APPROXIMATE LEAST SQUARES

Michael Lunglmayr[†] *, Christoph Unterrieder*[†]*, Mario Huemer*[⋆]

[†] Klagenfurt University, Embedded Systems and Signal Processing, 9020 Klagenfurt, Austria
[⋆] Johannes Kepler University Linz, Institute of Signal Processing, 4040 Linz, Austria
michael.lunglmayr@aau.at

ABSTRACT

We present a novel iterative algorithm for approximating the linear least squares solution with low complexity. After a motivation of the algorithm we discuss the algorithm's properties including its complexity, and we present theoretical results as well as simulation based performance results. We describe the analysis of its convergence behavior and show that in the noise free case the algorithm converges to the least squares solution.

Index Terms— least squares, approximation, iterative algorithm, complexity.

1. INTRODUCTION

The linear least squares (LS) approach is an important and extensively studied problem in many areas of signal processing with many practical applications from localization [1] to battery state estimation [2]. In applying the linear LS approach for a vector parameter $\mathbf{x}$, we assume a signal model $\mathbf{Hx}$ disturbed by noise $\mathbf{n}$ such that

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

where $\mathbf{H}$ is a known $m \times p$ observation matrix ($m \geq p$) with full rank p, $\mathbf{y}$ is a known $m \times 1$ vector (typically from measurements), $\mathbf{x}$ is an unknown $p \times 1$ parameter vector that is to be estimated and $\mathbf{n}$ is an $m \times 1$ noise vector. For the LS approach, the statistical properties of $\mathbf{n}$ need not to be known. For simplicity we only consider real vectors and matrices in this work, however, the presented concepts can easily be extended for complex vectors and matrices. The vector $\hat{\mathbf{x}}_{LS}$ that minimizes the cost function

$$J(\hat{\mathbf{x}}) = \sum_{i=1}^{m} (y_i - \mathbf{h}_i^T \hat{\mathbf{x}})^2 = (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}})^T (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}) \tag{2}$$

is the solution to the LS problem. Here $\mathbf{h}_i^T$ is the i^{th} row of $\mathbf{H}$ and y_i is the i^{th} element of $\mathbf{y}$, respectively. The LS solution is given by

$$\hat{\mathbf{x}}_{LS} = \mathbf{H}^{\dagger}\mathbf{y}, \tag{3}$$

with $\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ as the pseudoinverse of $\mathbf{H}$. Numerically more stable algorithms avoiding explicitly calculating $\mathbf{H}^{\dagger}$, e.g. based on the QR decomposition, can for example be found in [3]. A solution as in (3) is often called batch solution in literature [4].

For real time applications one usually wants to avoid the calculation of the batch solution due to its computational complexity and its large memory requirements. Alternatives are sequential algorithms such as the Sequential Least Squares (SLS) algorithm – described in the next section – or gradient based approaches such as the iterative LS (ILS) [3] algorithm. The latter algorithm is based on the steepest descent approach and iteratively calculates

$$\hat{\mathbf{x}}^{(k)} = \hat{\mathbf{x}}^{(k-1)} - \mu \nabla J(\hat{\mathbf{x}}^{(k-1)}), \tag{4}$$

for iteration k. Here $\nabla J(\hat{\mathbf{x}}^{(k-1)}) = -2\mathbf{H}^T\mathbf{y} + 2\mathbf{H}^T\mathbf{H}\hat{\mathbf{x}}^{(k-1)}$ is the gradient of $J(\hat{\mathbf{x}})$ at $\hat{\mathbf{x}}^{(k-1)}$. For $k \to \infty$, $\hat{\mathbf{x}}^{(k)}$ converges to $\hat{\mathbf{x}}_{LS}$ given that the iteration step width μ fulfills $0 < \mu < 1/(2s_1^2(\mathbf{H}))$ [3], with $s_1(\mathbf{H})$ as the largest singular value of $\mathbf{H}$. Alternatively, (4) can be written as

$$\hat{\mathbf{x}}^{(k)} = \hat{\mathbf{x}}^{(k-1)} + \mu \sum_{i=1}^{m} 2\mathbf{h}_i (y_i - \mathbf{h}_i^T \hat{\mathbf{x}}^{(k-1)}). \tag{5}$$

Analyzing the complexity of this approach one can see that $2pm+p$ multiplications are required per iteration. In addition, every iteration of ILS requires the availability of all elements of the measurement vector $\mathbf{y}$.

