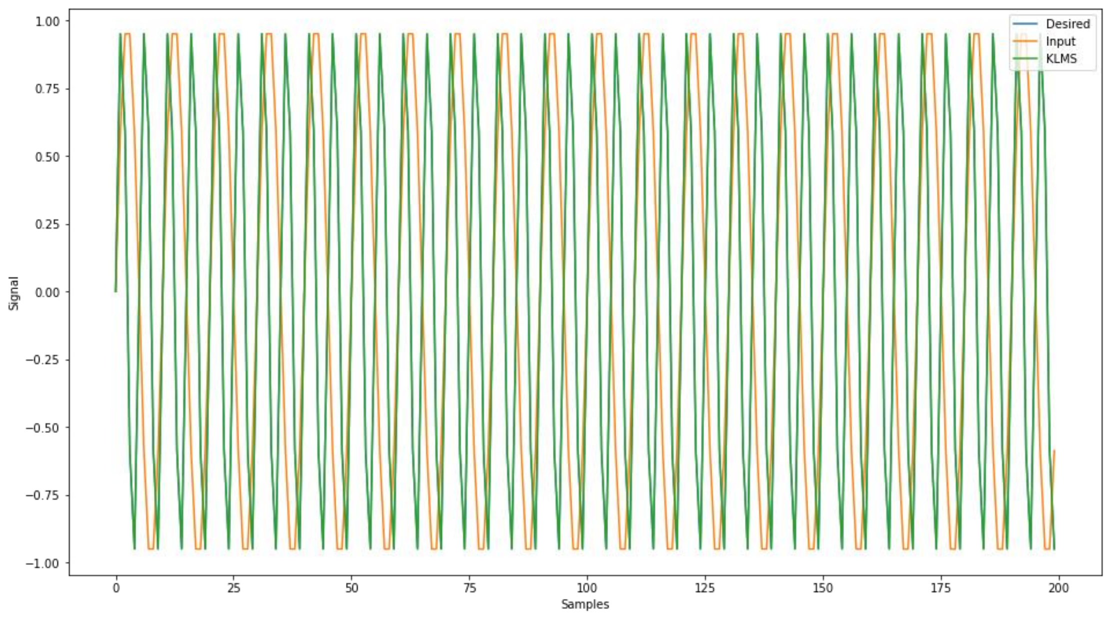


Fig. 6 Fitting curve of the first 200 filter outputs and desired responses

Fig. 6 shows that the outputs of the adaptive filter with KLMS demonstrate a high consistency with the desired responses, which both doubles the frequency of the signal inputs (sinewave). This suggests that the solution is reasonable and KLMS algorithm can efficiently address nonlinear frequency doubling problems.

2) Learning curve of training MSE by iterations

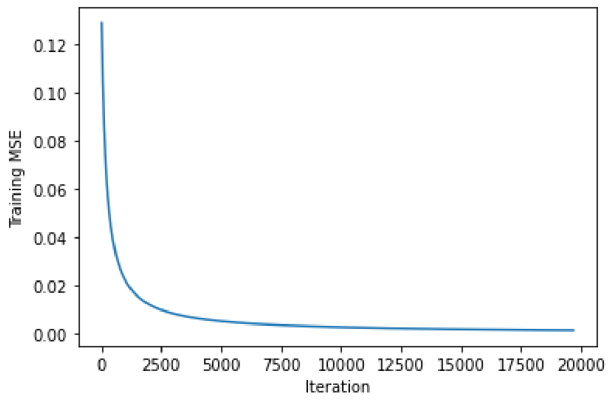
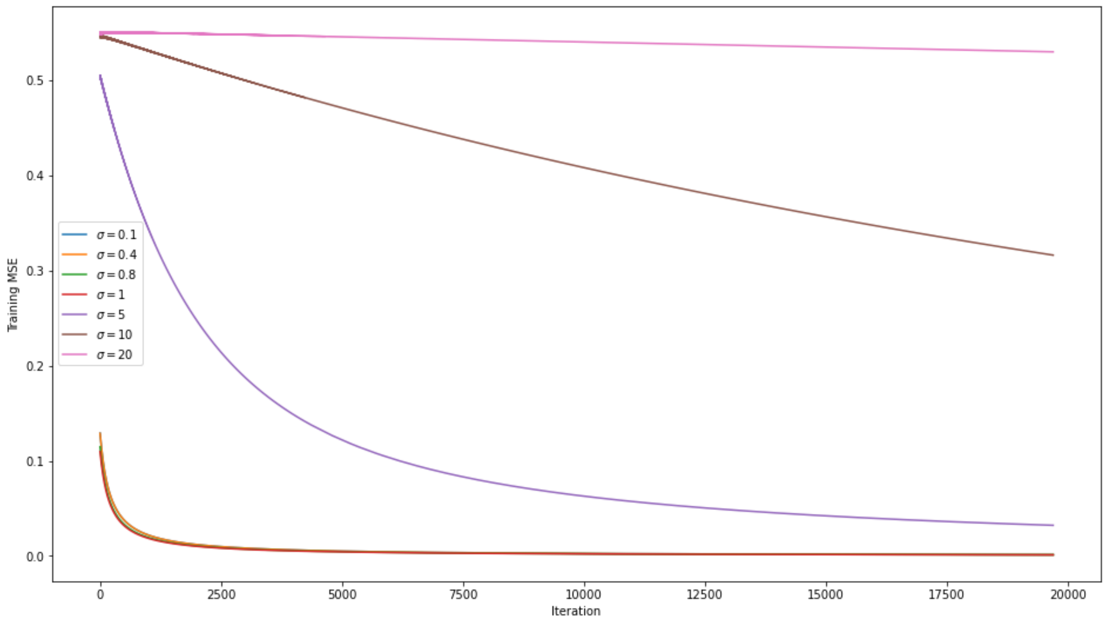


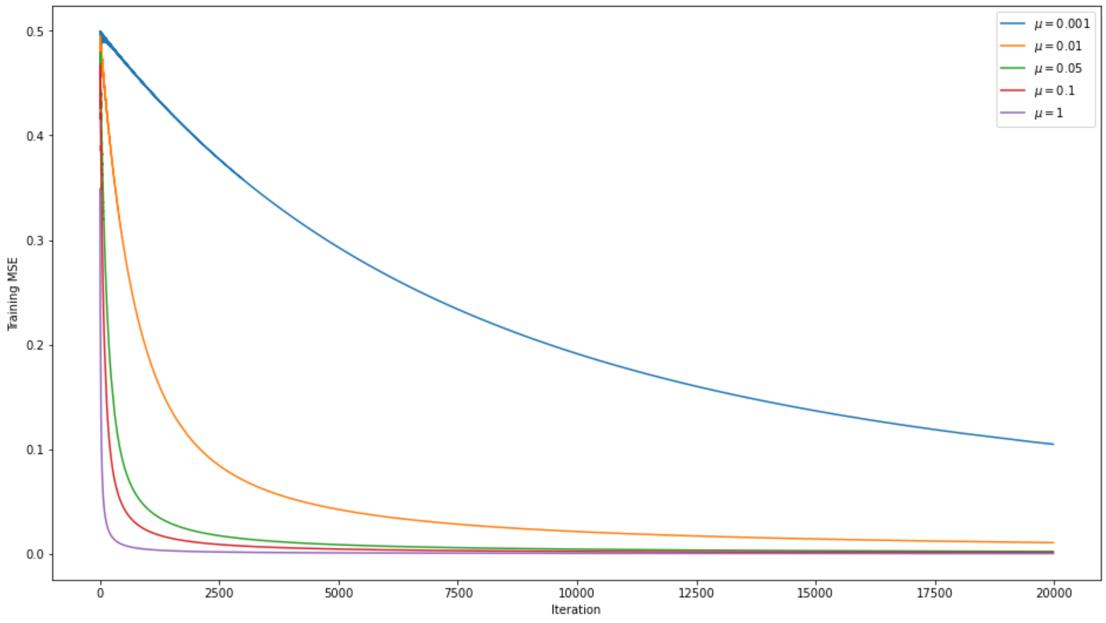
Fig. 7 Learning curve of the nonlinear adaptive filter with KLMS

