Null Space Projection Enhanced LMS Filters

Michael Lunglmayr, *Member, IEEE*, Thomas Paireder, *Student Member, IEEE*,
and Mario Huemer, *Senior Member, IEEE*

***Abstract*—The least mean squares (LMS) filter is one of the most important adaptive filters used in digital signal processing applications. We present a performance improvement method for LMS filters based on null space projection. The approach uses buffering and a subsequent null space projection denoising. Interestingly, while the performance of this approach scales with the buffer length, the complexity in terms of multiplications per sample does not. We show the performance gains obtained by this method and present an architecture implementing an LMS filter as well as the proposed null space projection enhancement. We give complexity comparisons as well as synthesis results for a field-programmable gate array (FPGA) showing the low complexity overhead of the proposed method. We furthermore present hardware validated bit-true simulation results demonstrating the performance capabilities of the approach.**

I. Introduction

ADAPTIVE filters are important in many fields of digital signal processing ranging from radar and communications engineering to speech and bio-medical applications [1]. One of the most important adaptive filters is the least mean squares (LMS) filter [2]. In this work, we present a low complexity improvement for LMS filters based on the null space projection capabilities of Kaczmarz-like algorithms [3], [4], [5]. For this, we draw a connection between the LMS filter and Kaczmarz algorithms and present a novel method to improve the performance of LMS filters using a null space projection enhancement.

The LMS filter can be seen as a stochastic gradient descent variant of the Kaczmarz algorithm [3]. The Kaczmarz algorithm is often used for least squares estimation problems using the model

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{n}, \tag{1}$$

with the corresponding least squares solution

$$\mathbf{w}_{\text{LS}} = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2. \tag{2}$$

Here, $\mathbf{y}$ is a known vector, typically obtained from measurements, $\mathbf{X}$ is a known matrix, often called system matrix, and $\mathbf{n}$ is an unknown noise vector. $\mathbf{w}$ is the vector to be estimated. The dimensions of the vectors are naturally defined by the dimensions of the matrix $\mathbf{X}$: $m \times p$. In this work, we will always assume that $\mathbf{X}$ has full rank and that $m > p$. The Kaczmarz algorithm will only converge to the exact solution if the noise vector is all-zero, otherwise it only approximately solves the least squares estimation problem. The approximation error depends on the noise power as well as the step size μ_k, as used in Alg. 1. The smaller μ_k becomes, the closer the algorithm's result will be to $\mathbf{w}_{\text{LS}}$. On the other hand, using large step sizes results in a fast reduction of the estimation error in early iterations. It can be shown that the maximum practically relevant step size of the algorithm is given by[1] $\frac{1}{\|\mathbf{x}_{k^\neg}\|_2^2}$, with $\mathbf{x}_{k^\neg}^T$ being a row of the matrix $\mathbf{X}$. Considering these counteracting effects of large and small step sizes, several strategies for choosing μ_k, typically starting with a large value in the beginning and reducing the step size over the iterations, have been proposed [6], [7]. For simplicity, we will not further consider such step size strategies in this work. Such strategies can of course be performed in addition to the proposed approach.

Algorithm 1 Kaczmarz Algorithm

Input: $\hat{\mathbf{w}}^{(0)} = \mathbf{0}$, $\mathbf{X}$, $\mathbf{y}$
Output: $\hat{\mathbf{w}}^{(N)}$
1: **for** $k = 1, \ldots, N$ **do**
2: $\quad k^\neg = (k-1) \bmod m + 1$
3: $\quad \hat{\mathbf{w}}^{(k)} = \hat{\mathbf{w}}^{(k-1)} - \mu_k \mathbf{x}_{k^\neg}(\mathbf{x}_{k^\neg}^T \hat{\mathbf{w}}^{(k-1)} - y_{k^\neg})$
4: **end for**

Analyzing Alg. 1, one can see the resemblance to the LMS algorithm. When using the step size $\frac{1}{\|\mathbf{x}_{k^\neg}\|_2^2}$, the Kaczmarz algorithm closely resembles the normalized LMS (NLMS) [8].

