Using the same "d(n)" and "x(n)" recordings as in assignment #3 and the same number of coefficients N=3, implement the following algorithms:

- LMS

- NLMS

- Newton (as in Assign #3)

- Steepest descent (as in Assign #3)

- Newton-LMS

- Affine Projection

- Recursive Least Squares

For the "learning curves", produce smoothed versions of the |e(n)|^2 error signal.

You must produce two plots, with perhaps additional zooms on the initial convergence and the steady state performance if you find that it helps comparing the algorithms:

For the "learning curves", produce smoothed versions of the |e(n)|^2 error signal.

You must produce two plots, with perhaps additional zooms on the initial convergence and the steady state performance if you find that it helps comparing the algorithms:

1) a plot where the step size "mu" (or the forgetting factor "lambda" in the RLS) has been adjusted for each algorithm, to have the fastest initial convergence in the learning curves. This allows to also compare the different misadjustment levels produced by the algorithms after convergence.

2) a plot where the step size "mu" (or the forgetting factor "lambda" in the RLS) has been adjusted so that each algorithm has roughly the same misadjustment in steady state (although the Newton and Steepest Descent algorithms do not suffer from misadjustment, theoretically). This allows to also compare the different initial convergence of the algorithms.

In each plot, also display with an horizontal line the performance (MMSE) achieved by the Wiener filter (computed from the MMSE equation, not from the smoothed |e(n)|^2).

One way to do this is using the Python function: plt.axhline(y=0.5, color='r', linestyle='-') (this would plot an horizontal line at y=0.5).

For each plot, you can choose the number of itrerations that you wish to use for coefficient update, the only limit will be set by the file size (959980 samples, about 20 seconds of audio at 48 kHz sampling rate).

For smoothing the |e(n)|^2 error signal, an all-positive linear phase FIR filter can be appropriate. An FIR linear phase filter with M+1 coefficients has a group delay of M/2 samples, and a settling time of M samples. You may have to discard the first M samples of the smoothed curved if they are too noisy. And you can use the value of the M/2 samples group delay to shift the smoothed |e(n)|^2 signal (or adjust the learning curve "iterations" x-scale), to have the same scale as for the original non-smoothed signals (no delay inserted).