# Hardware-Implemented Adaptive Calibration Algorithms for Antenna Arrays

Andreas Winterstein
German Aerospace Center (DLR)
D-82234 Oberpfaffenhofen, Germany
Email: andreas.winterstein@dlr.de

Achim Dreher
German Aerospace Center (DLR)
D-82234 Oberpfaffenhofen, Germany
Email: achim.dreher@dlr.de

***Abstract*—An adaptive calibration technique based on the least mean squares (LMS) algorithm is presented which does not rely on integrated calibration networks and is therefore suitable for e.g. array-fed reflector (AFR) antenna systems. Using hardware-implemented algorithms, makes the calibration real-time capable and independent of external computation.**

**Possibilities to enhance the LMS by step-size controlling techniques are introduced and compared by simulation, regarding their adaption speed. The work focuses on the transition of the presented algorithms towards digital hardware, especially field-programmable gate arrays (FPGAs). The implementation feasibility of the proposed techniques is assessed and timing considerations as well as resource-effective realization are discussed. Suitable algorithms are selected and their implementation on an FPGA is described. Differences to simulation are outlined and optimization through pipelining is explained.**

**The performance of the realized algorithms is evaluated on a Ka-band receiver system. Figures of merit are the adaption speed and the resulting signal-to-noise ratio (SNR) improvement. Step-size control offers slight performance improvements over basic LMS with fixed step-sizes. This comes at the expense of higher resource demand and implementation effort.**

## I. Introduction

In this paper, the calibration of antenna arrays in receiving mode is considered where no built-in calibration network can be used. This is the case for e.g. horn antennas or array-fed reflector (AFR) systems. The work focuses on implementations in digital hardware which allow to calibrate in real-time using the received signals. This makes calibration independent of external and/or offline signal processing. Especially for systems which need frequent recalibration, this is advantageous.

Starting from the well-known LMS algorithm, deterministic and adaptive algorithms to control its step-size parameter are compared by simulation. For suitable methods, a transition from simulation to hardware is described by breaking down the algorithms into basic building blocks like adders and multipliers. The resulting implementation concepts are realized on a Xilinx® Virtex-4™ FPGA, showing possible optimizations. The realizations are then tested in a demonstrator system and the results are compared to the simulation. Finally, further possible improvements and research are discussed.

## II. Calibration and step-size control methods

The calibration shall equalize any phase differences between the signals of $K$ antenna elements. This is achieved by essentially doing adaptive beamforming. Therefore, it is necessary to have a signal from a known direction incident on the array. Adaption is based on the LMS algorithm described in [1]. Fig. 1 shows a block diagram of the calibration unit containing an antenna, an analog-to-digital converter (ADC) and an adaptive filter structure. The dotted connection indicates the probable necessity of analog front-ends to mix the signal down before the ADC. The $i$-th filter input at discrete time $n$, is the signal vector $\mathbf{x}_i(n)$ of length $L$ which is connected to the finite impulse response (FIR) filter $\mathbf{h}_i(n)$ whose $L$ coefficients are adapted by the Update block. For this the input signal vector $\mathbf{x}_i(n)$, a scalar error signal $e_i(n)$, and the step-size parameter $\mu$ are necessary. To create $e_i(n)$, a reference signal is needed. In this work, it is assumed that the output signal of the 1st array element, $y_1(n)$, serves as reference. The filter update equation is as follows

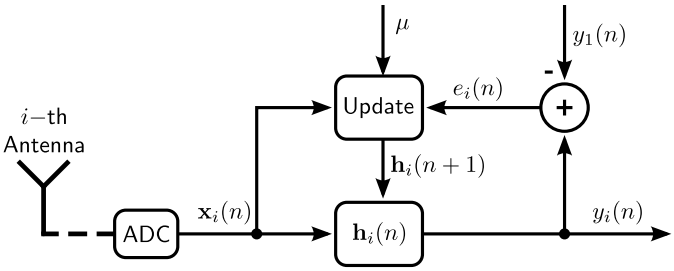


Fig. 1. Block diagram of the calibration unit. $i \in 2, ..., K$. The 1st channel is used as reference.

$$\mathbf{h}_i(n+1) = \mathbf{h}_i(n) + \mu e_i(n)\mathbf{x}_i^*(n) \tag{1}$$

whereby $\{\cdot\}^*$ denotes the complex conjugate. In the standard approach, $\mu$ has a fixed value whose influence is shown by simulation. In Fig. 2a, two curves show the adaption of a filter coefficient for both, a large and a small value of $\mu$. It can be seen that there is a trade-off: For a high step-size, the magnitude of the coefficient converges rapidly to a final value but then, oscillation around this optimum occurs. Moreover, large step-size values carry the risk of divergence of the whole adaption process. With a low step-size on the other hand, convergence is far slower, but the final state is held more steadily. Because of this, it is obvious that the use of a variable step-size $\mu(n)$ is favorable, which should be high when the optimum value is far away but diminishes as the coefficients

approach the optimum.