A difference between the algorithms is that the Kaczmarz algorithm uses the vectors $\mathbf{x}_{k^\neg}$ out of a finite set (the m rows of $\mathbf{X}$), while the LMS (due to its stream-based operation) always uses potentially different vectors $\mathbf{x}_k$ at every iteration k. This finite set of rows of the Kaczmarz algorithm requires re-using the vectors $\mathbf{x}_{k^\neg}$ if the number of iterations N of the algorithm is larger than the number of rows m. In Alg. 1, this is done cyclically (selected by step 2), however, other strategies such as selecting the rows randomly [9] have been proposed as well. Typically, Alg. 1 is initialized with $\mathbf{w}^{(0)} = \mathbf{0}$.

A less commonly known variant of the Kaczmarz algorithm is when using the model

$$\mathbf{X}^T\mathbf{z} = \mathbf{0}. \tag{3}$$

As it is described in [4], the algorithm then projects its input $\mathbf{z}^{(0)}$ on the null space of $\mathbf{X}^T$. When initializing the Kaczmarz algorithm with $\mathbf{z}^{(0)} = \mathbf{y}$, this means that the algorithm converges to $\mathbf{y} - \mathbf{X}\mathbf{w}_{\text{LS}}$, which can be interpreted as the least squares error signal. Alg. 1 only converges to the exact solution if the step size goes to zero [10], [11]. In contrast, for this null space projection application, the Kaczmarz algorithm converges to the exact solution for *any* step size (below the maximum to prevent divergence). According to the authors' experience, this is not well known to the engineering community. For this reason we give a concise proof of the

Manuscript received ;
M. Lunglmayr, T. Paireder and M. Huemer are with the Institute of Signal Processing, Johannes Kepler University Linz, 4040 Linz, Austria (e-mail: michael.lunglmayr@jku.at).
Digital Object Identifier

[1] Convergence is guaranteed for a positive step size lower than $2/\|\mathbf{x}_{k^\neg}\|_2^2$; however, using step sizes larger than $1/\|\mathbf{x}_{k^\neg}\|_2^2$ does not provide an advantage for this work.

convergence of the null space projection Kaczmarz algorithm that is only based on linear algebra in Section II.

This concept is not to be confused with so-called extended Kaczmarz algorithms, where null space projection steps are interleaved with iterations of the ordinary Kaczmarz algorithm, to enhance its convergence [4], [5], [12], [13]. In an adaptive filter context, such an interleaved application is inappropriate because the input samples arrive in a stream-based manner. In this work, however, we develop a novel architecture allowing utilizing the capabilities of null space projection in an adaptive filter context. For this reason, we incorporate a buffering and memory access scheme, allowing storing the input data and processing it using null space projection. We use null space projection to improve the performance of a subsequent LMS filter operation and use results of previous LMS iterations to speed up null space projection. This allows significant performance improvements while still maintaining a moderate computational complexity.

II. Null space projection Kaczmarz

Any vector $\mathbf{y} \in \mathbb{R}^m$ can be represented as $\mathbf{y} = \mathbf{y}_c + \mathbf{y}_0$, with $\mathbf{y}_c$ lying in the column space of $\mathbf{X}$ and $\mathbf{y}_0$ lying in the null space of $\mathbf{X}^T$. This is because these two orthogonal subspaces together span $\mathbb{R}^m$ [14]. Let $\mathbf{x}_{*,i}$ be the i^{th} column of $\mathbf{X}$, i.e. $\mathbf{x}_{*,i}^T$ the i^{th} row of $\mathbf{X}^T$. Using the Kaczmarz algorithm on the model $\mathbf{X}^T\mathbf{z} = \mathbf{0}$ with the step size $\mu_k = \frac{1}{\|\mathbf{x}_{*,k^\neg}\|_2^2}$ results in the cyclic update equations

$$\hat{\mathbf{z}}^{(k)} = \hat{\mathbf{z}}^{(k-1)} - \frac{1}{\|\mathbf{x}_{*,k^\neg}\|_2^2}\mathbf{x}_{*,k^\neg}\mathbf{x}_{*,k^\neg}^T\hat{\mathbf{z}}^{(k-1)} \tag{4}$$

$$= (\mathbf{I} - \frac{1}{\|\mathbf{x}_{*,k^\neg}\|_2^2}\mathbf{x}_{*,k^\neg}\mathbf{x}_{*,k^\neg}^T)\hat{\mathbf{z}}^{(k-1)}. \tag{5}$$