As shown in Fig. 7, the learning curve of the nonlinear adaptive filter with KLMS suggests that the training MSE exhibits a gradual decay by iterations until convergence. This also indicates the stability and efficiency of the KLMS algorithm in addressing frequency doubling problems and that the solution is reasonable.

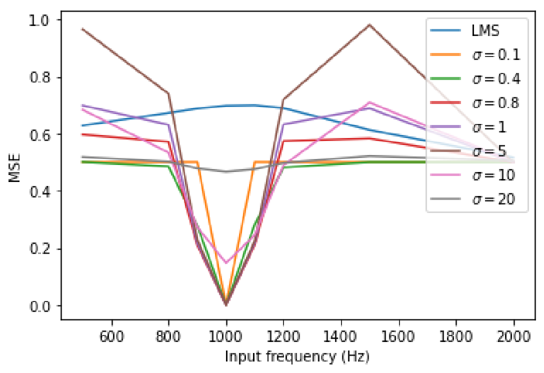
3) Generalization of the trained model with different input frequencies and effects of kernel size and step size

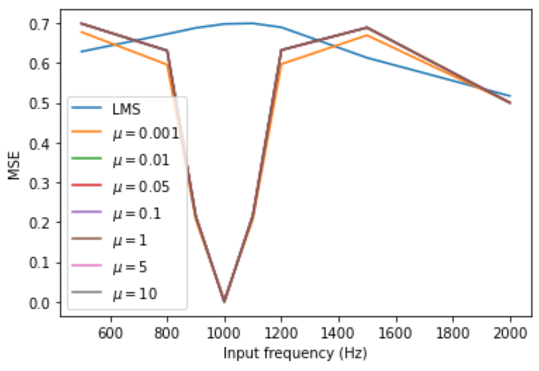


(1) Training MSE by iterations with different kernel sizes

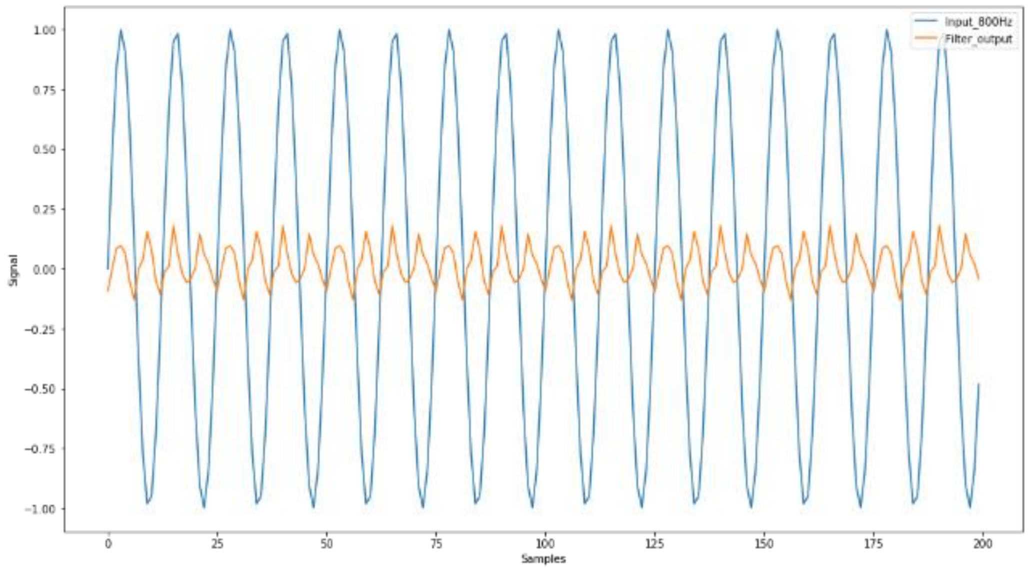


(2) Training MSE by iterations with different step sizes

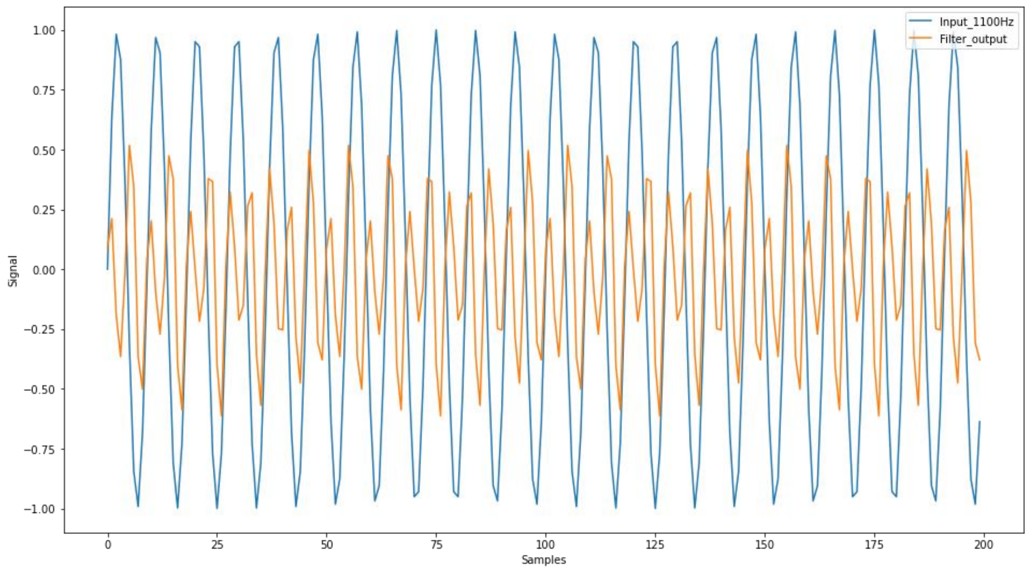




(3) Effect of kernel size on generalization (4) Effect of step size on generalization



(5) Curve of the first 200 filter outputs with an input frequency of 800Hz, $\sigma = 0.4$



(6) Curve of the first 200 filter outputs with an input frequency at 1100Hz, $\sigma = 0.4$

Fig. 8 Generalization of the trained frequency doubling model by KLMS

Generalization: When varying the input frequency from 500, 800, 900, 1000, 1100, 1200, 1500 to 2000Hz, we present the performances of the trained model in doubling the input frequency of the sinewave in Fig. 8. Only when the input frequency is closer to 1000Hz can the nonlinear adaptive filter show relatively better performances than the other cases where the input frequency is further away from 1000Hz. This suggests that the trained adaptive filter with KLMS has a weaker generalization performance, which is sensitive to the input frequency of the sinewave.

