FP6-IST-2003-506745 CAPANINA

Deliverable Number D28

Detailed design of adaptive beamforming algorithms for ground terminals and aerial platform antennas

Document Number	CAP-D28-WP33-UOY-PUB-01
Contractual Date of Delivery to the EC	1st November 2006
Actual Date of Delivery to the EC	31st July 2006
Author(s)	Emanuela Falletti, Luigi Rega, Fabrizio Sellone, Marco Urso
Participant(s) (partner short names)	POLITO
Editor (Internal reviewer)	Tomaž Javornik (JSI)
Workpackage	3.3
Estimated person months	22
Security (PUBlic, CONfidential, RE-Stricted)	PUB
Nature	Report
CEC Version	1.1
Total number of pages (including cover)	38

Abstract:

This report analyzes a numerically robust implementation of a beamforming algorithm that suppresses Doppler shift in receiving OFDM signals with an adaptive array antenna working in a HAP-to-train link. This algorithm is developed according to a multi-rank update RLS approach, in order to control the array weights on the basis of more than one known signal, simultaneously carried by orthogonal sub-carriers in the frequency domain. The analyzed algorithm is an extension of the solution presented in the Capanina Deliverable D17, that proposed a rank-1 RLS algorithm applied to a Single-Carrier IEEE 802.16 communication link. The algorithm performance is shown to be scalable with the rank, so that the implementation analysis of the multi-rank description is directly related to the rank-1 one.

The improvements in terms of convergence speed and residual error are evaluated by computer simulation with respect to other approaches and validated by VHDL synthesis of an *ad hoc* beamforming device designed in programmable logic.

Keyword list: Smart antennas, Beamforming, RLS, Multi-rank RLS, QR Decomposition, VHDL

DOCUMENT HISTORY

Date	Revision	Comment	Author/Editor	Affiliation
30/06/2006	01.0	First draft	L. Rega, E. Falletti	Polito
18/07/06	01.1	Document update and final draft	M. Urso, E. Falletti	Polito
28/07/06	01.2	Final revision	G. White	UoY

Document Approval (CEC Deliverables only)

Date of approval	Revision	Role of approver	Approver	Affiliation
31/07/06	01	Editor (internal reviewer)	Tomaž Javornik (JSI)	JSI
31/07/06	01	On behalf of Scientific Board	David Grace	UOY

EXECUTIVE SUMMARY

This report investigates the main implementation issues of a particular adaptive smart antenna algorithm, especially tailored for ground terminals receiving OFDM transmission from a HAP, based on the IEEE 802.16a standard.

The first issue considered is the **stability** of the algorithm, which can be seriously affected when quantization of the data is performed. The second issue is the **implementation complexity,** represented by the number of machine instructions to be computed per second; it yields a constraint on the minimum **clock frequency** required on a real-world receiver. The third issue, clearly related, is the number of **quantization bits** used to represent the data handled by the algorithm in its different parts, because it determines both the quantization error and the size of the requires memory.

It is worth firstly noticing that the beamforming approach discussed hereafter is an extension of that presented in Deliverable D17 [1], developed for a Single-Carrier IEEE 802.16 communication link. Since the algorithm performance is, in some way, scalable with the number of (sub)carriers, the implementation analysis discussed in this report is directly related to the Single-Carrier case.

In order to select a computationally light architecture, a time domain, or pre-FFT, beamforming approach is considered, whose characteristics are also well suited for the flat fading channel that may be experienced between HAP and train. In OFDM systems there is the possibility to work in the presence of a multitude of input signals simultaneously carried by orthogonal sub-carriers in the frequency domain. To exploit this fact, a multirank beamforming algorithm based on the standard Recursive Least Squares approach is the starting point of the analysis. However, with standard RLS approach, Multirank RLS is unstable if developed with finite precision arithmetic, and it requires prohibitively large computational effort. Thus, following the option already discussed in [1], the QR Decomposition-based recursive implementation is of the utmost interest to obtain a stable and low-complexity algorithm.