In the following, several step-size controlling techniques are presented. The evolution of the filter coefficient magnitudes is shown in Fig. 2b. Since the curves are very steep in the beginning, a detailed view of the initial development of the magnitudes for all methods is depicted in Fig. 2c. Finally, the step-size values over time are plotted in Fig. 2d.

The first and easiest possibility to achieve a diminishing $\mu$, is the step-wise reduction algorithm (SRA) which can be described by the formula

$$\mu_{\mathsf{sra}}(n+1) = \begin{cases} \mu_{\mathsf{sra}}(n)/D & \text{for } \mathsf{mod}(n, M) = 0 \\ \mu_{\mathsf{sra}}(n) & \text{else} \end{cases} \tag{2}$$

It is assumed that the adaption starts at $n = 1$, whereby $D > 1$, $D \in \mathbb{R}$ is the reduction factor and $M$ defines the number of samples during which the step-size value is hold. Since the algorithm can only reduce the step-size, it makes sense to introduce a lower limit. In the magnitude plots of Fig. 2, it can be seen that the performance improves over fixed step-size parameters: The initial convergence is as fast as for the high step-size value while in the end the optimum is held without fluctuations. Due to the simplicity of the method, the implementation effort is low. Despite good results, the method has a considerable drawback, namely that there must be some knowledge about the convergence speed, in order to decide on the parameter $M$. Also divergence can occur if the initial step-size is chosen too large.

A more sophisticated algorithm is the one proposed by Kwong [2], given by

$$\mu_{\mathsf{Kw}}(n+1) = \alpha\mu_{\mathsf{Kw}}(n) + \gamma\|e(n)\|^2 \tag{3}$$

with $\alpha < 1$, an automatic step-size decrease is introduced. Choosing $\gamma$ appropriately, this degradation is annihilated when the errors become large. By balancing the two parameters, the desired step-size behavior can be achieved. From the plot in Fig. 2c, it can be seen that initial convergence is slower than for SRA or large fixed step-size values. However, it is considerably faster than for low step-size and the final optimum is held steadily (Fig. 2b). The advantage of this algorithm over SRA is, that it is truly adaptive, i.e. the step-size will react to unexpected changes. A difficulty is the choice of the parameters $\alpha$ and $\gamma$. Appropriate values have to be found by experiment and the algorithm reacts very sensitively to inappropriate choices. Additionally, the method demands for floating-point arithmetics which can be problematic in FPGA design.

Another step-size control technique is the gradient sign changes algorithm (GSA) [3], [4]. Assuming that $\mathbf{h}_i(n)$ are filters of length $L$, the algorithm uses an individual step-size parameter for each filter tap. This leads to a step-size vector $\boldsymbol{\mu}_{\mathsf{GSA}}(n) = [\mu_{\mathsf{GSA},1}(n), ..., \mu_{\mathsf{GSA},L}(n)]^T$ for each filter. The magnitude changes of each coefficient are monitored. If it is monotonic increasing or decreasing over a certain number of samples, it is assumed that the coefficient is far from optimum. Therefore, the respective step-size is made larger in order to speed up convergence. If the magnitude changes are not monotonic, the coefficient fluctuates around an optimum. The step-size is then reduced to reach the final value more accurately. Each element $\mu_{\mathsf{GSA},j}(n), j \in 1, ..., L$ of the step-size vector $\boldsymbol{\mu}(n)$ is updated according to

$$\mu_{\mathsf{GSA},j}(n+1) = \begin{cases} \mu_{\mathsf{GSA},j}(n)/D & \text{for } c_{\mathsf{fluct}} = M_{\mathsf{fluct}} \\ \mu_{\mathsf{GSA},j}(n) \cdot D & \text{for } c_{\mathsf{mon}} = M_{\mathsf{mon}} \\ \mu_{\mathsf{GSA},j}(n) & \text{else} \end{cases} \tag{4}$$