Effect of kernel size: When the kernel size is less than 5, the KLMS algorithms all exhibit better performances than the cases where the kernel size is higher than 5, in doubling the input frequency at 1000Hz. The generalization performances, regardless of the kernel size, become worse as the input frequency is further away from 1000Hz. Furthermore, the KLMS algorithms with the kernel size ranging from 0.1 to 5 have weaker generalization performances. When the kernel size is equal to 0.4, the adaptive filter with KLMS exhibits the best performance in frequency doubling, with the lowest MSE. By contrast, when the kernel size is higher than 5 (e.g., 10, 20), the performances of the nonlinear filter with KLMS almost remain unchanged despite a change of the input frequency from 500 to 2000Hz and the MSE is approximately equal to 0.5.

Effect of step size: When the step size is no more than 1, the step size of the KLMS algorithm hardly affects the generalization performances of the nonlinear adaptive filter. This means that a change in the step size cannot greatly improve the generalization of the nonlinear adaptive filter in doubling other frequencies of the sinewave. However, when the step size is higher than 1, the nonlinear adaptive filter with KLMS performs badly.

Step 6 (Q3): **Compare the performances of the KLMS and KMEE algorithms in frequency doubling with Gaussian mixture noise**. The performances of the nonlinear adaptive filter can be reflected by the fitting curve (filter output vs desired response) and learning curve (training MSE vs iteration) as follows:

i. Nonlinear Filters for KLMS Algorithm

1) Fitting curve of filter outputs and noisy desired responses

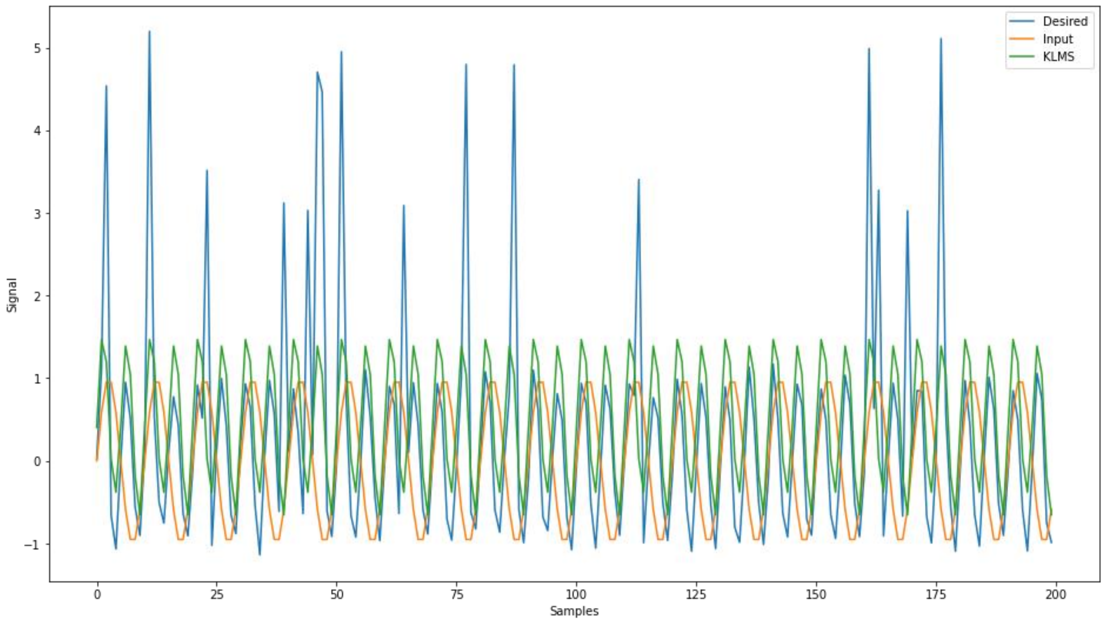


Fig. 9 Fitting curve of the first 200 filter outputs and noisy desired responses

As shown in Fig. 9, although the Gaussian mixture noise slightly deviates the filter outputs upward from the ideal desired responses, the filter outputs basically double the input frequency of the sinewave. This suggests that the adaptive filter with KLMS can basically filter out the adverse effects of the Gaussian mixture noise and is still effective in addressing frequency doubling problems to some extent.

2) Learning curve of training MSE by iterations

Although the Gaussian mixture noise can partly intervene the training process of the nonlinear adaptive filter, the training MSE still shows a notable decay by iterations until convergence. It is also easily understood that the training MSE is higher than that of the case without Gaussian mixture noise.