The Multirank QRD-RLS is proved to be stable on a finite precision arithmetic device, provided that a sufficient wordlength is used, and to require a lower computational effort than the standard Multirank RLS. A comparative analysis of the main implementation issues is also presented. To solve the Least-Square problem formulation, in the rank-1 as well as the rank-P case, it is a common practice to exploit the fact that each auto-correlation matrix and each cross-correlation vector involved in the formal solution of the problem can be written as Rank-P updates of the same quantities evaluated at the previous time instant. Then, a recursive implementation of the solution can be conceived, by computing a series of P subsequent Rank-1 updates, then applying the matrix inversion lemma (Multirank RLS). Unfortunately, because of the numerical instability of the standard RLS algorithm, perturbations tend to be amplified during the recursions, independently from the rank. This problem is solved by the QRD approach and it is exploited in the Multirank QRD-RLS.

In addition, the Multirank QRD-RLS shows good properties wth respect to the finite precision implementation:

- it will be shown that an upper bound exists for the entries of the autocorrelation matrix and the cross-correlation vector;
- the algorithm can be implemented with a relatively low computational effort in programmable logic.

In terms of Doppler resilience capability, the Multirank QRD-RLS and Multirank RLS behave identically with infinite precision arithmetic, while they are different in terms of computational effort and numerical stability. Furthermore, as rank increases, the beamformer approaches the optimum solution more closely and more quickly. In the signal and environmental conditions considered in the report, the Doppler shift becomes critical above a ± 9 kHz threshold, whereas, below that threshold, a proper choice of the algorithm parameters allows compensation of the Doppler shift.

For implementation of the algorithms on a finite precision arithmetic device, the problem of wordlength, i.e. the number of bits used to represent each data value in the algorithm, must be addressed. The analysis here considers both an analytical and an empirical study, that show good agreement. Thus, the most suitable choice for the wordlength of the integer part of the data is 6, i.e., 5 bits for the modulus and one for the sign, while, for the fractional part, significantly higher numbers of bit are necessary to achieve low residual errors w.r.t. the unquantized algorithm. A good compromise is shown to be 14 bits for the fractional parts. Summing up, a 20 bits total wordlength is necessary to guarantee acceptable

performance, whilst 26 bits are required for the computation of certain parts of the QRD algorithm that are particularly sensitive to round-off errors.

Nonetheless, irrespective of wordlength, the quantized implementation of the Multirank RLS is unstable and tends to diverge, while the QRD-RLS is able to preserve stability.

Finally, hardware device has been synthesized using the VHDL language, in order to validate the developed algorithm. The results of the VHDL synthesis match closely those obtained by computer simulations; this demonstrates that the proposed algorithm can be successfully synthesized in programmable logic, using the VHDL code developed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

One of the key issues in the communication system design between the HAP and an high speed train is the non-negligible relative movement of the link end-points. This means that the high gain antenna systems of both the platform and the train must have steering capabilities, that can be provided either by means of mechatronics devices, or by a beamforming system. Both solutions have been simultaneously addressed within the CAPANINA project [1, 3], showing that a trade-off exists between realization complexity and tracking performance.

Whereas for the HAP antenna system it may be preferable, in some cases, to implement a fixed, hand-over based, spotbeam coverage of the ground area, the ground terminal antenna is required to be able to steer its main radiation beam in real-time toward the position of the platform, thus employing adaptive tracking beamforming.

Furthermore, the propagation channel between the HAP and the ground terminal is likely to impair the transmitted signal with significant non-periodic Doppler effect, due to the motion of both link end-points, and with flat or slightly frequency-selective fading, due to atmospheric scattering reflections from very smooth surfaces of man-made structures [4].

In this situation, smart antennas systems, aimed at adaptively shaping the equivalent radiation pattern of the antenna array and simultaneously compensating for fading and Doppler effects, are one of the best candidates for the ground terminal transceiver. Following the conclusions of Deliverable D17 [1], in this report analyzes the implementation aspects of an adaptive smart antenna algorithm, especially tailored for ground terminals, that receives an OFDM transmission from the HAP. The HAP holds the transmitter while the receiver is mounted onto the ground terminal, possibly the high-speed train.