whereby $c_{\mathsf{mon}}$ and $c_{\mathsf{fluct}}$ are counters for adjacent samples where magnitude is monotonic or fluctuating, respectively. $M_{\mathsf{mon}}$ and $M_{\mathsf{fluct}}$ determine the number of samples after which a step-size adaption is performed. From the curves in Fig. 2 it can be seen that convergence is not as fast as that of SRA but faster than the algorithm presented by Kwong. An advantage of the GSA is that it is adaptive. Also the parameter choice is not as critical since only integer values are possible. The implementation effort, on the other hand, is considerably higher compared to that of the SRA.

Regarding the beginning of the adaption process in Fig. 2c, it is clear that the SRA yields the fastest convergence. It suggests that the hold time for the step-sizes can even be shortened. GSA is slightly slower, but excels the technique presented in [2]. The final values are held steadily by all three algorithms. For the hardware implementation of the calibration unit, SRA and GSA are selected since they show the best performance while the implementation effort seems manageable. These approaches will be discussed in the following.

## III. Implementation concept and hardware synthesis

For the implementation of the calibration unit, two assumptions are made: First, single tap filters are employed, i.e. $\mathbf{h}_i(n) \Rightarrow h_i(n)$. Second, the step-size multiplication is realized as a bit shifter which introduces the constraint $\mu = 2^b, b \in \mathbb{Z}$. This simplifies the implementation considerably, but the algorithm performance is still sufficient. Under these premises, the adaptive calibration system shown in Fig. 1 can be converted into the one in Fig. 3. The different parts are now discrete components. Each connection has a certain word width $W$ (number of parallel bit lines). In contrast to simulations, where all components work without delays, each block introduces an individual time delay by $N$ samples. During simulation, all signals are available without delay. That means, the whole filter and update procedure is completed before the next input is processed and the update rate equals the data rate of the whole system. When implementing the system in hardware this is no longer the case. There are two crucial points in the system depicted in Fig. 3: First, the input signal has to be delayed such that it is multiplied with the according error signal, i.e. $N_{\mathsf{delay}} = N_{\mathsf{h}} + N_{\mathsf{add}}$. Second, the filter coefficients must be held constant until the update cycle is completed. This period is defined by the length of the critical path $N_{\mathsf{crit}} = N_{\mathsf{delay}} + N_{\mathsf{xe}} + N_{\mathsf{shift}} + N_{\mathsf{innov}}$. Therefore, the update rate in the realization is $N_{\mathsf{crit}}$ times lower than the data rate. These delays seem to be a disadvantage because adaption