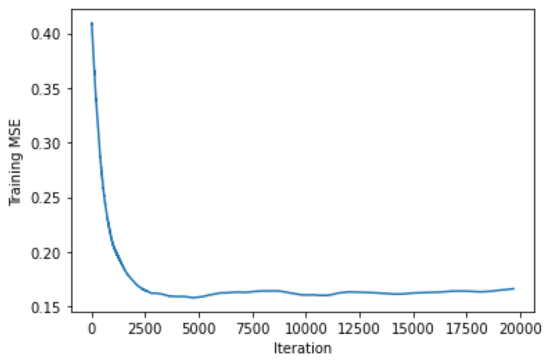
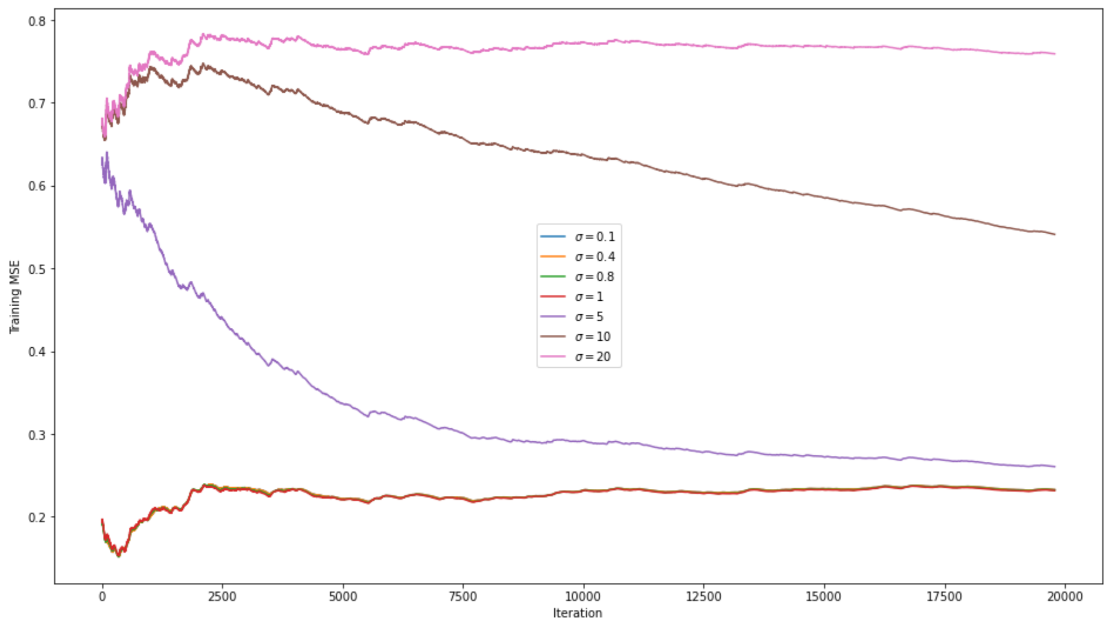
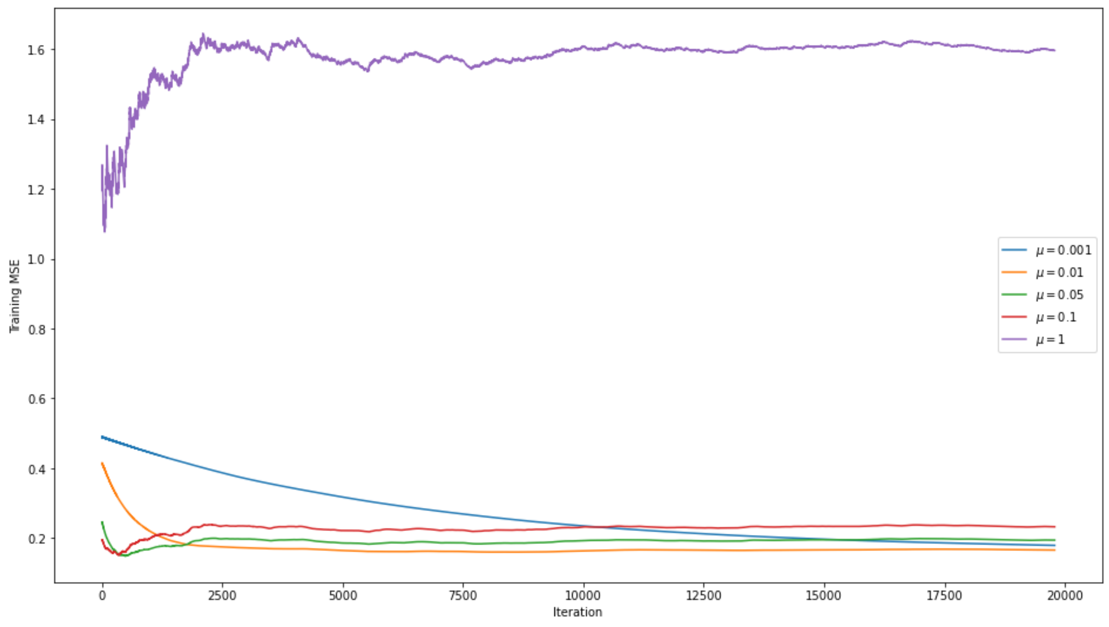


Fig. 10 Learning curve of the nonlinear adaptive filter with KLMS

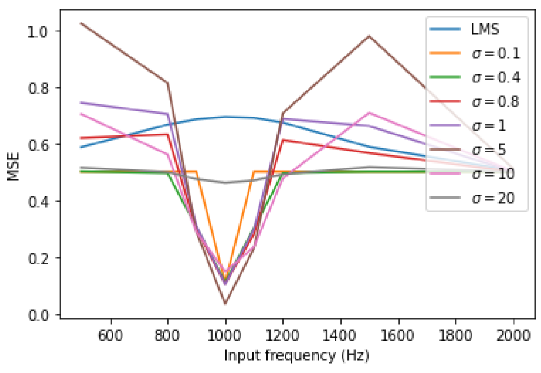
3) Generalization of the trained model by KLMS with different input frequencies and effects of kernel size and step size