$\hat{\mathbf{z}}^{(k-1)}$ can be represented as $\hat{\mathbf{z}}_c^{(k-1)} + \hat{\mathbf{z}}_0^{(k-1)}$, where $\hat{\mathbf{z}}_c^{(k-1)}$ is an element of the column space of $\mathbf{X}$ and $\hat{\mathbf{z}}_0^{(k-1)}$ is an element of the null space of $\mathbf{X}^T$. When initializing $\hat{\mathbf{z}}^{(0)} = \mathbf{y}$, the null space component obviously is $\hat{\mathbf{z}}_0^{(0)} = \mathbf{y}_0$. Due to the linearity of the matrix multiplication in (5), one can apply the Kazmarz update separately on the column space and the null space components. Analyzing the update of the null space component gives

$$\hat{\mathbf{z}}_0^{(1)} = (\mathbf{I} - \frac{1}{\|\mathbf{x}_{*,k^\neg}\|_2^2}\mathbf{x}_{*,k^\neg}\mathbf{x}_{*,k^\neg}^T)\hat{\mathbf{z}}_0^{(0)} = \hat{\mathbf{z}}_0^{(0)}, \tag{6}$$

because $\mathbf{x}_{*,k^\neg}^T\hat{\mathbf{z}}_0^{(0)} = 0$ (the vector $\mathbf{x}_{*,k^\neg}$ out of the column space of $\mathbf{X}$ is orthogonal to a vector out of the null space of $\mathbf{X}^T$). As this obviously also holds for the following iterations, in this use case, the null space component is unaltered during the iterations. As we will show in the following, the column space component becomes the all-zero vector if the number of iterations goes to infinity, thus justifying the name, null space projection Kaczmarz.

Analyzing the update of the column space component

$$\hat{\mathbf{z}}_c^{(k)} = (\mathbf{I} - \frac{1}{\|\mathbf{x}_{*,k^\neg}\|_2^2}\mathbf{x}_{*,k^\neg}\mathbf{x}_{*,k^\neg}^T)\hat{\mathbf{z}}_c^{(k-1)}, \tag{7}$$

one can see that this is an orthogonal projection of $\hat{\mathbf{z}}_c^{(k-1)}$ on the null space of $\mathbf{x}_{*,k^\neg}^T$ using the projection matrix $(\mathbf{I} - \frac{1}{\|\mathbf{x}_{*,k^\neg}\|_2^2}\mathbf{x}_{*,k^\neg}\mathbf{x}_{*,k^\neg}^T)$. This is because for a projection matrix $\mathbf{P}$ it holds that $\mathbf{I} - \mathbf{P}$ is a also a projection matrix. $\mathbf{I} - \mathbf{P}$ projects on the orthogonal complement of the space where $\mathbf{P}$ projects to [14]. For orthogonal projections with a matrix $\mathbf{P}$ it holds that $\|\mathbf{P}\mathbf{v}\|_2 \leq \|\mathbf{v}\|_2$ for an arbitrary vector $\mathbf{v}$ [14].

A projected vector (and of course also its norm) stays the same when the vector is already in the space the projection matrix projects onto. The column space component $\hat{\mathbf{z}}_c^{(k-1)}$ can only be in the null space of $\mathbf{x}_{*,k^\neg}^T$ if it is already the zero vector. In all other cases, its norm is reduced. As the number of iterations goes to infinity, $\hat{\mathbf{z}}_c^k$ is reduced to the all-zero vector.

III. Null space projection enhanced LMS Filter

In this section we describe a null space projection enhancement (NPE) for noise reduction of the measurements for an LMS filter. In Alg. 2, the pseudo code of this approach is described. For latter reference, a typical (simplified) LMS architecture is shown in Fig. 1. For simplifying the description

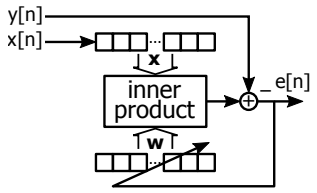

Fig. 1. Basic LMS architecture.

of the NPE we assume to have the inputs $x[n]$ of the LMS filter combined to a vector $\mathbf{x} \in \mathbb{R}^{l+p-1}$ as well as the measurements $y[n]$ combined into a vector $\mathbf{y} \in \mathbb{R}^l$. These vectors will act as buffer memories to perform the NPE. The used filter function

