THE RECURSIVE HESSIAN SKETCH FOR ADAPTIVE FILTERING

Robin Scheibler and Martin Vetterli

School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
{robin.scheibler,martin.vetterli}@epfl.ch

ABSTRACT

We introduce in this paper the recursive Hessian sketch, a new adaptive filtering algorithm based on sketching the same exponentially weighted least squares problem solved by the recursive least squares algorithm. The algorithm maintains a number of sketches of the inverse autocorrelation matrix and recursively updates them at random intervals. These are in turn used to update the unknown filter estimate. The complexity of the proposed algorithm compares favorably to that of recursive least squares. The convergence properties of this algorithm are studied through extensive numerical experiments. With an appropriate choice or parameters, its convergence speed falls between that of least mean squares and recursive least squares adaptive filters, with less computations than the latter.

***Index Terms*—** Recursive least squares, adaptive filtering, sketching.

1. INTRODUCTION

Adaptive filters are a cornerstone of classical statistical signal processing. They are routinely used for tasks such as system identification, echo cancellation, channel equalization, and beamforming [1]. These play a critical role in handsfree telephony, teleconferencing, digital communications, and many other practical systems. Two adaptive filters in particular have proved very popular: the least mean squares (LMS) and the recursive least squares (RLS) algorithms. On the one hand, LMS optimizes the mean squared error (MSE) using a stochastic gradient descent. On the other hand, RLS solves recursively a large least squares (LS) problem. While the former enjoys simplicity of implementation, low-complexity — linear in the filter length — and good stability properties, it sometimes lacks in terms of speed of convergence. The latter can offer a greater speed of convergence at the cost of a computational complexity quadratic in the filter length.

In recent years, a number of authors have presented approximation schemes relying on random projections to accelerate the solution of large LS problems [2, 3, 4]. Such methods are often referred to as sketching. Of particular interest is the iterative Hessian sketch (IHS) algorithm proposed by Pilanci and Wainwright [4]. Unlike other methods, the IHS algorithm gives sharp guarantees on how close the approximate solution is to that of the original LS problem, rather than just on the residual.

In this work, we propose the recursive Hessian sketch (RHS) algorithm. It is a randomized, approximate version of the RLS algorithm that relies on IHS to solve the underlying LS problem recursively. The benefit of this formulation is to allow a reduction in computational complexity at the cost of some convergence speed, thus bridging the gap between LMS and RLS. Rather than compute the exact inverse autocorrelation matrix at every step, RHS keeps N sketches that are updated independently with probability q at each round. At every update, the IHS iterations are run and a new filter is produced. Parameters N and q control the trade-off between complexity and convergence. Unfortunately, the convergence of IHS to the LS solution is only proved for sub-Gaussian and randomized orthonormal systems sketches [4]. Both of these sketches are not suitable for RHS and we rely instead on random row sampling. The proof of convergence of the algorithm is the topic of on-going work and we demonstrate its effectiveness through comprehensive numerical experiments only.

The rest of this paper is organized as follows. Section 2 introduces the necessary material about adaptive filters and the IHS algorithm. In Section 3, we describe the proposed algorithm and evaluate its complexity. The results of numerical experiments are presented in Section 4. We conclude in Section 5.

2. BACKGROUND

Throughout the paper we denote all matrices by bold upper case and vectors by bold lower case letters. The time index is $n \in \mathbb{N}$ and the Euclidean norm operation is $\|\mathbf{x}\| = (\mathbf{x}^T\mathbf{x})^{1/2}$.

2.1. Adaptive Filters

The adaptive filtering problem aims at finding an unknown filter $\mathbf{h} \in \mathbb{R}^d$ from a known reference signal x_n, and its filtered samples corrupted by noise $d_n = (x \star h)_n + v_n$, where $v_n \sim \mathcal{N}(0, \sigma_v^2)$ is additive white Gaussian noise, and $\star$ is the discrete convolution operation. The system is illustrated in Fig. 1.

Authors are with LCAV-EPFL. This work was supported by the Swiss National Science Foundation grant 200021_138081 – Non Linear Sampling Methods. All the code used to produce the results of this paper is available at `http://github.com/LCAV/sketchrls`

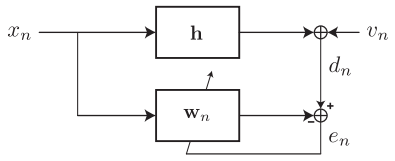

Fig. 1: Block-diagram of the adaptive filtering problem.