Based on the principle of ILS we propose a novel iterative way of approximating the least squares solution that we call approximate least squares (ALS). As we will show, the complexity of this approach is significantly lower than for ILS and it requires only one measurement value y_i per iteration.

When analyzing (5), the gradient can be interpreted as a sum of the partial gradients

$$d_i(\hat{\mathbf{x}}^{(k-1)}) = -2\mathbf{h}_i (y_i - \mathbf{h}_i^T \hat{\mathbf{x}}^{(k-1)}) \tag{6}$$

as schematically depicted in Fig. 1. The idea of ALS is to use only *one* of these partial gradients per iteration. Instead of moving a small step (due to μ) in a steepest descent way in the negative direction of the gradient as done by ILS, ALS moves a small step in the negative direction of only a partial gradient. This has the advantage of a lower complexity, but – as we will

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

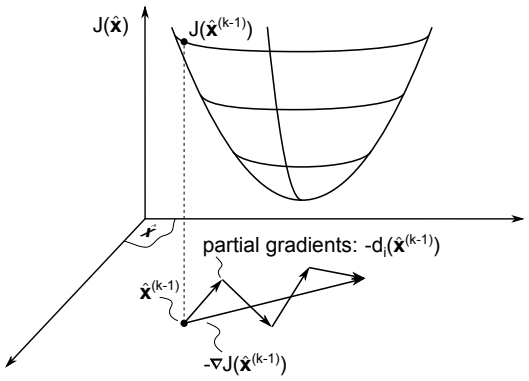

Fig. 1. Gradient and partial gradients of ILS.

discuss below – also has the disadvantage of a higher noise sensitivity. Following this general idea, two issues have to be addressed. First, the number of iterations of the algorithm to achieve satisfying performance results may be higher than the number of rows of $\mathbf{H}$. Second, the noise sensitivity has to be reduced. To cope with the first issue we suggest to reuse the rows $\mathbf{h}_i^T$ of $\mathbf{H}$ in a cyclic manner. Let the operator " $\urcorner$" be defined such that for a positive natural number i: $i^\urcorner = ((i-1) \bmod \mathrm{m}) + 1$. From this it follows that $i^\urcorner \in \{1, \ldots, m\}$. For better readability we do not write the dependence of this operator on m in the operator's symbol. For ALS, m is always the number of rows of the matrix $\mathbf{H}$. An ALS iteration is now defined as

$$\hat{\mathbf{x}}^{(k)} = \hat{\mathbf{x}}^{(k-1)} + \mu 2 \mathbf{h}_{k^\urcorner} (y_{k^\urcorner} - \mathbf{h}_{k^\urcorner}^T \hat{\mathbf{x}}^{(k-1)}). \tag{7}$$

That means for ALS that if k reaches m, for the following iterations the first rows of $\mathbf{H}$ and the first elements of $\mathbf{y}$ are used again in a cyclic manner. We will now address the second issue, namely the noise sensitivity. As we will discuss below, if $\mathbf{n} = \mathbf{0}$ then $\hat{\mathbf{x}}^{(k)}$ converges to $\hat{\mathbf{x}}_{LS}$ as $k \to \infty$. For the usual case $\mathbf{n} \neq \mathbf{0}$, a noise dependent error remains. This error can be greatly reduced by introducing a simple averaging process in the last m iterations. A formal justification for this averaging will be given within the error analysis in Sect. 3. Summarizing, this leads to an overall formulation of the algorithm:

Algorithm: ALS

$\hat{\mathbf{x}}_{ALS} = \mathbf{0}$
$\hat{\mathbf{x}}^{(0)} = \mathbf{0}$
for $k = 1 \ldots N$ **do**
  $\hat{\mathbf{x}}^{(k)} = \hat{\mathbf{x}}^{(k-1)} + \mu 2 \mathbf{h}_{k^\urcorner} (y_{k^\urcorner} - \mathbf{h}_{k^\urcorner}^T \hat{\mathbf{x}}^{(k-1)})$
  if $k > N - m$ **then**
    $\hat{\mathbf{x}}_{ALS} = \hat{\mathbf{x}}_{ALS} + \hat{\mathbf{x}}^{(k)}$
  end if
end for
$\hat{\mathbf{x}}_{ALS} = \frac{1}{m} \hat{\mathbf{x}}_{ALS}$

Here N denotes the number of iterations of the algorithm and $\hat{\mathbf{x}}_{ALS}$ is the approximation of $\hat{\mathbf{x}}_{LS}$ that is output by the algorithm. When analyzing the above algorithm, and only counting the multiplications, one can see that $(2p+1)N$ overall multiplications are required to perform the algorithm. Compared to ILS a factor of around m fewer multiplications per iteration are required. Although more iterations are usually needed for ALS its overall complexity is significantly lower as will be demonstrated in Sect. 5. This decrease in complexity is bought with only a small degradation in performance. An additional advantage of ALS is that per iteration only one value y_i and only one row $\mathbf{h}_i^T$ of $\mathbf{H}$ are required. This significantly reduces the required number of other operations (additions, memory accesses,...) and also simplifies the memory management as well as the architecture when thinking of a hardware implementation.