Algorithm 2 Null space projection enhancement

Input: $\hat{\mathbf{w}}$, $\mathbf{x}$, $\mathbf{y}$, l, p
Output: $\tilde{\mathbf{y}}$

```
z^(0) = y − filter(x, ŵ)
for j = 1, ..., p do
    x_j = [x_{p−j}, ..., x_{p−j+l}]^T
    μ_j = 1/‖x_j‖_2^2
    z^(j) = z^(j−1) − μ_j x_j x_j^T z^(j−1)
end for
ỹ = y − z^(p)
```

in Alg. 2 is assumed to perform a convolution of the two input vectors (an FIR filtering operation) and to return an output vector that is as long as the measurement vector.

The null space projection enhancement will be performed in parallel to the LMS operation. For this, we use two pairs of vectors (i.e. memories) $\mathbf{x}_1$, $\mathbf{y}_1$ and $\mathbf{x}_2$, $\mathbf{y}_2$, respectively. While the LMS filter processes one pair, the null space projection enhancement is used for noise reduction on the measurements of the other pair. After the null space projection enhancement, the LMS filter will use the noise reduced measurements (by NPE) for better estimation performance. By ensuring a fast enough processing of the NPE, this block based algorithm will work seamlessly in combination with the stream-based nature of the LMS filter. After l iterations of the LMS filter, the memories for $\mathbf{x}_1$, $\mathbf{y}_1$ and $\mathbf{x}_2$, $\mathbf{y}_2$, respectively, are exchanged and the current estimate $\hat{\mathbf{w}}$ of the LMS filter is handed over to

the NPE. There it is used to speed up the null space projection by using $\mathbf{y} - \text{filter}(\mathbf{x}, \hat{\mathbf{w}})$ as a closer starting vector to the least squares error and therefore requiring less iterations for convergence. With this speed up, we found that typically p iterations are sufficient for the null space enhancement of an LMS filter. This low number of null space projection iterations allows the parallel processing of NPE with low complexity.

Fig. 2 shows the performance of the null space projection enhanced LMS (NPE-LMS) for the system identification scenario described in Sec. IV-C. For different block lengths $l \in \{5p, 10p, 15p\}$ and a fixed signal-to-noise ratio (SNR) of 20 dB the figure illustrates the root mean squared error (RMSE) norms $\sqrt{\text{mean}\|\mathbf{w} - \hat{\mathbf{w}}^{(k)}\|_2^2}$ over the iterations k. The RMSE is averaged over 1000 test cases at each iteration. As one can see in this figure, a significant performance improvement over the NLMS filter can already be obtained by using block lengths of $l = 5p$. As a reference, we included the performance of a recursive least squares (RLS) filter [8]. As we comment in Sec. IV-C, the computational complexity of the NPE-LMS is significantly below that of an RLS filter. Due to similar numerical properties, the null space projection can be implemented in digital hardware with the same low bit width requirements as an LMS filter.

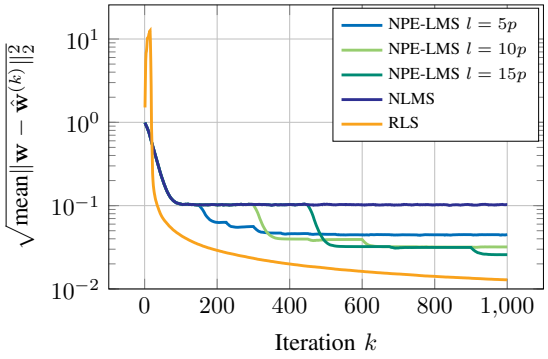

Fig. 2. RMSE of LMS, NPE-LMS and RLS.