Two popular algorithms for adaptive filtering are the LMS and RLS algorithms [1]. The LMS algorithm is a stochastic gradient descent solving the following minimization problem

$$\mathbf{w}_n = \arg\min_{\mathbf{w}} \mathbb{E}|e_n|^2, \tag{1}$$

where $e_n = d_n - \mathbf{w}_n^T \mathbf{x}_n$. Computing the gradient of the cost function and dropping the expectation leads to the update rule

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e_n \mathbf{x}_n, \tag{2}$$

where μ is the step size. The normalized LMS (NLMS) algorithm solves the problem of choosing the value of μ by using the adaptive step size $\mu = \mu'/(\mathbf{x}^T\mathbf{x})$. In the absence of noise setting $\mu' = 1$ leads to the optimal learning rate [1].

The RLS algorithm solves the following exponentially weighted LS problem

$$\mathbf{w}_n = \arg\min_{\mathbf{w}} \left\| \mathbf{\Lambda}_n^{1/2}(\mathbf{X}_n\mathbf{w} - \mathbf{d}_n) \right\|^2, \tag{3}$$

where $\mathbf{X}_n = [\mathbf{x}_n \cdots \mathbf{x}_1]^T$, $\mathbf{x}_n = [x_n, \ldots, x_{n-d+1}]^T$, $\mathbf{d}_n = [d_n, \ldots, d_1]^T$, $\mathbf{\Lambda}_n = \mathrm{diag}(1, \ldots, \lambda^{n-1})$ and λ is a forgetting factor allowing the algorithm to adapt to a time-varying filter. The LS solution to this problem is $\mathbf{w}_n = \mathbf{R}_n^{-1}\mathbf{b}_n$ where $\mathbf{R}_n = \mathbf{X}_n^T\mathbf{\Lambda}_n\mathbf{X}_n$ and $\mathbf{b}_n = \mathbf{X}_n^T\mathbf{\Lambda}_n\mathbf{d}_n$. This solution can be recursively computed from $\mathbf{w}_{n-1}$, $\mathbf{x}_n$, and d_n. Since $\mathbf{X}_n = [\mathbf{x}_n\ \mathbf{X}_{n-1}^T]^T$ and $\mathbf{d}_n = [d_n\ \mathbf{d}_{n-1}^T]^T$, we have

$$\mathbf{R}_n = \lambda\mathbf{R}_{n-1} + \mathbf{x}_n^T\mathbf{x}_n, \quad \mathbf{b}_n = \lambda\mathbf{b}_{n-1} + d_n\mathbf{x}_n, \tag{4}$$

$$\mathbf{P}_n = \mathbf{R}_n^{-1} = \lambda^{-1}\mathbf{P}_{n-1} - \lambda^{-1}\frac{\mathbf{P}_{n-1}\mathbf{x}_n\mathbf{x}_n^T\mathbf{P}_{n-1}}{\lambda + \mathbf{x}_n^T\mathbf{P}_{n-1}\mathbf{x}_n}, \tag{5}$$

where the last update is obtained from the recursion for $\mathbf{R}_n$ and the matrix inversion lemma [5]. A little algebra yields the final form of the algorithm as described in Algorithm 1[1].

2.2. Iterative Hessian Sketch

In general, sketching methods use random projections to reduce the size of an LS problem with guarantees that the solution to the reduced problem stays close to the original problem. The goal is a reduction in computation. The IHS is such a method recently proposed by Pilanci and Wainwright to solve a class of constrained optimization problems of the form

$$\min_{\mathbf{x}\in\mathcal{C}} \frac{1}{2n}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 \tag{6}$$

[1]Algorithm 1 is block RLS with $\mathbf{X}_{n,L} = [\mathbf{x}_{n+L} \cdots \mathbf{x}_{n+1}]^T$ and $\mathbf{d}_{n,L} = [d_{n+L}, \ldots, dn+1]^T$. Setting $L = 1$ yields standard RLS.