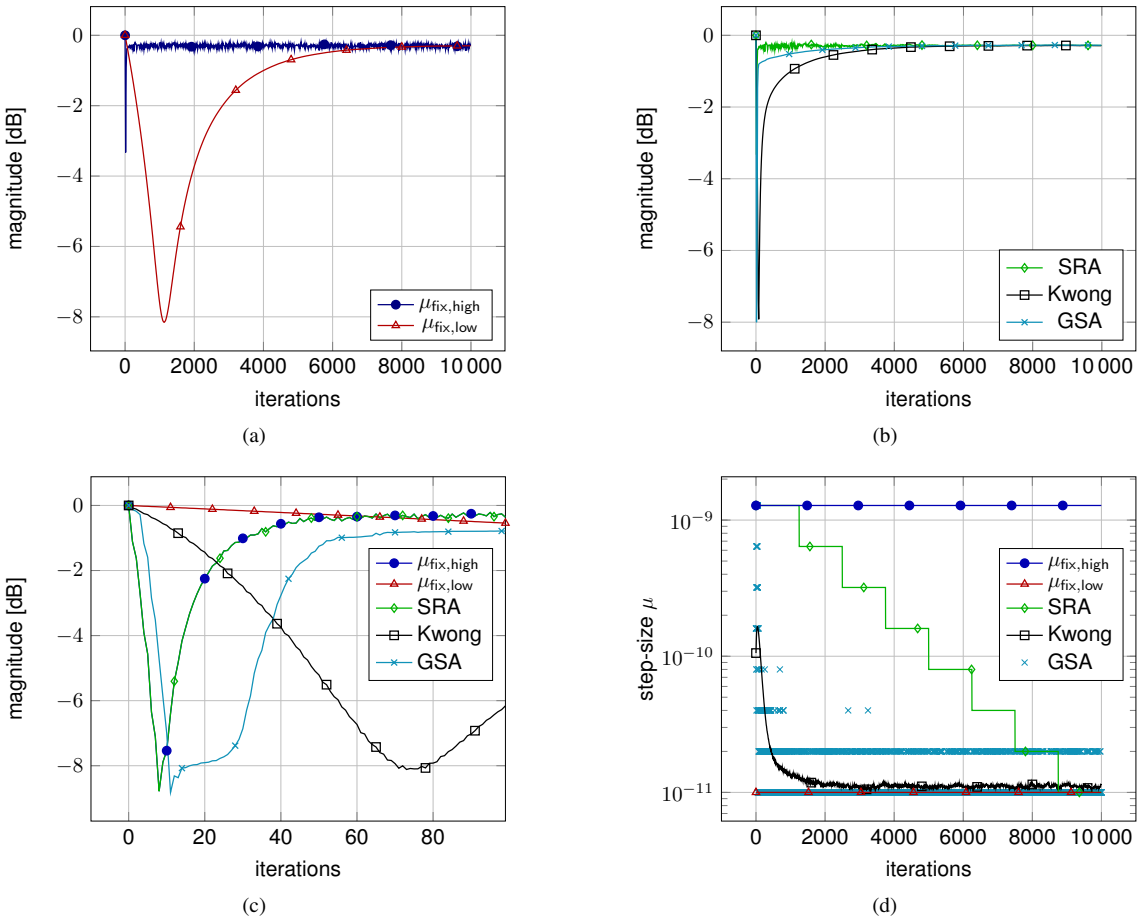


Fig. 2. Simulated evolution of the filter coefficient magnitude for different step-size concepts. Phase plots are not shown for space reasons but phase adapts faster in general. The upper row shows the adaption process for (a) high and low fixed step-size values and (b) for variable step-size techniques. (c) shows a detailed plot of the initial magnitude development for all techniques. Evolution of the step-size itself over the whole time is shown in (d).

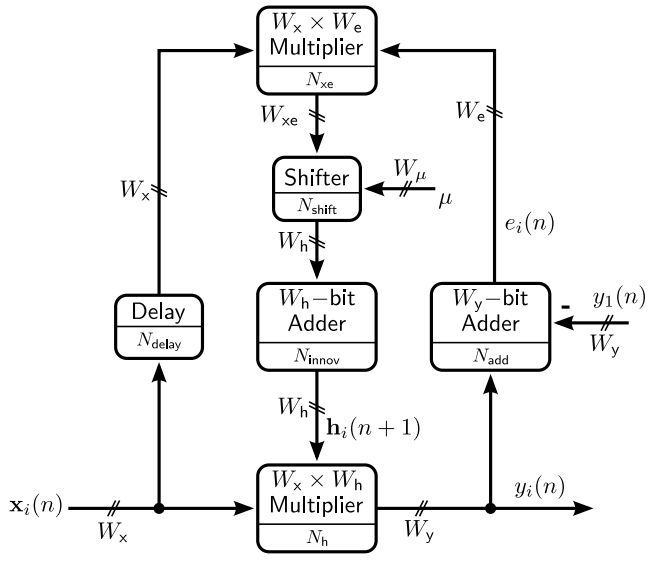


Fig. 3. Implementation concept of the calibration unit. Component delays and word widths are taken into account.

needs more time. However, it can be exploited by serializing processes which normally run in parallel as shown later on.

Since it decides the costs in terms of resource demand, the implementation of the different step-size techniques is of special interest. For SRA this is straight forward, employing an iteration counter which causes an arithmetic shift of $\mu$ on overflow. Due to the more complex update process in (4), things are more complicated for GSA. First, it is necessary to compare the magnitudes of the last filter coefficient $h_i(n-N_\text{crit})$ and the actual one. Since it is insignificant for the comparison, the squared magnitudes will be used. This avoids taking the square root which is a non-trivial task to realize in digital hardware. According to the comparison result, one of the counters $c_{i,\text{mon}}, c_{i,\text{fluct}}$ is altered while the other is set to zero.