2. RELATION TO PRIOR WORK

ALS not only has similarities to ILS but also to the SLS approach [4]. For SLS the update equation

$$\hat{\mathbf{x}}^{(k)} = \hat{\mathbf{x}}^{(k-1)} + \mathbf{K}_k (y_k - \mathbf{h}_k^T \hat{\mathbf{x}}^{(k-1)}) \tag{8}$$

is sequentially calculated m times, requiring an update of the gain vector $\mathbf{K}_k$ at every iteration. Although, the algorithm can deliver $\hat{\mathbf{x}}_{LS}$ after m iterations, the update of $\mathbf{K}_k$ requires significant effort, including the multiplication of full matrices (although symmetry can be exploited to reduce the complexity). ALS uses the same update equation (usually more than m times), with the simplified choice $\mathbf{K}_k = 2\mu \mathbf{h}_{k^\urcorner}$.

Update equation (8) is arithmetically similar to the Least Mean Squares (LMS) filter update step [5]. However, the LMS update step uses a random (filter input) vector and one sample of a desired signal as input, whereas the ALS update step only uses one sample y_i of the measurement vector $\mathbf{y}$ as input. Also the original formulation of the LMS algorithm for the so-called ADALINE [6, 7] approach was based on a random input vector, providing an adaptive approach with a potentially unlimited set of input patterns. Instead of the random input vector in the LMS case the deterministic and fixed rows of the observation matrix H are used in the update equation of the ALS. The row vectors $\mathbf{h}_i^T$ and the measurement values y_i are cyclically re-used. Another difference is the averaging at the last m iterations which is unique for the ALS algorithm. And finally, the convergence behavior of ALS can be described in a completely deterministic manner, whereas the convergence of the LMS is usually only described in the mean. Anyhow, the authors are confident that some ideas improving LMS – e.g. adjusting the step size [8, 9] – might be also used to further improve the performance of ALS.

3. CONVERGENCE BEHAVIOR

By rewriting (7) as

$$\hat{\mathbf{x}}^{(k)} = (\mathbf{I} - 2\mu\mathbf{h}_{k'}\mathbf{h}_{k'}^T)\hat{\mathbf{x}}^{(k-1)} + 2\mu\mathbf{h}_{k'}y_{k'} \tag{9}$$

and defining the error vector of ALS $\mathbf{e}^{(k)} = \hat{\mathbf{x}}^{(k)} - \mathbf{x}$ together with $\mathbf{M}_{k'} = (\mathbf{I} - 2\mu\mathbf{h}_{k'}\mathbf{h}_{k'}^T)$ one gets

$$\mathbf{x}^{(k)} = (\mathbf{I} - 2\mu\mathbf{h}_{k'}\mathbf{h}_{k'}^T)(\mathbf{x} + \mathbf{e}^{(k-1)}) + 2\mu\mathbf{h}_{k'}y_{k'} \tag{10}$$

$$= \mathbf{x} - 2\mu\mathbf{h}_{k'}\mathbf{h}_{k'}^T\mathbf{x} + \mathbf{M}_{k'}\mathbf{e}^{(k-1)} + 2\mu\mathbf{h}_{k'}(\mathbf{h}_{k'}^T\mathbf{x} + n_{k'}). \tag{11}$$

Here $n_{k'}$ is the k'^{th} element of $\mathbf{n}$. Subtracting $\mathbf{x}$ left and right from the equation leads to

$$\mathbf{e}^{(k)} = \mathbf{M}_{k'}\mathbf{e}^{(k-1)} + 2\mu\mathbf{h}_{k'}n_{k'}. \tag{12}$$