Algorithm 1 BlockRLS

Require: λ, δ, $\mathbf{P}_0 = \delta^{-1}I_d$, $\mathbf{w}_0 = 0$
Ensure: $\mathbf{w}_n = \arg\min_{\tilde{\mathbf{w}}} \|\mathbf{\Lambda}_n^{1/2}(\mathbf{X}_n\tilde{\mathbf{w}} - \mathbf{d}_n)\|_2^2$
  for every L new samples do:
  $\mathbf{Z} \leftarrow \mathbf{P}_n\mathbf{X}_{n,L}^T$
  $\mathbf{G} \leftarrow \lambda^{-L}\mathbf{Z}(\mathbf{\Lambda}_L^{-1} + \lambda^{-L}\mathbf{X}_{n,L}\mathbf{Z})^{-1}$
  $\mathbf{w}_{n+L} \leftarrow \mathbf{w}_n + \mathbf{G}(\mathbf{d}_{n,L} - \mathbf{X}_{n,L}\mathbf{w}_n)$
  $\mathbf{P}_{n+L} \leftarrow \lambda^{-L}(\mathbf{P}_n - \mathbf{G}\mathbf{Z}^T)$

where $\mathcal{C}$ is a constraint set [4], $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\mathbf{x} \in \mathbb{R}^d$, and $\mathbf{y} \in \mathbb{R}^n$. In this work, we always have $\mathcal{C} = \mathbb{R}^d$, and the problem simplifies to unconstrained LS. In that case, the IHS algorithm reduces to the following iterative process

$$\mathbf{x}_i = \mathbf{x}_{i-1} + (\mathbf{A}^T\mathbf{S}_i^T\mathbf{S}_i\mathbf{A})^{-1}\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}_{i-1}) \tag{7}$$

for $i = 1, \ldots, N$, $\mathbf{x}_0 = 0$, and $\{\mathbf{S}_i \in \mathbb{R}^{m\times n}\}_{i=1}^N$, $m \leq n$, are sketching matrices chosen independently at random. Sketching matrices can be, for example, matrices with normal iid entries, the so-called fast Johnson-Lindenstrauss transform [6], or a matrix sampling the rows of $\mathbf{A}$ at random. Their key property is to preserve the norms of the columns of $\mathbf{A}$ up to some controlled distortion. They must satisfy $\mathbb{E}[\mathbf{S}^T\mathbf{S}] = \mathbf{I}_n$.

3. RECURSIVE HESSIAN SKETCH

3.1. Algorithm

The idea of the recursive Hessian sketch (RHS) is to apply the IHS algorithm to solve (3) recursively as new data streams in. To derive the RHS algorithm we follow a strategy similar to that of RLS and add the sketching matrix $\mathbf{S}$ in the problem. Instead of computing the inverse autocorrelation matrix $\mathbf{P}$ explicitly at every step, we maintain N sketched matrices $\{\tilde{\mathbf{P}}_i\}_{i=1}^N$ that we update with probability q. Compared to conventional RLS, such a scheme only requires to do an inverse matrix update every $(Nq)^{-1}$ iterations on average. Provided the update of a sketched matrix $\tilde{\mathbf{P}}_i$ can be done efficiently, we hope to have a computational complexity gain. We expect this approximate scheme to have an impact on the speed of convergence of the algorithm. By varying the parameters, we hope to get an interesting trade-off between complexity and convergence.

The algorithm needs a sketching matrix that can be updated in an online fashion. A variant of the random row sampling proposed in [4] can be recursively defined

$$\mathbf{S}_n = \begin{bmatrix} b_n/\sqrt{q} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{n-1} \end{bmatrix}, \quad b_n \sim \begin{cases} 1 & \text{w.p. } q \\ 0 & \text{w.p. } 1-q \end{cases}. \tag{8}$$

Note that in contrast to the random row sampling of [4] that selects a deterministic number of rows, the number of rows of $\mathbf{S}_n$ is a binomial random variable $m \sim \mathcal{B}(n, q)$. However, as n grows large, it becomes tightly concentrated around its mean $\mathbb{E}[m] = nq$. One can verify that $\mathbb{E}[\mathbf{S}^T\mathbf{S}] = \mathbf{I}_n$

Such a sketching matrix can be implemented in the following way. For every new sample received, N random samples $\{b_i\}_{i=0}^{N-1}$ are drawn independently from a Bernoulli(q) distribution (as in (8)). If at least one of $b_i = 1$, an update is done.