$$\|h_i(n-N_\text{crit})\|^2 < \|h_i(n)\|^2 \begin{cases} c_{i,\text{mon}} \geq 0 \rightarrow & \text{inc}(c_{i,\text{mon}}) \\ c_{i,\text{mon}} < 0 \rightarrow & \text{inc}(c_{i,\text{fluct}}) \end{cases} \tag{5}$$

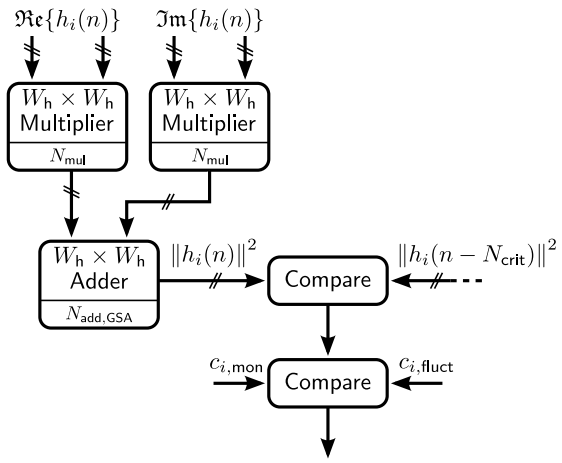


Fig. 4. Implementation scheme of the GSA. The calculation of $\|h_i(n - N_\mathsf{crit})\|^2$ is analog to that of $\|h_i(n)\|^2$ and was left out for reasons of space.

and

$$\|h_i(n - N_\mathsf{crit})\|^2 \geq \|h_i(n)\|^2 \begin{cases} c_{i,\mathsf{mon}} \leq 0 \rightarrow & \mathsf{dec}(c_{i,\mathsf{mon}}) \\ c_{i,\mathsf{mon}} > 0 \rightarrow & \mathsf{inc}(c_{i,\mathsf{fluct}}) \end{cases} \tag{6}$$

Here, $\mathsf{inc}(\cdot)$ and $\mathsf{dec}(\cdot)$ denote increment and decrement of a counter respectively. In this scheme, negative values of $c_{i,\mathsf{mon}}$ indicate monotonic decrease of the magnitude while positive values show increase. $c_{i,\mathsf{fluct}}$ is always non-negative. A schematic implementation structure is given in Fig. 4. Real and imaginary part of the current coefficients are squared and added. The same happens for the old coefficients $h_i(n-N_\mathsf{crit})$ which have to be stored beforehand. In two comparator stages the necessary alteration of the counters is decided.

The presented hardware concept can now be synthesized if concrete values for word widths and delays are defined. For this a digital receiver unit from a research project described in [5] and [6] is taken. The system at hand aims at multiple beam capability and has two identical receiver paths. Each of these uses a four element antenna array to receive signals which are mixed down before being processed inside an FPGA. The structure shown in Fig. 3 was implemented along with the presented step-size control techniques. The values of the different components are shown in Table I. If the processed signals are complex, input and output widths are doubled as real and imaginary part need separate buses. The table also shows the processing delay in samples. The comparator stages have a delay of $N_\mathsf{com} = 1$. It can be seen that an adaptive cycle takes $N_\mathsf{crit} = 31$ samples. Given a sample rate of 100 MHz which is reasonable for a Virtex-4™, this leads to a duration of 310 ns per iteration.

Currently the step-size calculation for GSA as shown in Fig. 4 uses 16 multipliers and 8 adders for all four channels. This can be optimized as follows. The calculation of the new step-size can start after new filter coefficients are calculated. The result is needed after $N_\mathsf{delay} + N_\mathsf{xe} = 14$ samples. The

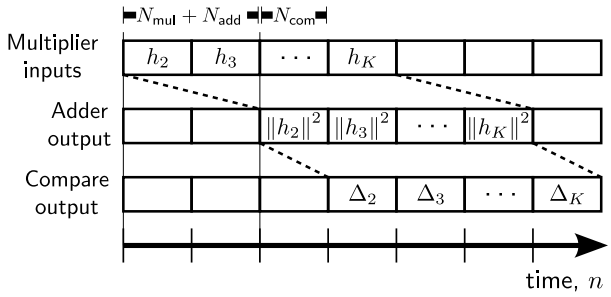


Fig. 5. Pipelining scheme for GSA.