As it can be seen in Fig. 2, the noise reduction directly depends on the ratio of block length to parameter count. We prove this observation by examining the mean reduction of noise power while assuming $\mathbf{z}$ has converged to the least squares solution $\mathbf{z}_{\text{LS}}$. In the following, the subscript l denotes that the corresponding vectors and matrices have l rows. With

$$\mathbf{z}_{\text{LS}} = (\mathbf{I}_l - \mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T)\,\mathbf{y}_l \tag{8}$$

the remaining noise after denoising is

$$\mathbf{n}_{\text{den,LS}} = \mathbf{y}_l - \mathbf{z}_{\text{LS}} = \mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\,\mathbf{n}_l. \tag{9}$$

Assuming $\mathbf{n}_l$ to be zero mean, we obtain the following reduction in noise power

$$\begin{aligned}
\frac{P_{\mathbf{n},\text{den}}}{P_{\mathbf{n}}} &= \frac{\text{tr}(\text{E}[\mathbf{n}_{\text{den,LS}}\,\mathbf{n}_{\text{den,LS}}^T])}{\text{tr}(\text{E}[\mathbf{n}_l\,\mathbf{n}_l^T])} \\
&= \frac{\text{tr}\left(\mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\,\text{E}[\mathbf{n}_l\,\mathbf{n}_l^T]\,\mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\right)}{\text{tr}(\text{E}[\mathbf{n}_l\,\mathbf{n}_l^T])} \\
&= \frac{\text{tr}\left(\mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\mathbf{C}_{\mathbf{nn}}\,\mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\right)}{\text{tr}(\mathbf{C}_{\mathbf{nn}})} \\
&= \frac{\sigma_{\mathbf{n}}^2\,\text{tr}\left(\mathbf{I}_l\,\mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\mathbf{X}_l\,(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\right)}{\sigma_{\mathbf{n}}^2\,\text{tr}(\mathbf{I}_l)} \\
&= \frac{\text{tr}\left(\mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\mathbf{X}_l^T\right)}{l} = \frac{\text{tr}\left(\mathbf{X}_l^T\mathbf{X}_l(\mathbf{X}_l^T\mathbf{X}_l)^{-1}\right)}{l} \\
&= \frac{\text{tr}(\mathbf{I}_p)}{l} = \frac{p}{l}.
\end{aligned} \tag{10}$$

The expected value is taken with respect to $\mathbf{n}_l$. $\mathbf{C}_{\mathbf{nn}}$ denotes the covariance matrix of the noise vector. The noise is assumed to be uncorrelated and therefore $\mathbf{C}_{\mathbf{nn}} = \sigma_{\mathbf{n}}^2\mathbf{I}_l$. For $p = l$, (10) yields 1 as in this case denoising is not possible. The analytical result corresponds well to the simulated RMSE in Fig. 2. For example, according to (10) a doubling of the block size from $5p$ to $10p$ halves the remaining noise power. In fact, the simulation shows a decrease of the RMSE by a factor of ~ 1.4, which is approximately equal to an MSE reduction by a factor of 2. This confirms that the null space projection (preceded by a filtering operation) adapts close to the optimum solution in a low number of iterations. Consequently, the assumption made during the derivation of (10) is practically valid.

IV. Null Space Projection Enhancement Architecture

A. Architecture

Combining an LMS filter and the described null space projection enhancement leads to the development of an architecture schematically depicted in Fig. 3. For comparison, the main functional blocks are annotated by the line number of the corresponding operation in Alg. 2. The architecture needs 5 memories of length l: two for storing $\mathbf{x}_1$ and $\mathbf{x}_2$, two for storing $\mathbf{y}_1$ and $\mathbf{y}_2$ and one for storing $\mathbf{z}$ used in the null space projection enhancement. The memories for $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{z}$ are assumed to be dual-ported at the read output with ports A and B, respectively. The write ports are schematically depicted at the top of the memories in Fig. 3, while the read ports are depicted at the bottom. We will first describe the operation of the architecture assuming already filled memories with past input samples. In the next section, we will then describe the operation of the architecture after reset.