Algorithm 2 RHS

Require: λ, δ, N, q, $\mathbf{R} = \delta \mathbf{I}_d$, $\{\tilde{\mathbf{P}}_i = \delta^{-1}\mathbf{I}_d\}_{i=1}^N$, $\mathbf{w}, \mathbf{b} = 0$.
Ensure: $\mathbf{w} = \arg\min_{\tilde{\mathbf{w}}} \|\mathbf{\Lambda}_n^{1/2}(\mathbf{X}_n\tilde{\mathbf{w}} - \mathbf{d}_n)\|_2^2$

```
Draw at random {b_i}_{i=1}^N ~iid Bernoulli(q)
if b_i = 1 for some i = 0, ..., N-1 then
    L ← n − k_r
    k_r ← n
    R ← λ^L R + X_{n,L}^T Λ_L X_{n,L}
    b ← λ^L b + X_{n,L}^T Λ_L d_{n,L}
    for i = 1, ..., N do
        if b_i = 1 then
            L ← n − k_p(i)
            k_p(i) ← n
            z ← P̃_i x_n
            P̃_i ← λ^{-L} ( P̃_i − zz^T / (qλ^L + x_n^T z) )
        end if
        w ← w + (1/n) P̃_i (b − Rw)
    end for
end if
```

First, $\mathbf{R}_n$ and $\mathbf{b}_n$ are updated with all new samples received since the last update. This can be done as a block update. Then, for all $\{i : b_i = 1\}$, we update the corresponding $\tilde{\mathbf{P}}_i$. Assuming L samples arrived since the previous update, we have

$$\tilde{\mathbf{R}}_n = \mathbf{X}_n^T \mathbf{\Lambda}_n^{1/2} \mathbf{S}_n^T \mathbf{S}_n \mathbf{\Lambda}_n^{1/2} \mathbf{X}_n = q^{-1}\mathbf{x}\mathbf{x}^T + \lambda^L \tilde{\mathbf{R}}_{n-L} \quad (9)$$

and using again the matrix inversion lemma, as in RLS,

$$\tilde{\mathbf{P}}_n = \lambda^{-L}\tilde{\mathbf{P}}_{n-L} - \lambda^{-L}\frac{\tilde{\mathbf{P}}_{n-L}\mathbf{x}_n\mathbf{x}_n^T\tilde{\mathbf{P}}_{n-L}}{q\lambda^L + \mathbf{x}_n^T\tilde{\mathbf{P}}_{n-L}\mathbf{x}_n}. \quad (10)$$

It is important to note that whereas block RLS would require a rank-L update, RHS only requires N rank-1 update, which can be significantly cheaper provided $N \ll L$. The final step is to run the N iterations of IHS algorithm

$$\mathbf{w}_{i,n} = \mathbf{w}_{i-1,n} + \frac{1}{n}\tilde{\mathbf{P}}_{i,n}\left(\mathbf{b}_n - \mathbf{R}_n\mathbf{w}_{i-1,n}\right), \quad i = 1, \ldots, N, \quad (11)$$

with $\mathbf{w}_{0,n} = \mathbf{w}_{N,n-L}$, the solution produced by the algorithm during the previous update. Pseudocode is given in Algorithm 2.

3.2. Complexity Analysis

The RHS is a block update algorithm but where the block size is random. In general, performing block updates allows some computational gain compared to updating at every sample. For this reason, we compare the complexity of RHS to that of the block RLS described in Algorithm 1. Although block RLS only updates the filter every L sample, its output is exactly that of conventional RLS at the corresponding sample. The main advantage of block update is to make use of fast multiplication by $\mathbf{X}_{n,L}$. Since it is an $L \times d$ Hankel matrix, multiplying a vector by $\mathbf{X}_{n,L}$, or its transpose, has complexity $O((d+L)\log(d+L))$. This can be done by flipping upside-down the rows to obtain a Toeplitz matrix, take its circulant extension and use the FFT algorithm to compute the product [7].

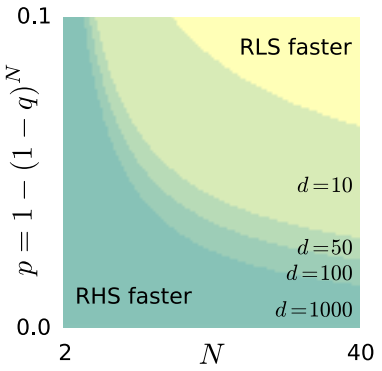

Fig. 2: The shaded areas indicate where RHS has lower complexity than block RLS for various values of d.