computation as shown in Fig. 4 takes $N_\mathsf{mul} + N_\mathsf{add,GSA} + 2 \cdot N_\mathsf{com} = 7$ samples (plus one to alter $\mu$, if necessary). This difference can be used to calculate step-sizes for several filters serially instead of in parallel. This is done by pipelining as shown in Fig. 5: At the first time instant, the coefficients of filter 2, $h_2(n)$ and $h_2(n-N_\mathsf{crit})$ are fed into the multipliers. At the next instant, the coefficients of filter 3 are used and so on. Pipelining means that these input changes do not interfere with each other during processing so that after the respective delays $N_\mathsf{mul}+N_\mathsf{add,GSA}$ the adder outputs show first $\|h_2\|^2$, $\|h_3\|^2$ and so on. By applying multiplexers at the input and output of the whole computing unit, the results can be routed to the correct filter. By this, resource demand drops to 4 multipliers and 2 adders. The multiplexing network can be realized by cheap standard logic. Saving resources on the FPGA is important especially for space applications where power consumption is a big issue. A similar pipelining could also be used for the whole innovation calculation process in Fig. 3. However, this would be a trade-off between saving resources and slowing down the filter update rate.

## IV. Hardware performance and results

Measurements with the complete receiver system were conducted to evaluate the performance of the different calibration techniques.

Since the measurement procedure differs considerably from the simulation conditions, the following points should be noted: It is not easily possible to monitor the development of the step-size and the filter coefficients during adaption since these are internal values. To show the coefficient development, the number of adaptive iterations is varied and the final values

TABLE I
Component data for the example system

| Component | Bit width Input 1 | Input 2 | Output | Delay |
|---|---|---|---|---|
| Delay line | $2 \times 30$ | – | $2 \times 30$ | $N_\mathsf{delay} = 14$ |
| $W_\mathsf{x} \times W_\mathsf{h}$ Multiplier | $2 \times 30$ | $2 \times 16$ | $2 \times 47$ | $N_\mathsf{h} = 11$ |
| $W_\mathsf{y}$-bit Adder | $2 \times 31$ | $2 \times 31$ | $2 \times 32$ | $N_\mathsf{add} = 3$ |
| $W_\mathsf{x} \times W_\mathsf{e}$ Multiplier | $2 \times 30$ | $2 \times 30$ | $2 \times 40$ | $N_\mathsf{xe} = 11$ |
| Shifter | $2 \times 40$ | 8 | $2 \times 16$ | $N_\mathsf{shift} = 1$ |
| $W_\mathsf{h}$-bit Adder | $2 \times 16$ | $2 \times 16$ | $2 \times 16$ | $N_\mathsf{innov} = 2$ |
| $W_\mathsf{h} \times W_\mathsf{h}$ Multiplier | 16 | 16 | 16 | $N_\mathsf{mul} = 3$ |
| $W_\mathsf{h} \times W_\mathsf{h}$ Adder | 16 | 16 | 16 | $N_\mathsf{add,GSA} = 2$ |

are plotted. Although the time grid is much coarser, the adaption speed can be attained and compared. As mentioned in section III the step-size multiplication is implemented as a shifter. The value of $\mu$ is the number of bit shifts, so the step-sizes cause a multiplication of the term $e_i(n)\mathbf{x}_i^*(n)$ by $2^\mu$. The result is then truncated to the coefficient word width. With proper balance of bit shift and truncation, this leads to a suitable innovation signal. Inside the calibration unit, signals are truncated at several points (Table I). This is necessary to limit component word widths but introduces noise and may even cause information loss through clipping.

Measurements are conducted using a complete receiver system: The chain starts with a 26.4 GHz sinusoidal signal generator which drives a transmit horn antenna. The signal is received by an AFR whereby four patch antennas are used. Analog front-ends perform the downmix to base band where the signal is digitized and processed. After the calibration unit, there is a single output sum signal which is measured by a signal analyzer. Besides the magnitude of one filter coefficient, the SNR level at the output of the beamformer is measured to have a criterion for the adaption quality. The results are shown in Fig. 6. Since there are many sources for random variations, all measurement points are averaged over 20 successive adaption passes.

To cover the necessary number of adaption steps and keep track of the initial behavior, the number of iterations was varied non-linearly: The first 16 variants increased the iteration number from 10 to 160. Afterwards, there were 16 variants from $2^8$ to $2^{23}$, each doubling the number of iterations. Six different calibration techniques were analyzed: Three with fixed step-size values of $\mu \in \{8, 12, 16\}$, one with SRA starting at $\mu_{\mathsf{SRA}} = 16$, and two using the GSA with $\mu_{\mathsf{GSA}} \in \{8, 16\}$.