The main focus of the work is to investigate the suitability of the algorithm to be implemented on an electronic device (e.g., an FPGA), that works in finite-precision arithmetics. One of the first issues to be investigated is the **stability** of the algorithm, that can be seriously affected by the quantization errors. The second issue is the **implementation complexity,** represented by the number of instructions to be computed per second; it yields a constraint on the minimum **clock frequency** required by the device. The third issue, clearly related, is the number of **quantization bits** used to represent the data handled by the algorithm in its different parts, because it determines both the quantization error and the size of the allocated memory.

The beamforming approach discussed hereafter is an extension of that presented in Deliverable D17, developed for a Single-Carrier IEEE 802.16 communication link. Since the algorithm performance is, in some way, scalable with the number of (sub)carriers, the implementation analysis discussed this report is directly suitable to the Single-Carrier case.

Furthermore, since the computational complexity is an issue and it has to be kept as low as possible, a time domain, or pre-FFT, beamforming is considered. This means that its scheme can be directly adopted in a Single-Carrier system, where a frequency-domain beamforming would be economically disadvantageous and the time-domain approach is the far most commonly one. Last but not least, the time-domain solution is also well suited for the kind of flat fading channel between HAP and train.

In contrast to a Single-Carrier system, in OFDM there is the possibility to work in the presence of a multitude of input signals simultaneously carried by some orthogonal sub-carriers in the frequency domain. To exploit this fact, a multirank beamforming algorithm, based on the standard Recursive Least Squares (Multirank RLS) approach, is proposed in [2]. However, as the well known standard RLS, also the Multirank RLS is unstable if developed with finite precision arithmetic [5], and it requires remarkable computational effort. Since it is known that the QR Decomposition (QRD) approach makes the RLS algorithm stable (see [1] and references therein), the QRD *recursive* implementation is of the utmost interest to obtain a stable and low-complexity algorithm. A multirank, recursive, QRD RLS (Multirank QRD-RLS) is adopted for the architecture addressed in this report. It can be recognized to be formally similar to the block- RLS solution proposed for a different single-signal application context in [6], which is based on Householder reflections, instead of Givens rotations. In this report the Multirank QRD-RLS is proved to be stable on a finite precision arithmetic device, provided that a sufficient wordlength is used, and to require a lower computational effort than the Multirank RLS [2].

Finally an hardware device has been synthesized using the VHDL language, in order to validate the

analysis performed on the selected Multirank QRD-RLS algorithm. The results of the VHDL synthesis follow the ones obtained by Matlab simulations, which demonstrates that the algorithm can efficiently work on a specific device in finite-precision arithmetics, using the developed VHDL code.

The report is organized as follows: the extension from the rank-1, i.e. the Single-Carrier, algorithm to the multirank approach is presented in Section 2, along with a brief investigation of performance in terms of Doppler shift rejection. Implementation issues in finite precision arithmetic are discussed in Section 3, where the sufficient wordlength to represent the quantized data is derived and the computational effort required by the numerical implementation the algorithm is investigated. Finally, the validation of the algorithm using VHDL language is provided in Section 4, followed by the conclusions in Section 5.

2 A multi-rank version of the classic RLS beamforming algorithm for ground terminals

In Deliverable D17 [1], a beamforming algorithm suitable for ground terminals has been presented, based on an RLS approach. Given the promising performance result offered by this algorithm, it has been chosen for a deeper analysis, aimed at investigating the suitability of this algorithm for a practical implementation, whose main challenge is represented by finite-precision arithmetic and quantization.

For the sake of generality, the approach presented in [1] is firstly extended here to a multi-dimensional signal, i.e. for OFDM modulation, showing that the performance of the RLS algorithm can be improved if, instead of considering only one sample every step, P samples are used to refine the estimation computed by the algorithm (P-dimensional signal).

If we suppose to work with P input signal vectors $\mathbf{x}_p[n]$ with $p = 1, 2, ... P$ and P desired signals $d_p[n]$ with $p = 1, 2, ... P$, it is possible to consider that the same linear processor $\mathbf{w}[n]$ discussed in [1] elaborates the P input signals in order to produce P outputs $y_p[n]$ with $p = 1, 2, ... P$, as close as possible to the corresponding desired signals $d_p[n]$, $p = 1, 2, ... P$.