For the comparison, we only establish an approximate count of operations. This is justified for several reasons. First, both algorithms use the same primitives, that is matrix-vector and matrix-matrix products as well as FFT, and thus the big-O constants should be approximately the same for both algorithms. Second, the exact runtime of these primitives is highly dependent on the implementation and the architecture of the machine used. Our goal here is to show that there is an interesting regime for the RHS algorithm.

Let us start with the Block RLS complexity. Counting from Algorithm 1 and dividing by the block size L, we obtain

$$C_{\text{BRLS}} = O\big((d+L+1)(d+L)\log(d+L)/L + d^2 + (2d^2+d)/L + (L+1)d + L^2\big), \quad (12)$$

where d is the dimension of the filter. We assumed a naive algorithm for the multiplication by a non-Toeplitz matrix. Notice that there is a quadratic term in d independent of the block size. This means that the asymptotic complexity of block RLS is identical to that of conventional RLS.

We can do a similar count for the RHS algorithm described in Algorithm 2. Since the algorithm is not deterministic, we will use the average complexity. When the number of samples is large, the actual complexity should be very close to its expectation. Let us firs compute the update probability p. An update happens only if at least one of $b_i = 1$, which happens with probability $p = 1 - (1-q)^N$. Then, the average number of sketched matrices to update is $\mathbb{E}\sum b_i = Nq$. Finally, the update probability is fixed to $p = 1/L$ so that the average block size is the same as that of block RLS. Adding the N iterations of IHS we obtain the following average complexity per sample

$$C_{\text{RHS}} = O\big((d+1)(d+L)\log(d+L)/L + 3d/L + 3Nq(d^2+d) + (2d^2+3d)N/L\big) \quad (13)$$

Fig. 2 shows regions of the (N, p) space where RHS has a lower computational complexity than block RLS. We observe that even for large filter lengths d there are significant regions where a gain can be obtained. In the following section, numerical experiments will reveal that these regions are not incompatible with a fast convergence rate.

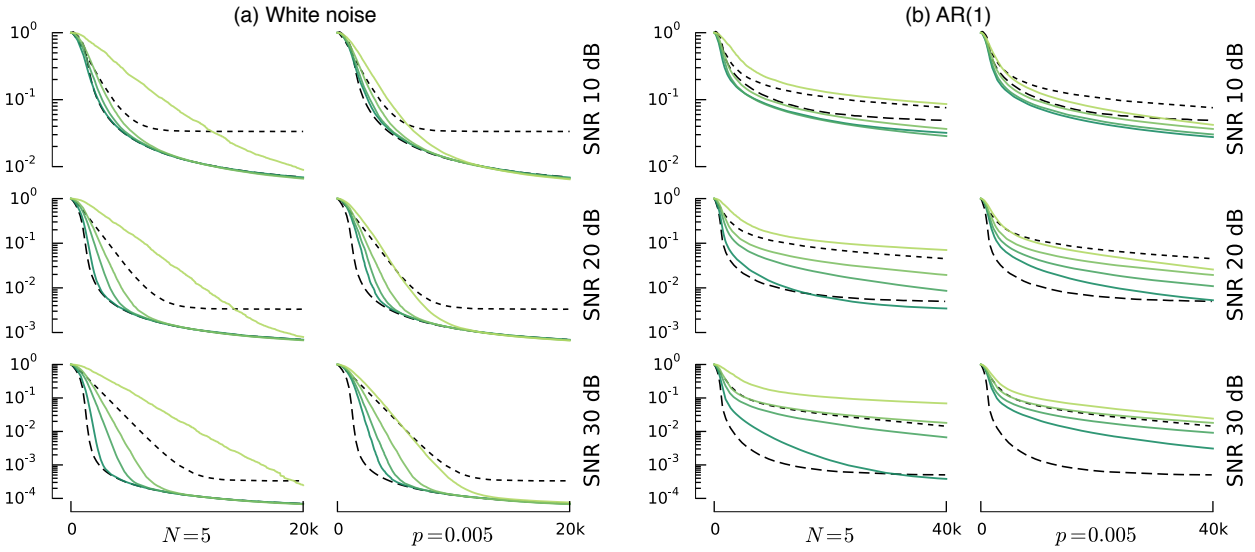

Fig. 3: Evaluation of the MSE over time for 300 realizations of an adaptive filter of length $d = 1000$. The driving signal x_n is (a) white noise and (b) AR(1) process. The short and long dashed lines are NLMS and RLS respectively. The solid lines are RHS with, on the right, fixed $N = 5$ and from darker to lighter $p = 0.05, 0.01, 0.005, 0.001$, and, on the left, fixed $p = 0.005$ and from darker to lighter $N = 20, 10, 5, 2$. From top to bottom the SNR is 10, 20, and 30 dB. The step size of NLMS is $\mu = 0.5$ in all cases except (b) 30 dB, where it is $\mu = 1$. For RLS and RHS, $\delta = 20, 10, 10$ at SNR 10, 20, 30 dB respectively. The forgetting factor is fixed to $\lambda = 0.9999$.