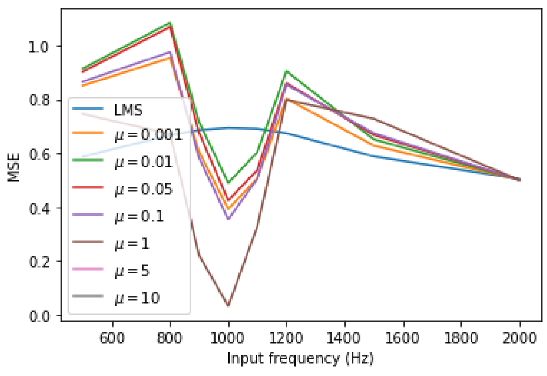


(1) Training MSE by iterations with different kernel sizes

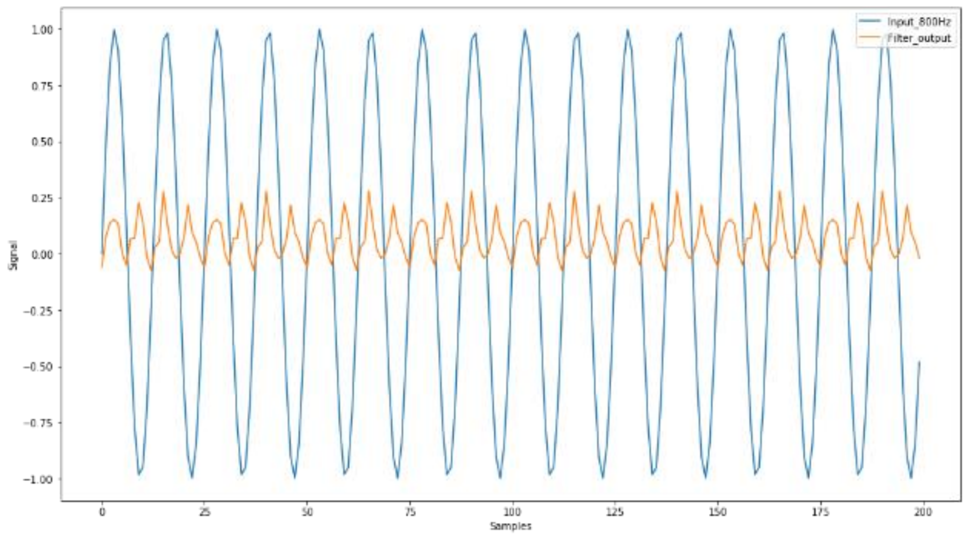


(2) Training MSE by iterations with different step sizes

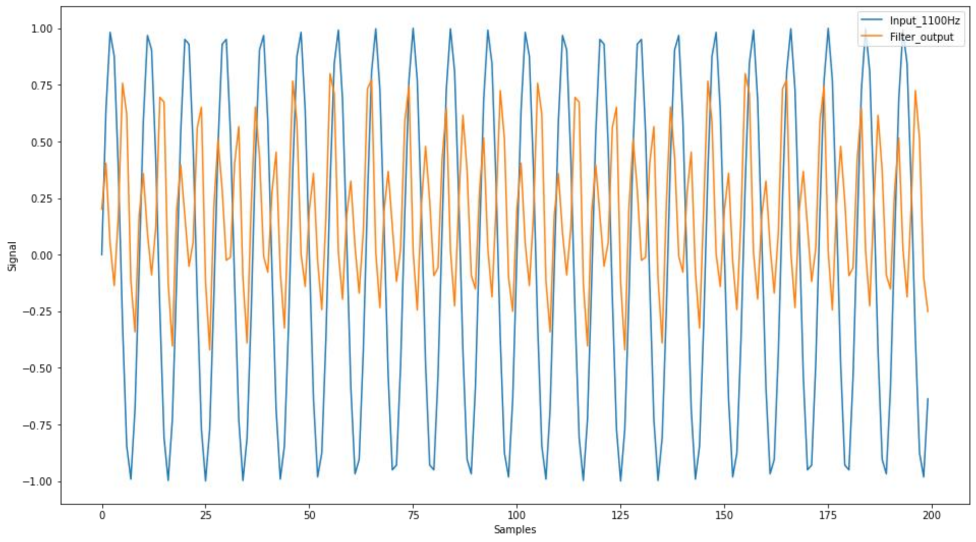




(3) Effect of kernel size on generalization (4) Effect of step size on generalization



(5) Curve of the first 200 filter outputs with an input frequency at 800Hz, $\sigma = 0.4$



(6) Curve of the first 200 filter outputs with an input frequency at 1100Hz, $\sigma = 0.4$

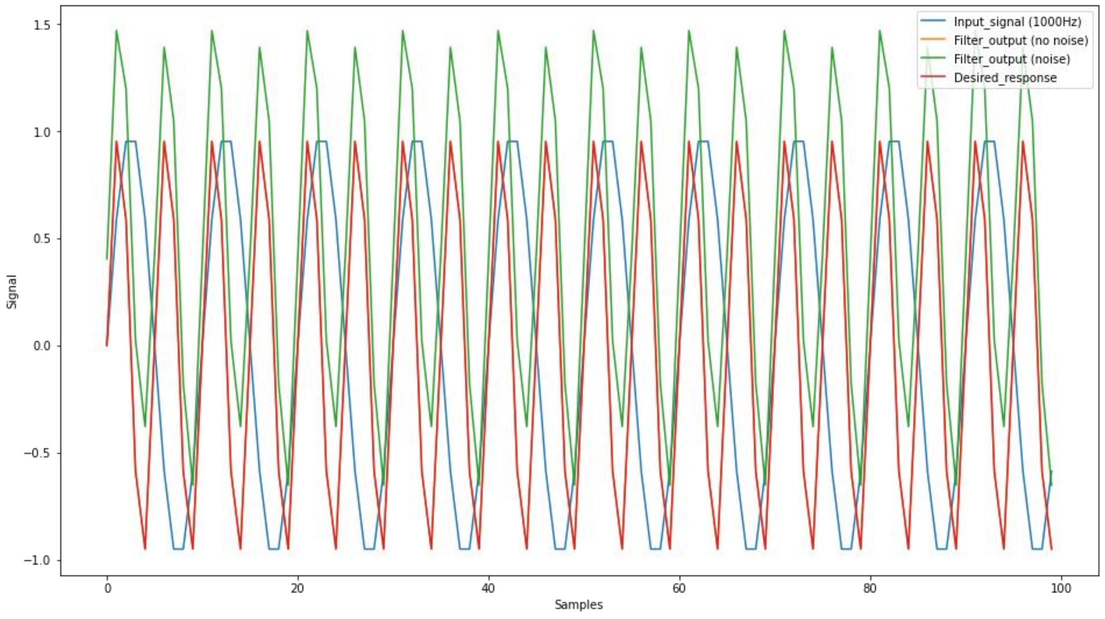
Fig. 11 Generalization of the trained frequency doubling model by KLMS

Generalization: Like the case without Gaussian mixture noise, the trained model with the Gaussian mixture noise does not exhibit a good generalization performance by different input frequencies, either. Only when the input frequency is closer to 1000Hz, the nonlinear adaptive filter with KLMS achieves a good performance in doubling the input frequency of the sinewave.

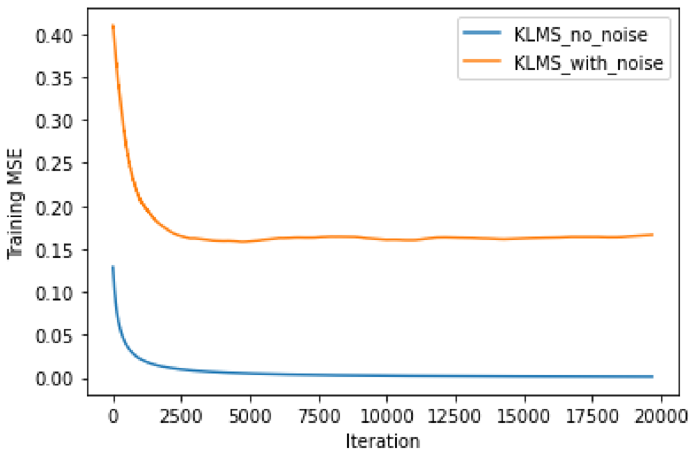
Effect of kernel size: The effect of kernel size on filter performances is almost the same as the case without the Gaussian mixture noise. It is worth noting that when the kernel size is equal to 5, the nonlinear adaptive filter with KLMS achieves the best performance in doubling the input frequency at 1000Hz and the worst generalization performance in doubling other input frequencies, compared with other kernel sizes.

Effect of step size: When the step size is less than 1, the step size of the KLMS algorithm hardly affects the generalization performances of the nonlinear adaptive filter. This means that a change in the step size cannot greatly improve the generalization of the nonlinear adaptive filter in doubling other input frequencies of the sinewave. When the step size is equal to 1, the adaptive filter with KLMS achieves the lowest MSE and the best generalization performance in doubling the input frequencies, compared with other step sizes. However, when the step size is higher than 1, the nonlinear adaptive filter with KLMS performs badly.