The inputs $x[k]$ and $y[k]$ of the adaptive filter are stored alternatingly in the corresponding memories with index 1 and 2, respectively. After storing l samples of $x[k]$ and $y[k]$ into the memories of, e.g. index 1, the storage of the inputs is switched to the memories with e.g. index 2 and vice versa. In the following, we will describe the operation on the example of writing new samples into the memories with index 1. For a more detailed description on the memory managment we refer to Sect. IV-B. The LMS filter consecutively reads samples from the memories $\mathbf{x}_1$ and $\mathbf{y}_1$. It performs its operation using two multipliers, one for the (sequentially calculated) inner product and one for the step size calculation. After read, the positions in $\mathbf{x}_1$ and $\mathbf{y}_1$ of the memories are overwritten by the new inputs $x[n]$ and $y[n]$, respectively. This proceeds until the end address l of the memories is reached. Then the memories are switched and the same procedure is performed on the second pair of memories. During the LMS operations and the input of the new samples, the null space projection enhancement is performed on the memories with index 2. The filter (1:) uses the previously estimated $\hat{\mathbf{w}}$ to filter the input values $\mathbf{x}_2$. These filter coefficients are transferred from the LMS part every time the memories are switched. The NPE filter's output (1:) samples are subtracted from $\mathbf{y}_2$ and the results are stored in $\mathbf{z}$.

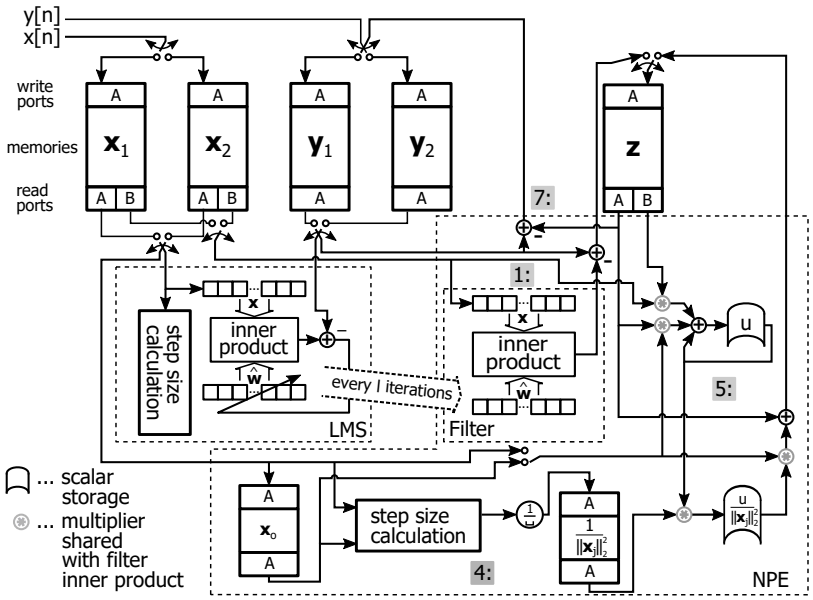

Fig. 3. Null space projection enhanced LMS architecture.

To finish the null space projection enhancement during the time span the LMS needs to process l samples, we occupy two multipliers for the inner product of the filtering operation. Meanwhile, we use one additional multiplier to perform the step size calculations for the null space projection. The p step sizes are stored in a small memory for the following null space projection. After finishing filtering and subtraction from the samples of $\mathbf{y}_2$, p null space projection iterations are performed on the elements of $\mathbf{z}$ using the memory $\mathbf{x}_2$ as well as the calculated step sizes (4:). The null space projection iterations share the two hardware multipiers that have been used for the filtering operation before. After the null space projection, the results stored in memory $\mathbf{z}$ are subtracted from the elements of memory $\mathbf{y}_2$ (7:), to obtain the desired noise reduction. To consider the overlap between the blocks (due to the convolution matrix structure of $\mathbf{X}$ used in the null space projection) we used the memory $\mathbf{x}_o$ for storing and accessing the corresponding values of $x[n]$.

B. Block Processing and Memory Management

To give a better overview on the processing of the input data and the allocation of the memories, we show the block processing and corresponding memory allocation in Fig. 4. We depict the scenario after a reset of the circuit. As one can see in this figure, due to the operation of the architecture described above, the input is divided into blocks of length l. To prevent a delay of this approach, the LMS algorithm starts by processing the first block without the NPE. After finishing block 1 the LMS proceeds with block 2. In parallel, the NPE processes block 1.

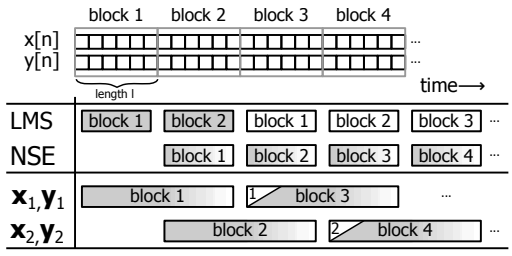

Fig. 4. Processing of blocks and memory allocation.