When defining $\mathbf{\Delta}_{k'} = 2\mu\mathbf{h}_{k'}n_{k'}$ one can write the above equation as

$$\mathbf{e}^{(k)} = \prod_{i=1}^{k}\mathbf{M}_{i'}\mathbf{e}^{(0)} + \\ + \mathbf{\Delta}_{k'} + \mathbf{M}_{(k-1)'}(\mathbf{\Delta}_{(k-2)'} + \ldots + (\mathbf{M}_2\mathbf{\Delta}_1)\ldots), \tag{13}$$

with $\mathbf{e}^{(0)}$ as the initial error. Here the product of the matrices is defined as $\prod_{i=1}^{k}\mathbf{M}_{i'} = \mathbf{M}_{k'}\mathbf{M}_{(k-1)'}\ldots\mathbf{M}_1$. When analyzing the above equation one can see that the error at iteration k depends on the initial error $\mathbf{e}^{(0)}$ represented in $\mathbf{e}^{(k)}$ by the part $\mathbf{e}_0^{(k)} = \prod_{i=1}^{k}\mathbf{M}_{i'}\mathbf{e}^{(0)}$ as well as on an error term introduced by noise represented by $\mathbf{e}_\Delta^{(k)} = \mathbf{\Delta}_{k'} + \mathbf{M}_{(k-1)'}(\mathbf{\Delta}_{(k-2)'} + \ldots + (\mathbf{M}_2\mathbf{\Delta}_1)\ldots)$. With this one can write

$$\mathbf{e}^{(k)} = \mathbf{e}_0^{(k)} + \mathbf{e}_\Delta^{(k)}. \tag{14}$$

If no noise is present then

$$\mathbf{e}^{(k)} = \mathbf{e}_0^{(k)} = \prod_{i=1}^{k}\mathbf{M}_{i'}\mathbf{e}^{(0)}. \tag{15}$$

When choosing k as an integer multiple of m and defining $\mathbf{M} = \prod_{i=1}^{m}\mathbf{M}_i$ one obtains

$$\mathbf{e}^{(k)} = \prod_{i=1}^{k}\mathbf{M}_{i'}\mathbf{e}^{(0)} = \mathbf{M}^{\frac{k}{m}}\mathbf{e}^{(0)}. \tag{16}$$

In [10] we show that for the choice

$$0 < \mu \leq \frac{1}{2\max\limits_{i=1\ldots m}\|\mathbf{h}_i^T\|_2^2}, \tag{17}$$

the matrix $\mathbf{M}$ has a 2-norm smaller than one (although the proof is not complicated it is omitted here due to length constraints). This implies that all eigenvalues of $\mathbf{M}$ have absolute values smaller than one. From this it directly follows that $\mathbf{e}_0^{(k)}$ converges to zero as $k \to \infty$, i.e. $\hat{\mathbf{x}}_{ALS} = \hat{\mathbf{x}}_{LS} = \mathbf{x}$. This means that if no noise is present $\hat{\mathbf{x}}^{(k)}$ converges to $\mathbf{x}$. However if $\mathbf{n} \neq \mathbf{0}$ a persistent error $\mathbf{e}_\Delta^{(k)}$ remains. In [10] we will give a more detailed analysis of $\mathbf{e}_\Delta^{(k)}$, showing that $\mathbf{e}^{(k)}$ features almost a periodic behavior from an index k_p on, wherefrom $\mathbf{e}_0^{(k)}$ can be considered negligible. This particular index k_p, which can also be specified analytically, can be used to define N e.g. as $N = k_p + m$.

By analyzing the ALS algorithm one can see the importance of the averaging in the final m iterations. As we already noted $\mathbf{e}^{(k)}$ is highly dependent on the noise for large k ($\mathbf{e}_0^{(k)}$ vanishes with increasing k). The averaging over the last $\hat{\mathbf{x}}^{(k)}$ vectors yields

$$\mathbf{e}_{ALS} = \hat{\mathbf{x}}_{ALS} - \mathbf{x} = \left(\frac{1}{m}\sum_{k=N-m+1}^{N}\hat{\mathbf{x}}^{(k)}\right) - \mathbf{x} \tag{18}$$

$$= \frac{1}{m}\left(\sum_{k=N-m+1}^{N}\hat{\mathbf{x}}^{(k)} - m\mathbf{x}\right) \tag{19}$$

$$= \frac{1}{m}\sum_{k=N-m+1}^{N}\left(\hat{\mathbf{x}}^{(k)} - \mathbf{x}\right) \tag{20}$$

$$= \frac{1}{m}\sum_{k=N-m+1}^{N}\mathbf{e}^{(k)}. \tag{21}$$

That means by averaging over the last m vectors $\hat{\mathbf{x}}^{(k)}$ an averaging over the corresponding error vectors occurs. Since for a practical application it is highly unlikely that all these error vectors have equal length and point in the same direction (in this case averaging would have no effect) this averaging step typically significantly reduces the error norm. The averaging only has to be done *once*, it therefore presents only a minor complexity increase (overall only pm additions and p multiplications with the constant $1/m$).