4) Effects of the noise in the trained model by KLMS



(1) Fitting curve of filter outputs (with and without noise) and ideal desired responses



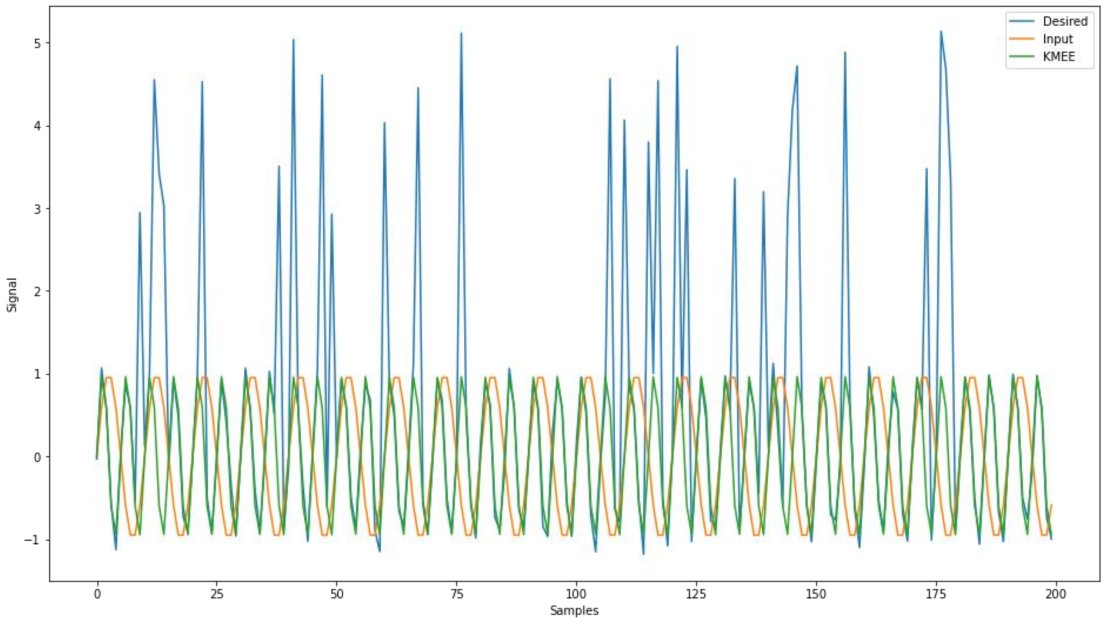
(2) Learning curve (with and without noise) of training MSE by iterations

Fig. 12 Effects of the noise in the trained model by KLMS

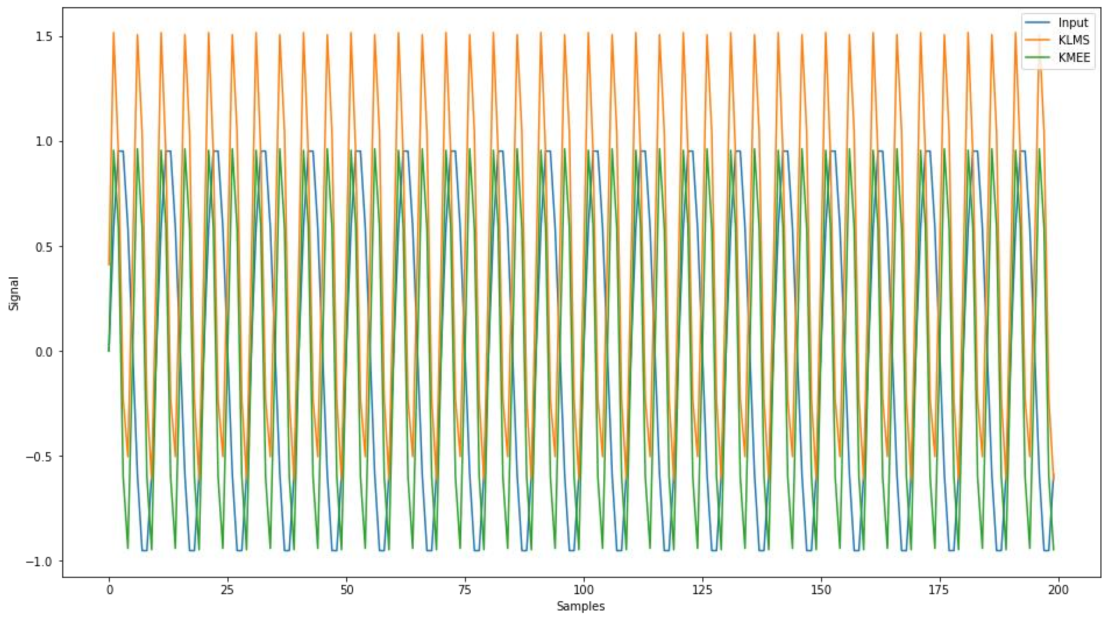
As illustrated in Fig. 12, the Gaussian mixture noise weakens the performances of the nonlinear adaptive filter with KLMS in doubling the input frequency of sinewave at 1000Hz. This can be reflected by both the learning curve and fitting curve of filter outputs and ideal desired responses, with a higher training MSE and inconsistency. The presence of the Gaussian mixture noise leads to an increase in the training MSE of the nonlinear adaptive filter with KLMS, but hardly changes the convergence of the KLMS algorithm. In addition, the Gaussian mixture noise makes the filter outputs higher than ideal desired responses. Totally, the two adaptive filters with KLMS (with and without noise) both achieve good performances in doubling the input frequency of the sinewave.

ii. Nonlinear Filters for KMEE Algorithm

1) Fitting curve of filter outputs and desired responses



(1) Fitting curve of the first 200 filter outputs and noisy desired responses by KMEE



(2) Curves of the first 200 sinewave inputs and filters outputs by KLMS and KMEE

Fig. 13 Fitting curve of filter outputs and desired responses by KLMS and KMEE

As illustrated in Fig. 13, the fitting curve shows that the filter outputs by KMEE are well in line with the ideal desired responses and are hardly affected by the Gaussian mixture noise. It can also be easily found that the nonlinear adaptive filter with KMEE achieves better performances in doubling the input frequency of the sinewave than that with KLMS, with a lower deviation from the ideal desired responses. This suggests that the KMEE algorithm performs better than the KLMS algorithm in filtering out the effects of the Gaussian mixture noise.

2) <u>Learning curve of training MSE by iterations</u>

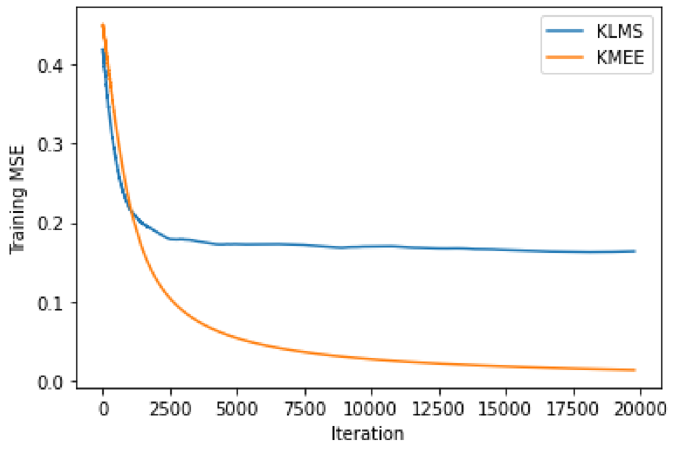


Fig. 14 Learning curve of the nonlinear adaptive filter by KLMS and KMEE

Fig. 14 shows that the training MSE of nonlinear adaptive filter with KLMS and KMEE algorithms separately converges to 0.164 and 0.014 but with almost equivalent convergence speed. This suggests that the KMEE algorithm outperforms the KLMS in minimizing the errors between the filter outputs and ideal desired responses, and that the KMEE algorithm can efficiently filter out the Gaussian mixture noise.