After the LMS finishes processing of block 2 and the NPE finishes processing of block 1, the LMS algorithm restarts processing of block 1, while the NPE continues with block 2. After this, both units process the following blocks in consecutive order. As depicted in Fig. 4, this leads to each memory holding a complete block twice the block duration in samples. In the third block duration, the noise-reduced samples (depicted by white areas in the blocks) are read out by the LMS block. The memory positions where the LMS just read from are used to store the new input samples $x[k]$ and $y[k]$.

These considerations allow to analyze the results in Fig. 2 in more detail. Due to the block-based nature of the NPE, the LMS can access the denoised input data only after a delay of $2l$ samples. The LMS then restarts with the denoised block 1. As a result, the NPE-LMS performance reported in Fig. 2 at iteration $4l$ (e. g. iteration 300 for $l = 5p$) was obtained with the input data of only two blocks. This means, if an application only allows sampling of, e.g. 3 blocks, but permits a delay afterwards, one could process the buffered block 3 after re-processing the noise-reduced blocks 1 and 2. In the example of $l = 5p$, this would allow to obtain the performance shown at iteration 375 ($5l$) with only 225 ($3l$) input samples.

C. Performance and Complexity Comparison

In the architecture described above, we used 2 multipliers in the LMS part and 3 multipliers in the NPE part. Counting the multiplications per iteration, one obtains $2p + 3$ for the LMS ($2p + 1$ for the LMS operation and 2 for the step size update). For NPE on a block of length l we require: pl multiplications for the filter, $l + 2(p - 1)$ for the step size calculation, and $p(2l + 1)$ for the null space projection. This means that in addition to the LMS, the NPE needs less than $3p+4$ (for meaningful values $l > p$) multiplications per input sample. Therefore the total complexity of the NPE-LMS is

TABLE I
SYNTHESIS RESULTS

Altera Cyclone V SX C6	NLMS	NPE-LMS
Logic Utilization (in ALMs)	$783/41,509$	$2,186/41,509$
Registers	$795/166,036$	$2195/166,036$
Embedded 18-bit Multipliers	$2/224$	$5/224$
Fmax Slow 1100mV 85C Model	112 MHz	102 MHz

slightly higher than that of two NLMS filters. It is important to note that the number of multiplications per sample *does not increase* with the buffer length l, only the required memory size and, potentially, the delay increases.

So far we did not consider the effort for memories and additional logic. For a more detailed complexity report, we present synthesis results ($p = 15$, $l = 5p$; for $l = 15p$ the utilization would only increase by 29 adaptive logic modules (ALMs) and 28 registers), comparing the NLMS as well as the NPE-LMS in an FPGA implementation in Tab. I.

Compared to the LMS implementation, the NPE-LMS requires about three times the number of logic elements and 3 additional multipliers, which is still a moderate resource utilization. In addition, the NPE-LMS approach requires 5 block RAMs for storage of the buffer variables. These results have been obtained using a bit width of 16 bits for representation of all numbers in the implementation. We compared the performance of these implementations for a system identification example using $p = 15$. The elements of $\mathbf{w}$ and $x[n]$ have been randomly selected from a uniform distribution in $[-1, 1]$. Gaussian noise was added at the output of the unknown system to obtain the simulated SNR. The measurements have been scaled to prevent fixed point overflows (leading to equally scaled estimation results). Fig. 5 shows RMSE results of bit-true fixed-point simulations for the NPE-LMS over multiple SNR values and bit widths. As reference, we include the performance of the NLMS filter as well as of the RLS filter, both simulated using double precision floating point. The former is used to depict the best performance obtainable by an NLMS filter implementation with practically unlimited number precision, while the latter is used as optimal reference performance. The simulation results lead to the conclusion that even with a moderate bit width of $B = 14$ and a block length of $l = 5p$ a significant performance improvement can be achieved. This gain is accomplished with a relatively moderate hardware overhead, especially when put into reference with the much higher complexity of the RLS ($O(p^2)$ multiplications per iteration). As an example, at an SNR of 15 dB, $B = 14$ and $l = 5p$ the proposed solution features a 6.9 dB lower RMSE than the NLMS without NPE.