4. SIMULATION RESULTS

We first show simulation results of a typical example of least squares estimation: the estimation of amplitudes of sine signals in noise. For this demonstration example we chose $\mathbf{H}$ as a 100×8 matrix with elements $H_{n,k} = \cos(2\pi n T_s f_k)$. The frequencies f_k are not necessarily integer multiples of a base frequency. The elements of the noise vector have been sampled indepentently from a normal distribution with zero mean and a standard deviation $\sigma = 10^{-2}$. The amplitudes $\mathbf{x}$ have been estimated using 100 values, forming the vector $\mathbf{y}$. The step size for ILS was chosen as $\mu = 1/(2\,s_1^2(\mathbf{H}))$ and for ALS as $\mu = 1/(2\max_{i=1\ldots m}\|\mathbf{h}_i^T\|_2^2)$. The estimation performance has been measured by calculating the norm of the difference vector between the true vector $\mathbf{x}$ and the estimated vectors, respectively. Fig. 2 shows a typical simulation result for ILS and ALS. In this figure, $\hat{\mathbf{x}}_{ALS}$ is the estimated parameter vector resulting after averaging the final m out of N

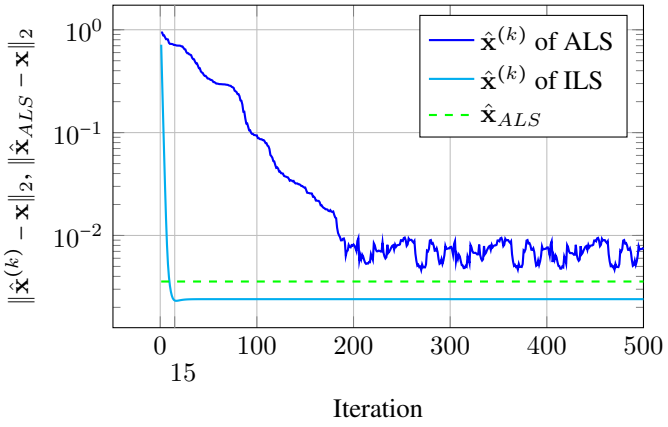

Fig. 2. Example of errors of ILS and ALS

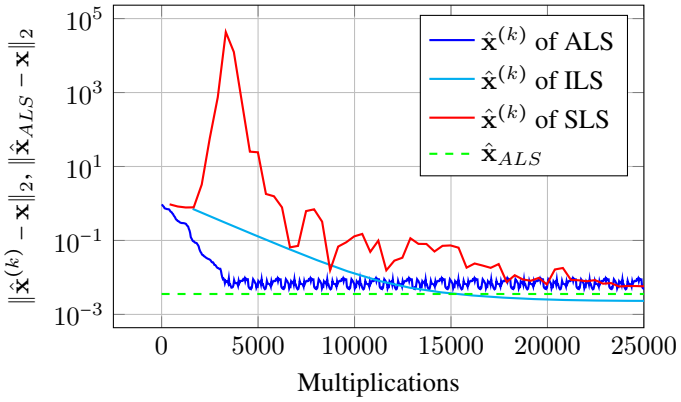

Fig. 3. Error norms over to the number of multiplications

vectors $\hat{\mathbf{x}}^{(k)}$, represented as a horizontal line for illustration purposes. As one can see, ILS requires significantly less iterations than ALS, but with about $m = 100$ times more multiplications per iteration. The performance of ALS is only slightly worse than the performance of ILS but ALS features a significantly lower overall complexity. In this figure one can observe an interesting behavior of ALS. After a certain number of iterations the influence of $\mathbf{e}_0^{(k)}$ becomes negligible. This reflects in an oscillatory behavior of the error norm as can be seen in Fig. 2. This oscillatory behavior comes from the fact that the values y_i are cyclically re-used in the N ALS iterations. As a consequence also the noise values appear in a cyclic manner. The averaging at the end of ALS is most effective if N is chosen large enough so that the effects of $\mathbf{e}_0^{(k)}$ are negligible. Such a value for N can be found with simulations or based on analytical results as will be presented in [10]. To provide a fair comparison, in Fig. 3 we compared ALS, ILS and SLS in terms of its error norms over the number of calculated multiplications. As one can see, if the error performance of ALS is sufficient for a given application, its complexity is significantly lower. In this example ILS needs about 3 times more multiplications than ALS to obtain the same error norm. But as stated above, this complexity analysis is only based on the number of multiplications per iteration. Including other operations (additions, memory accesses) would furthermore favor ALS. Due to page constraints we omitted a more detailed complexity analysis in this paper. If the error performance of ALS is not sufficient for a given application one could choose a different approach, but extended variants of the ALS, e.g. with adjusting μ during the iterations show promising first results towards further reducing the error norm. One can immediately see the benefits of such an approach in (12) because the noise dependent part of the error vector scales with μ, as will be described in detail in [10]. But as extensive performance simulations showed, ALS' performance is on average very close to the LS solution. Table. 