Regarding the coefficient magnitudes for fixed step-size values in Fig. 6a, the simulated behavior shown in Fig. 2a is reproduced. The higher the step-size, the faster the adaption takes place. Smaller step-sizes show less variation of the final value. The techniques with variable step-sizes (Fig. 6b) reach adaption speeds close to that of fixed $\mu = 16$. The final values are held more stable in general. SRA and GSA with $\mu = 8$ attain a quality similar to the low fixed step-sizes. Interestingly, the initial choice of $\mu_{\mathsf{GSA}}$ has a long lasting influence on performance. This is influenced by the choice of the counter limits in (4). If these are rather large, the step-size can only be changed slowly as shown here. Given a large start step-size, GSA can then adapt faster than SRA. In Fig. 6c and 6d, it can be seen that for all methods the coefficient phase adapts faster than the magnitude. This comes from the coefficients getting near zero during the adaption process.

From a system point of view, it is important in how far calibration improves the SNR of the output. For the uncalibrated system, i.e. with equal filter coefficients, SNR is around 34 dB. In Fig. 6e and 6f, it can be seen that the SNR performance depends rather on phase than on magnitude. Regarding the final values, all methods are capable of increasing the SNR by more than 6 dB. The methods with variable step-size have a slight advantage both in speed and final performance. Especially for few iteration steps, the attained values are better.

## V. Conclusion

In this paper the complete design cycle of an adaptive calibration unit with step-size control for antenna arrays was shown, starting from simulations over an implementation concept to validation and measurement on actual hardware. Additionally, some optimizations were presented regarding the usage of hardware resources. Due to the final implementation, timing analysis, and resource effectiveness were of interest. It was shown that step-size control offers an advantage over fixed step-sizes. The final coefficient values can in general be reached faster and/or held more steadily. Moreover, in terms of calibration performance, measured by SNR, step-size control methods yield better results. This comes at the expense of additional implementation effort and resource demand. Depending on the application, it has to be decided if better performance justifies the additional cost. As long as it can be assured that the adaption does not diverge for large step-sizes, a fixed $\mu$ is an attractive choice due to simple implementation. If this is not the case, the use of the GSA is a good alternative since its performance is even slightly better and it is a truly adaptive technique in contrast to SRA.

For further investigations, the calibration unit should be made robust against interference from unwanted directions. The use of a known training sequence may be an option. An additional optimization could be the introduction of pipelining in the innovation calculation part.

## Acknowledgment

The authors would like to thank all partners of the GeReLEO project for providing the demonstrator system which was used for validation.

## References

[1] S. S. Haykin, *Adaptive Filter Theory*, 3rd ed. New Jersey: Prentice-Hall, 1996, p. 367 ff.

[2] R. Kwong and E. Johnston, "A variable step size LMS algorithm," *Signal Processing, IEEE Transactions on*, vol. 40, no. 7, pp. 1633 –1642, Jul. 1992.

[3] D. Goelz, C. Mandel, and P. Meissner, "Fast Adapting PMD Equalizers Using Adaptive Step-Size Control," in *PhotonicsGlobal@Singapore, 2008. IPGC 2008. IEEE*, Dec. 2008, pp. 1 – 4.

[4] R. Harris, D. Chabries, and F. Bishop, "A variable step (VS) adaptive filter algorithm," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 2, pp. 309 – 316, Apr. 1986.

[5] L. Greda, A. Winterstein, A. Dreher, and M. Brück, "Demonstrator concept for a satellite multiple-beam antenna for high-rate data relays," in *Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC), 2012 6th*, Sep. 2012, pp. 96 – 100.

[6] L. A. Greda, B. Knüpfer, J. S. Knogl, M. V. T. Heckler, H. Bischl, and A. Dreher, "A multibeam antenna for data relays for the german communications satellite heinrich-hertz," in *2010 Proceedings of the Fourth European Conference on Antennas and Propagation (EuCAP)*, April 2010, pp. 1 – 4.