3) <u>Distributions of error between the desired responses and filter outputs</u>

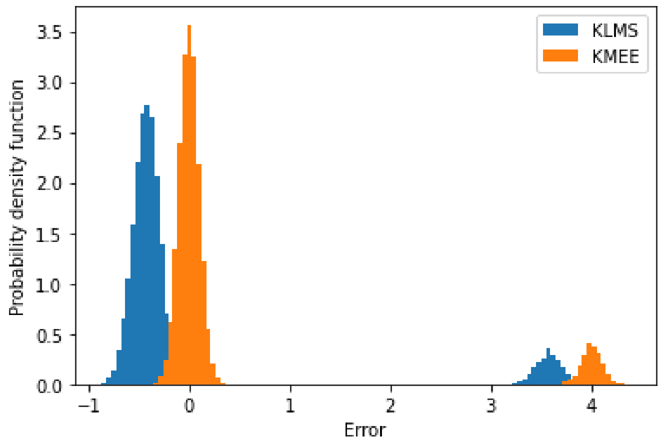
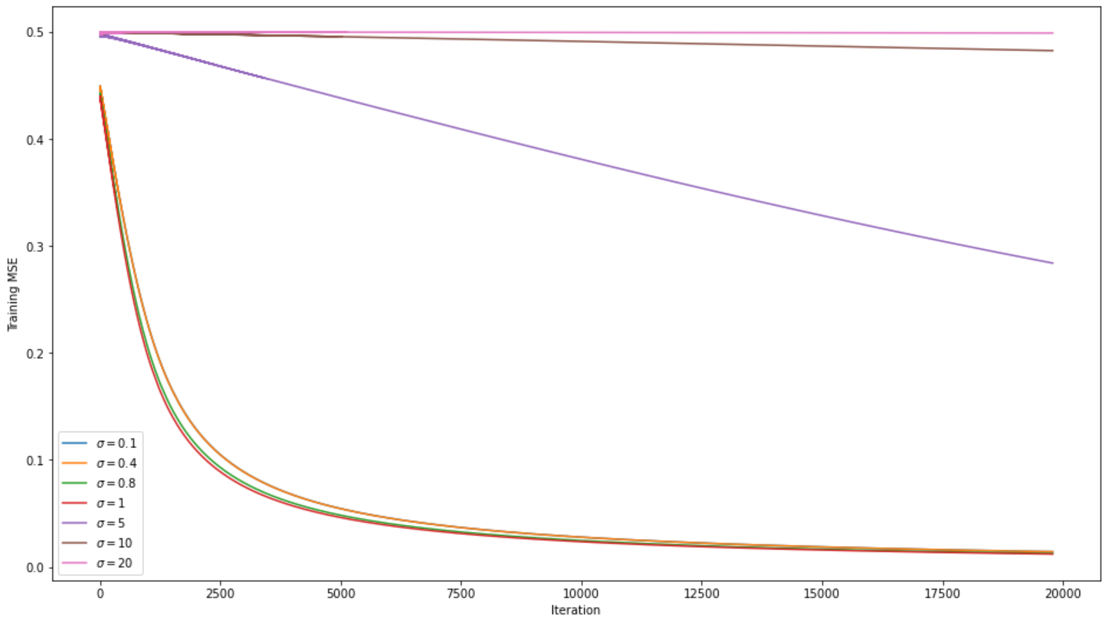


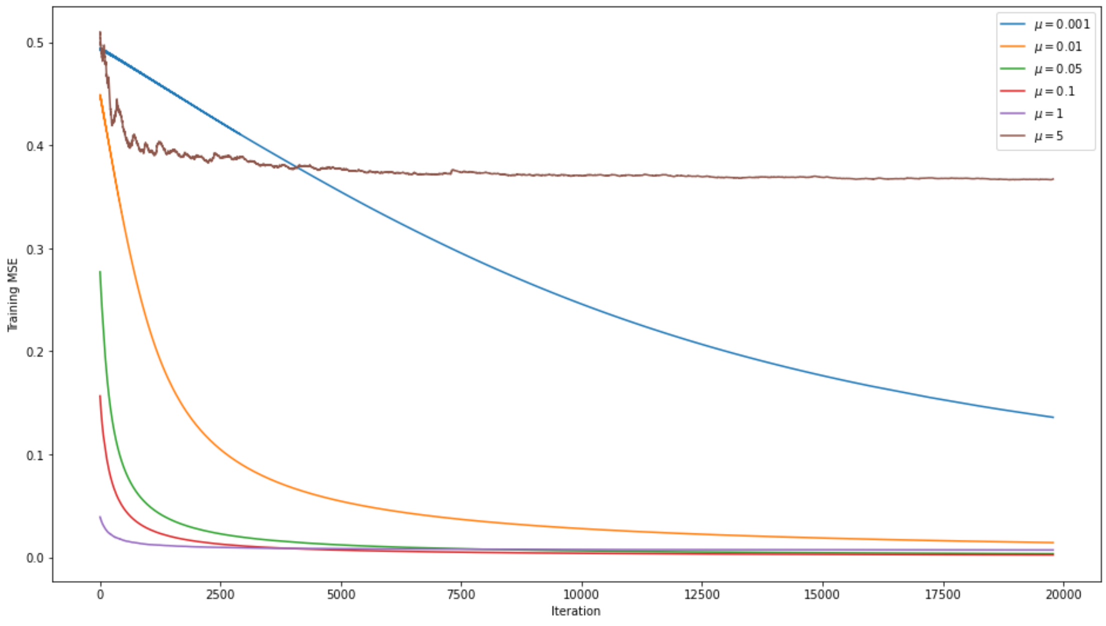
Fig. 15 Probability density function of errors between desired responses and filter outputs by KLMS and KMEE algorithms

As shown in Fig. 15, the probability density of errors by the KMEE algorithm is more similar to the probability density of the Gaussian mixture noise we set (shown in Fig. 1) that that by the KLMS, particularly for the two mean values (0 and 4) of the Gaussian mixture noise. This indicates that the KMEE algorithm performs better than the KLMS in filtering out the effect of the Gaussian mixture noise and making the filter outputs in line with the ideal desired response. In addition, we can find that the KMEE algorithm generates more concentrated error peaks whereas the KLMS generates wider error distributions with a larger error variance.