To this purpose, by extension of [1], a new cost function can be written for this novel scenario as follows

$$J_P(\mathbf{w}[n]) \triangleq \sum_{p=1}^{P} \sum_{\ell=1}^{n} \lambda^{n-\ell} |e_p[\ell, n]|^2 \tag{1}$$

where the error function is now a collection of errors between each input signal and the corresponding desired one.

The novel Least Squares optimization problem can be stated as follows

$$\mathbf{w}_P[n] = \arg \min_{\mathbf{w}[n]} \sum_{p=1}^{P} \sum_{\ell=1}^{n} \lambda^{n-\ell} |e_p[\ell, n]|^2 = \arg \min_{\mathbf{w}[n]} \sum_{p=1}^{P} \sum_{\ell=1}^{n} \lambda^{n-\ell} |d_p[\ell] - \mathbf{w}^H[n] \mathbf{x}_p[\ell]|^2 \tag{2}$$

It will be referred to as *Rank-P Least Squares problem,* since it uses P input vector signals and P desired signals to update the weight vector.

2.1 The OFDM transceiver model

In an OFDM transmitter, the binary data stream is modulated and parallelized into N_d sub-streams to fill an equal number of frequency sub-carriers. Then, N_p modulated pilot sub-carriers are inserted, evenly spaced among the others, along with N_z zero sub-carriers required to avoid aliasing. The whole group of sub-carriers is transformed in the time domain via an N_{FFT}-point Inverse Fast Fourier Transform (IFFT) and then serialized, to form the so called *OFDM symbol*. Then, the signal is cyclically extended. Finally, after frequency up-conversion, the signal is transmitted.

At the receiver side, whose basic structure is depicted in Figure 1, each sensor of the array receives a signal resulting by the sum of the direct signal, multipath, interferers and noise. The latter is modeled as an additive white Gaussian noise (AWGN), mutually independent at each antenna element. The received signals could be subject to Doppler shift and fading.

The beamforming weight vector is designed to both steer the equivalent beampattern toward the HAP and to recover the Doppler shift on the received signal on the basis of the received pilots and zeros sub-carriers of the OFDM symbol. Indeed, the receiver knows the training sequences carried by the pilot sub-carriers, so that it can exploit them, along with the zero sub-carriers as the set of reference signals necessary to the beamformer to adapt its M-element weight vector $\mathbf{w}[n]$ at the n-th OFDM symbol.

The receiver of Figure 1 is divided in two main parts: the upper part is used to estimate beamforming weights, while the lower one is the classical OFDM receiver which performs, after beamforming, the reversed process encountered at the transmitter. In the upper part, the desired signals $d_p[k]$ are the known sequences for all the indexes p spanning the pilot sub-carriers and the zeros sub-carriers. The time index k refers to the k-th OFDM symbol. Let $\mathbf{U}[k] \in \mathbb{C}^{M, N_{\mathrm{FFT}}}$ be the matrix whose columns are the received array signal vectors taken from the k-th OFDM symbol after cyclic prefix extraction. The