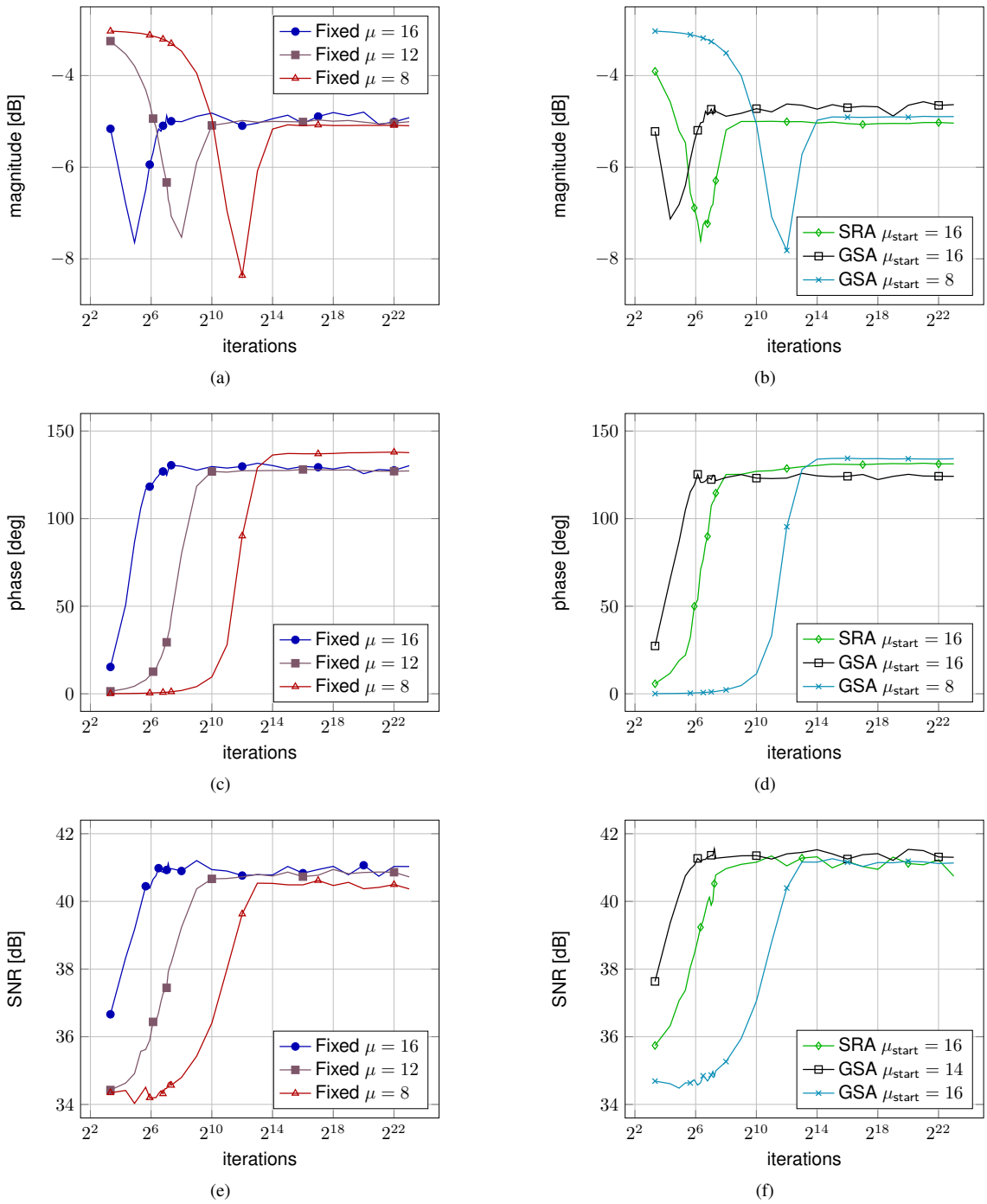


Fig. 6. Results of adaption processes with different length. (a) and (c) depict the coefficient development for fixed step-sizes, magnitude and phase respectively. In (b) and (d) the same is shown for step-wise reduction and gradient sign algorithm, the latter with two different initial step-size settings. The magnitude plots are normalized to the maximum value defined by the coefficient word width. Calibration starts with −3 dB in order to have enough margin for adaption. Phase adapts more quickly than magnitude. (e) and (f) show the SNR of the receiver output signal after calibration for the different techniques. As can be seen in (e) for the low step-size, SNR for the uncalibrated system is around 34 dB.