4. NUMERICAL EXPERIMENTS

In this section we assess the practical performance of the proposed algorithm and compare it to that of the NLMS and RLS algorithms. For the experiment two driving signals x_n are used, unit variance white noise, and an autoregressive process of order 1 (AR(1)) generated by filtering white noise with filter $[1, 0.9]$. The unknown filter $\mathbf{h}$ is sampled uniformly at random from the sphere $\mathbb{S}^{d-1}$ with $d = 1000$ and the reference signal d_n is generated by convolving x_n with $\mathbf{h}$ and adding white Gaussian noise with variance $\sigma_v = 10^{-\frac{\text{SNR}}{20}}$, SNR $= 10, 20, 30$ dB. For RLS and RHS, the regularization parameter is fixed to $\delta = 20, 10, 10$ for SNR 10, 20, and 30 dB, respectively. The forgetting factor is fixed to $\lambda = 0.9999$ to ensure stability. The step size for NLMS is fixed to $\mu = 0.5$ in all cases except in the case of AR(1) at 30 dB SNR where $\mu = 1$. These step sizes were picked to balance speed of convergence and residual errors. Two experiments are done with the parameters of RHS. First, N is fixed to 5 and p is varied over 0.001, 0.005, 0.01, and 0.05. Second, p is fixed to 0.005 and N is varied over 2, 5, 10, and 20. The filter is then run until $n = 20000$ and $n = 40000$ for the white noise and AR(1) input signals, respectively.

In Fig. 3, the evolution of the MSE over time for 300 realizations of the adaptive algorithms is plotted for all configurations just described. From the simulation results we observe that RHS can have a fast convergence. For all $p \geq 0.005$ tested, it is faster than NLMS. While slower than RLS in most cases, RHS reaches the same residual errors much lower than that of NLMS. The exception is for AR(1) driving signal at 10 dB SNR where surprisingly RHS has faster convergence than RLS and seems to attain a lower residual error. In general performance degrades for AR(1) signal compared to white noise, but RHS maintains acceptable performance, somewhat between that of NLMS and RLS.

5. CONCLUSION

Leveraging recent advances in sketching for solving least squares problems, we proposed the recursive Hessian sketch (RHS), a new adaptive filtering algorithm solving the same exponentially weighted least squares problem as the conventional RLS but in an approximate way. Two parameters, the number of sketches N and the update probability q, control the computational complexity and the convergence of the algorithm. We found that there are interesting operating points where the computational complexity is lower than that of RLS while maintaining a fast convergence and low residual error. This was demonstrated through numerical experiments for a large number of combinations of parameters.

Left to prove is the convergence of the IHS algorithm — on which RHS relies — when using the random row sampling sketch. Another open question is the general stability of the algorithm over a large range of parameters and driving signals. Of particular interests is its performance on real world signals such as speech or music.

6. REFERENCES

[1] S. Haykin, *Adaptive filter theory*. Prentice Hall, 2014.

[2] C. Boutsidis and P. Drineas, "Random projections for the nonnegative least-squares problem," *Linear algebra and its applications*, vol. 431, no. 5-7, pp. 760–771, 2009.

[3] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlos, "Faster least squares approximation," *Numerische mathematik*, vol. 117, pp. 219–249, Feb. 2011.

[4] M. Pilanci and M. J. Wainwright, "Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares," 2014, arXiv:1411.0347v1.

[5] M. A. Woodbury, "Inverting modified matrices," *Memorandum report*, vol. 42, p. 106, 1950.

[6] N. Ailon and B. Chazelle, "The Fast Johnson–Lindenstrauss transform and approximate nearest neighbors," *SIAM Journal on computing*, vol. 39, pp. 302–322, Jan. 2009.

[7] R. H. Chan and M. K. Ng, "Conjugate Gradient Methods for Toeplitz Systems," *SIAM Review*, vol. 38, pp. 427–482, Sept. 1996.