V. CONCLUSION

We presented a method for improving LMS filters by using null space projection for noise reduction. A sophisticated buffering scheme was developed to enable a parallel operation of LMS and NPE. We presented a fixed-point hardware implementation of the proposed approach and demonstrated the achievable performance gains at varying bit widths by simulation. The complexity of our solution was analyzed in terms of required multiplications as well as by showing synthesis results. We showed that an estimation performance close to that of double precision can be achieved with an implementation in 14 bits fixed point.

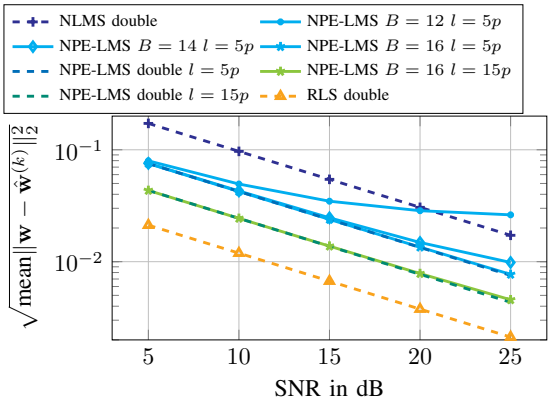

Fig. 5. RMSE of LMS, NPE-LMS and RLS over SNR.

REFERENCES

[1] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1997.
[2] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *1960 IRE WESCON Convention Record, Part 4*, Institute of Radio Engineers. New York: Institute of Radio Engineers, 8 1960, pp. 96–104. [Online]. Available: http://www-isl.stanford.edu/ widrow/papers/c1960adaptiveswitching.pdf
[3] S. Kaczmarz, "Przyblizone rozwiazywanie ukladów równan liniowych. – angenährte Auflösung von Systemen linearer Gleichungen," *Bull. Acad. Polon. Sci. Lett.*, vol. A35, pp. 355–357, 1937.
[4] C. Popa, "Extensions of block-projections methods with relaxation parameters to inconsistent and rank-deficient least-squares problems," *BIT Numerical Mathematics*, vol. 38, no. 1, pp. 151–176, Mar 1998. [Online]. Available: https://doi.org/10.1007/BF02510922
[5] S. Petra and C. Popa, "Single projection kaczmarz extended algorithms," *Numerical Algorithms*, vol. 73, no. 3, pp. 791–806, Nov 2016. [Online]. Available: https://doi.org/10.1007/s11075-016-0118-7
[6] M. Lunglmayr, C. Unterrieder, and M. Huemer, "Step-adaptive approximate least squares," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, Aug 2015, pp. 1108–1112.
[7] M. Lunglmayr and M. Huemer, "Parameter optimization for step-adaptive approximate least squares," in *Computer Aided Systems Theory – EUROCAST 2015*, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds. Cham: Springer International Publishing, 2015, pp. 521–528.
[8] S. Haykin, *Adaptive Filter Theory*. Prentice-Hall, 2002.
[9] T. Strohmer and R. Vershynin, "A randomized kaczmarz algorithm with exponential convergence," *Journal of Fourier Analysis and Applications*, vol. 15, no. 2, p. 262, Apr 2008. [Online]. Available: https://doi.org/10.1007/s00041-008-9030-4
[10] T. M. Whitney and R. K. Meany, "Two algorithms related to the method of steepest descent," *SIAM Journal on Numerical Analysis*, vol. 4, no. 1, pp. 109–118, 1967. [Online]. Available: http://www.jstor.org/stable/2949739
[11] A. Galántai, *Projectors and Projection Methods*. Springer US, 2004.
[12] A. Zouzias and N. Freris, "Randomized extended kaczmarz for solving least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 2, pp. 773–793, 2013. [Online]. Available: https://doi.org/10.1137/120889897
[13] C. Popa, "Characterization of the solutions set of inconsistent least-squares problems by an extended kaczmarz algorithm," *Korean Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 51–64, Jan 1999. [Online]. Available: https://doi.org/10.1007/BF02941906
[14] G. Strang, *Introduction to Linear Algebra*, 4th ed. Wellesley, MA: Wellesley-Cambridge Press, 2009.