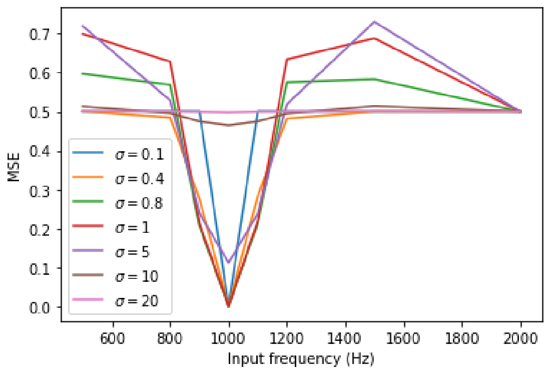
4) Generalization of the trained model by KMEE with different input frequencies and effects of kernel size and step size

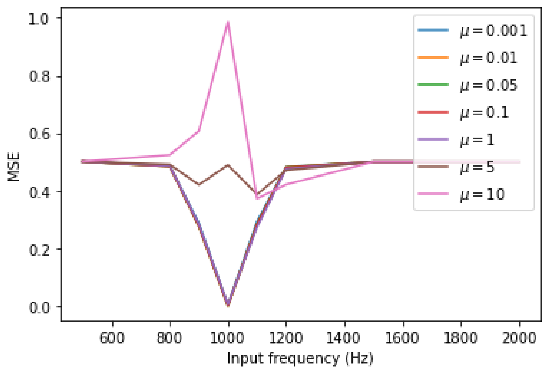


(1) Training MSE by iterations with different kernel sizes

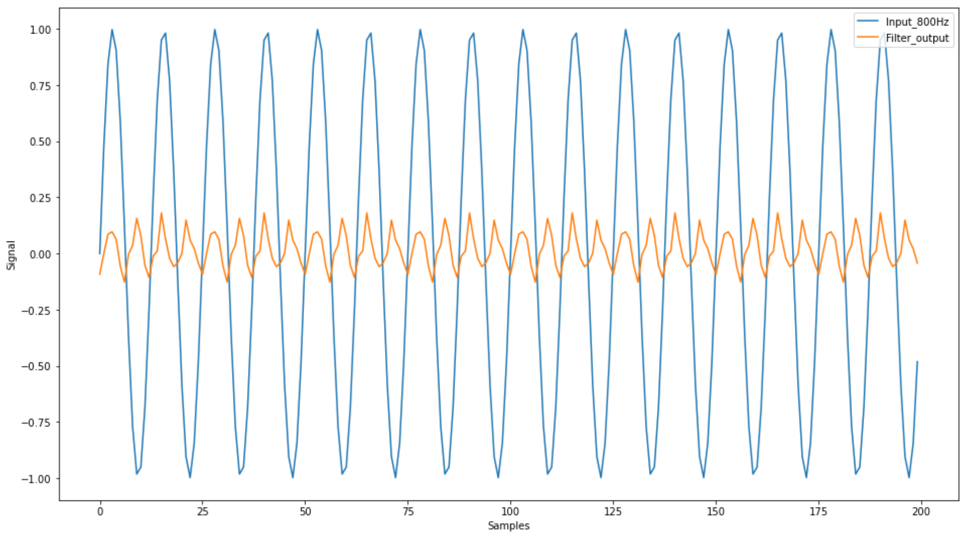


(2) Training MSE by iterations with different step sizes

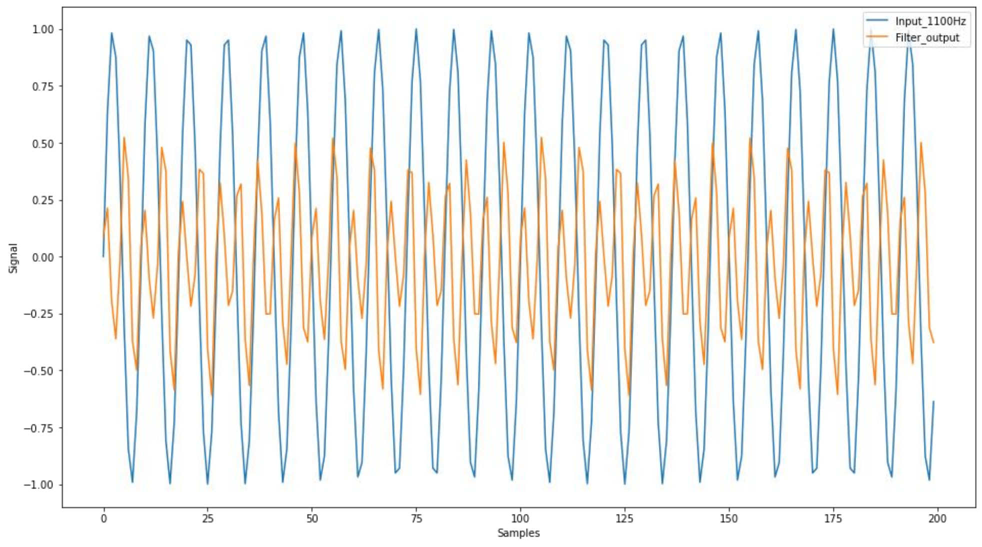




(3) Effect of kernel size on generalization (4) Effect of step size on generalization



(5) Curve of the first 200 filter outputs with an input frequency at 800Hz, $\sigma = 0.4$



(6) Curve of the first 200 filter outputs with an input frequency at 1100Hz, $\sigma = 0.4$

Fig. 16 Generalization of the trained frequency doubling model by KMEE

Generalization: Like the KLMS algorithm, the trained frequency doubling model by the KMEE algorithm does not shows a good generalization performance by different input frequencies, either. The KMEE algorithm with different kernel sizes exhibits the similar MSE behaviors at different input frequencies to the KLMS. Only when the input frequency is closer to 1000Hz, the nonlinear adaptive filter with KMEE achieves the best performance in doubling the input frequency of the sinewave and outperforms that with the KLMS algorithm.

Effect of kernel size: When the kernel size is less than 5, the nonlinear adaptive filter with KMEE achieves good performances in doubling the input frequency at 1000Hz, but the generalization performances are not very good as the input frequency is far away from 1000Hz. It is worth noting that when the kernel size is very small (0.1), the generalization performances of the nonlinear adaptive filter with KMEE in the range of 800-1200Hz are not as good as that with a larger kernel size. When the kernel size is higher than 5, the MSE of the nonlinear adaptive filter almost remains unchanged and is maintained at about 0.5, regardless of the input frequency of the sinewave. Compared with the KLMS algorithm, the KMEE algorithm performs better in the training MSE by iterations but does not show any superiority in generalization performances.

Effect of step size: When the step size is less than 1, the step size of the KMEE algorithm hardly affects the generalization performances of the nonlinear adaptive filter, showing almost the same changing trend of MSE by input frequency. This means that a change in the step size cannot greatly improve the generalization of the nonlinear adaptive filter in doubling other input frequencies of the sinewave. However, when the step size (5 and 10) is higher than 1, the nonlinear adaptive filter with KMEE performs badly, particularly for a larger step size (i.e., 10).

Major findings about comparison between the KMEE and KLMS algorithms in the nonlinear adaptive filter performances with Gaussian mixture noise:

(1) The convergence curve of the training MSE of the KMEE algorithm versus the KLMS algorithm suggest that the KMEE algorithm has the similar speed but better misadjustment.

(2) The KMEE algorithm performs better in signals with Gaussian mixture noise compared with the KLMS algorithm. The KMEE algorithm generates more concentrated error peaks while the KLMS generates wider error distributions.