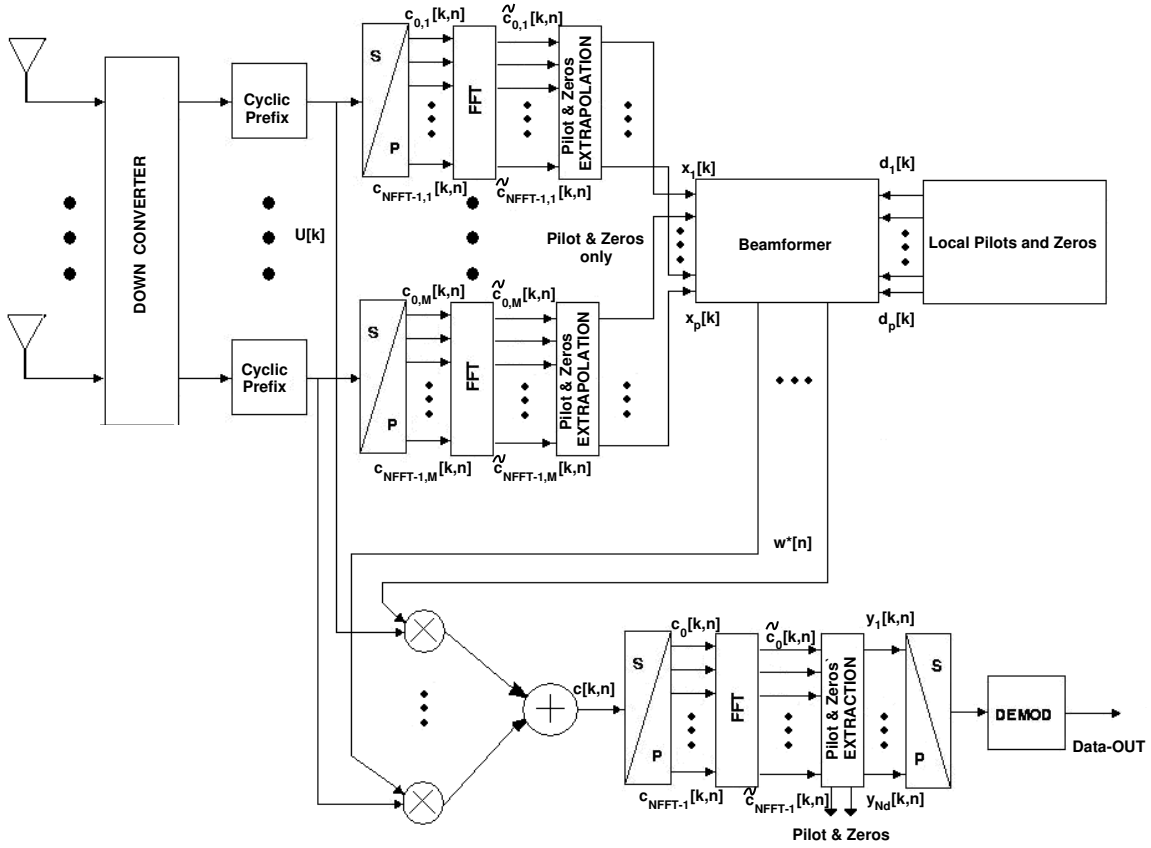


Figure 1: Basic structure of an OFDM receiver with time domain beamformer.

signal after beamforming, spatially filtered by the the weight vector $\mathbf{w}[n]$, can be written as $\mathbf{c}[k,n] = \mathbf{U}^{\mathrm{T}}[k]\mathbf{w}^{*}[n]$. Being the matrix $\mathbf{F}$ the FFT operator, the signal after the FFT block becomes $\tilde{\mathbf{c}}[k,n] = \mathbf{F}\mathbf{c}[k,n]$. In order to select the p-th sub-carrier, let us formally multiply $\mathbf{c}[k,n]$ by $\mathbf{g}_p$ that is a vector made by all zeros except for the p-th entry which is one. So, the signal received from the p-th subcarrier is written as

$$y_p[k,n] = \mathbf{g}_p^{\mathrm{T}}\mathbf{F}\mathbf{U}^{\mathrm{T}}[k]\mathbf{w}^{*}[n] = \mathbf{w}^{\mathrm{H}}[n]\mathbf{U}[k]\mathbf{F}^{\mathrm{T}}\mathbf{g}_p = \mathbf{w}^{\mathrm{H}}[n]\mathbf{x}_p[k] \tag{3}$$

where it is useful to identify the vector $\mathbf{x}_p[k] = \mathbf{U}[k]\mathbf{F}^{\mathrm{T}}\mathbf{g}_p$.

The purpose of the beamformer is to reproduce the spatial signature of the desired signal impinging on array while minimizing the interferers and noise contributions and compensating the Doppler shift. For the beamformer addressed here, we suppose that P received pilots and zeros sub-carriers of the n-th OFDM symbol $\mathbf{x}_p[n],\ p = 1,\ 2,\ \ldots\ P$ with $P = N_p + N_z$ are simultaneously available for processing along with P corresponding desired signals $d_p[n]$. Thus, as anticipated, the same linear processor $\mathbf{w}[n]$ can be used to elaborate the P input signals in order to produce P outputs $y_p[n],\ p = 1,\ 2,\ \ldots\ P$ as close as possible to the corresponding desired signals.