1 shows performance results for random $\mathbf{H}$ matrices. The entries of these matrices have been sampled from a uniform distribution out of $[0, 1]$. Every simulation has been done for white Gaussian noise with $\sigma \in S = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$, respectively, with 100 random matrices $\mathbf{H}$ per σ value and 100 random vectors $\mathbf{x}$ (with random entries also sampled from a uniform distribution out of $[0, 1]$) per $\mathbf{H}$ matrix. For every σ value the averages $\overline{\|\hat{\mathbf{x}}_{ALS} - \mathbf{x}\|_2}$ and $\overline{\|\hat{\mathbf{x}}_{LS} - \mathbf{x}\|_2}$ over the simulated results have been calculated. Table. 1 shows the maximum relative increase of ALS' averaged error norm over the averaged error norms of LS, whereas the maximization has been done over the elements of S: $r_{max} = \max_S \left(\frac{\overline{\|\hat{\mathbf{x}}_{ALS} - \mathbf{x}\|_2}}{\overline{\|\hat{\mathbf{x}}_{LS} - \mathbf{x}\|_2}} - 1 \right)$. We furthermore want to note that the relative increase of the averaged error norms remained nearly constant over all simulated σ values. As one can see in this table, the performance of ALS shows on average only a minor degradation compared to the LS solution.

$\dim(\mathbf{H})$	r_{max}	$\dim(\mathbf{H})$	r_{max}
100×1	9.3%	1000×1	9.5%
100×2	9.7%	1000×2	9.5%
100×3	11.2%	1000×3	10.7%
100×5	11.3%	1000×5	12.8%
100×10	16%	1000×10	15.3%

Table 1. Performance results for random matrices.

5. CONCLUSION

We presented a novel algorithm for approximating the solution of the linear least squares problem. We discussed its convergence behavior and demonstrated that the algorithm provides a close solution to the least squares solution with low complexity. The presented algorithm shows promising potential for further extension in theory and implementation as well for use in a variety of applications.

6. REFERENCES

[1] Choi, K.H., Ra, W.-S., Park, S.-Y.; Park, J.B., "Robust Least Squares Approach to Passive Target Localization Using Ultrasonic Receiver Array," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 4, pp. 1993-2002, Apr. 2014.

[2] Unterrieder, C., Lunglmayr, M., Marsili, S., Huemer, M., "Battery state-of-charge estimation using polynomial enhanced prediction," *IET Electronics Letters*, vol. 48, no. 21, pp. 1363-1365, Oct. 2012.

[3] Björck A., *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[4] Kay S. M., *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, 2005.

[5] Widrow B., Glover J. R., McCool J. M., Kaunitz J., Williams C. S., Heam R. H., Zeidler J. R., Dong E., Goodlin R. C., "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, pp. 1692-1716, Dec. 1975.

[6] Widrow, B., Hoff M. E., "Adaptive Switching Circuits," *IRE WESCON Convention Record*, Part 4, pp. 96-104, 1960.

[7] Widrow, B., "Thinking about thinking: the discovery of the LMS algorithm," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 100-106, Jan. 2005.

[8] Harris R., Chabries D., Bishop P., "A variable step (VS) adaptive filter algorithm," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-34, pp. 309-316, Apr. 1986.

[9] Bhotto, M.Z.A., Antoniou, A., "A Family of Shrinkage Adaptive-Filtering Algorithms," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1689-1697, Apr. 2013.

[10] M. Lunglmayr, C. Unterrieder, M. Huemer, "Approximate Least Squares: Convergence and Performance Analysis," in preparation.