2.2 Rank-P Least Squares problem solution

In order to derive a closed form solution to problem (2) let us define the following quantities, $\forall\ p = 1,\ 2,\ ...\ P$:

$$\begin{aligned}
\mathbf{X}_p[n] &\triangleq [\mathbf{x}_p[1],\ \mathbf{x}_p[2],\ ...\ \mathbf{x}_p[n]] \quad \in \mathbb{C}^{M,n} && (4)\\
\mathbf{y}_p[n] &\triangleq [\mathbf{y}_p[1,n],\ \mathbf{y}_p[2,n],\ ...\ \mathbf{y}_p[n,n]]^T = \mathbf{X}_p^T[n]\mathbf{w}^*[n] \quad \in \mathbb{C}^{n,1} && (5)\\
\mathbf{d}_p[n] &\triangleq [\mathbf{d}_p[1],\ \mathbf{d}_p[2],\ ...\ \mathbf{d}_p[n]]^T \quad \in \mathbb{C}^{n,1} && (6)\\
\mathbf{e}_p[n] &\triangleq \mathbf{d}_p[n] - \mathbf{y}_p[n] \quad \in \mathbb{C}^{n,1} && (7)\\
\mathbf{\Lambda}[n] &\triangleq \text{diag}\,\{[\lambda^{n-1},\ \lambda^{n-2},\ ...\,\lambda,\ 1]\} \quad \in \mathbb{C}^{n,n} && (8)
\end{aligned}$$

Note that they are the same quantities already defined in [1], but they depend on the signal subscript p.

The solution to problem (2) can be found by zeroing the complex gradient of (1) taken with respect to $\mathbf{w}^*[n]$. By defining the auto-correlation matrix $\mathbf{R}_{\mathbf{xx}}[n]$ and the cross-correlation vector $\mathbf{r}_{\mathbf{x}d}[n]$ as

$$\mathbf{R}_{\mathbf{xx}}[n] \triangleq \sum_{p=1}^{P} \mathbf{X}_p[n]\mathbf{\Lambda}[n]\mathbf{X}_p^H[n] \quad \in \mathbb{C}^{M,M} \tag{9}$$

$$\mathbf{r}_{\mathbf{x}d}[n] \triangleq \sum_{p=1}^{P} \mathbf{X}_p[n]\mathbf{\Lambda}[n]\mathbf{d}_p^*[n] \quad \in \mathbb{C}^{M,1} \tag{10}$$

the required solution can be written as

$$\mathbf{w}_P[n] = \mathbf{R}_{\mathbf{xx}}^{-1}\mathbf{r}_{\mathbf{x}d}[n]. \tag{11}$$

We can notice that if $P = 1$, then the classical LS (Least Squares) solution shown in [1] is obtained.

2.2.1 Standard Multi-rank Recursive Least Square solution

As discussed in [1], the direct solution proposed in Equation (11) is unfeasible from a practical point of view, since the computational effort necessary to obtain the optimum weight vectors requires a full matrix inversion, which has $O(M^3)$ complexity, with M standing for the number of rows of the square matrix $\mathbf{X}_p[n]$, which grows, as $\mathbf{\Lambda}[n]$ and $\mathbf{d}_p[n]$, with the time index n.

To circumvent this problem, it is a common practice in the Rank-1 case to exploit the fact that the auto-correlation matrix $\mathbf{R}_{\mathbf{xx}}[n]$ and the cross-correlation vector $\mathbf{r}_{\mathbf{x}d}[n]$ can be written as time updates of the same quantities evaluated at the previous time instant [1]. Analogously, for the Rank-P case, the auto-correlation matrix and the cross-correlation vector can be written as a rank-P update of the same quantities evaluated at the previous time instant, as

$$\mathbf{R}_{\mathbf{xx}}[n] = \lambda\mathbf{R}_{\mathbf{xx}}[n-1] + \sum_{p=1}^{P} \mathbf{x}_p[n]\mathbf{x}_p^H[n] \tag{12}$$

$$\mathbf{r}_{\mathbf{x}d}[n] = \lambda\mathbf{r}_{\mathbf{x}d}[n-1] + \sum_{p=1}^{P} \mathbf{x}_p[n]d_p^*[n] \tag{13}$$

Consequently, as for the Rank-1 case, a recursive implementation of the solution (11) can be conceived by exploiting the matrix inversion lemma on Equation (12).

Having P rank updates to perform on a matrix, Equation (12) can be rewritten as follows

$$\begin{aligned}
\mathbf{R}_1[n] &\triangleq \lambda\mathbf{R}_{\mathbf{xx}}[n-1]\\
\mathbf{R}_2[n] &\triangleq \mathbf{R}_1[n] + \mathbf{x}_1[n]\mathbf{x}_1^H[n]\\
\mathbf{R}_3[n] &\triangleq \mathbf{R}_2[n] + \mathbf{x}_2[n]\mathbf{x}_2^H[n] && (14)\\
&\vdots\\
\mathbf{R}_{P+1}[n] &\triangleq \mathbf{R}_P[n] + \mathbf{x}_P[n]\mathbf{x}_P^H[n] = \mathbf{R}_{\mathbf{xx}}[n]
\end{aligned}$$

obtaining $\mathbf{R}_{\mathbf{xx}}[n]$ which represents the Rank-P update of $\mathbf{R}_{\mathbf{xx}}[n-1]$.

In the same way, Equation (13) can be rewritten as follows

$$\begin{aligned} \mathbf{r}_1[n] &\triangleq \lambda \mathbf{r}_{\mathbf{x}d}[n-1] \\ \mathbf{r}_2[n] &\triangleq \mathbf{r}_1[n] + \mathbf{x}_1[n]\mathbf{x}_1^H[n] \\ \mathbf{r}_3[n] &\triangleq \mathbf{r}_2[n] + \mathbf{x}_2[n]\mathbf{x}_2^H[n] \\ &\vdots \\ \mathbf{r}_{P+1}[n] &\triangleq \mathbf{r}_P[n] + \mathbf{x}_P[n]d_P^*[n] = \mathbf{r}_{\mathbf{x}d}[n] \end{aligned} \tag{15}$$

obtaining $\mathbf{r}_{\mathbf{x}d}$ which represents the Rank-P update of $\mathbf{r}_{\mathbf{x}d}[n-1]$.

By invoking the Matrix Inversion Lemma[1], it is possible to write

$$\mathbf{R}_{p+1}^{-1}[n] = \left(\mathbf{I} - \mathbf{k}_p[n]\mathbf{x}_p^H[n]\right)\mathbf{R}_p^{-1}[n] \qquad p = P-1, \ldots 1 \tag{16}$$

where

$$\mathbf{k}_p[n] \triangleq \frac{\mathbf{R}_p^{-1}[n]\mathbf{x}_p[n]}{1 + \mathbf{x}_p^H[n]\mathbf{R}_p^{-1}[n]\mathbf{x}_p[n]} = \mathbf{R}_{p+1}^{-1}[n]\mathbf{x}_p[n] \tag{17}$$

is known as the *Kalman gain vector.*

Thus, it is possible to exploit P times the matrix inversion lemma (one for each rank-1 update) on the same matrix by iterating Equation (16), and write

$$\mathbf{R}_{P+1}^{-1}[n] = \prod_{p=P}^{1}\left(\mathbf{I} - \mathbf{k}_p[n]\mathbf{x}_p^H[n]\right)\mathbf{R}_1^{-1}[n] \tag{18}$$

$$\mathbf{R}_{P+1}^{-1}[n] = \frac{1}{\lambda}\prod_{p=P}^{1}\left(\mathbf{I} - \mathbf{k}_p[n]\mathbf{x}_p^H[n]\right)\mathbf{R}_{P+1}^{-1}[n-1] \tag{19}$$

If we define for convenience

$$\mathbf{U}_r[n] \triangleq \prod_{p=P}^{r}\left(\mathbf{I} - \mathbf{k}_p[n]\mathbf{x}_p^H[n]\right) \qquad r = 1, 2, \ldots P \tag{20}$$

with $\mathbf{U}_{P+1}[n] = \mathbf{I}$, and

$$\mathbf{R}_r[n] \triangleq \mathbf{R}_{r-1} + \mathbf{x}_{r-1}[n]\mathbf{x}_{r-1}^H[n] \tag{21}$$

as in Equation (14), it is possible to rewrite Equation (18) as

$$\mathbf{R}_{P+1}^{-1}[n] = \mathbf{U}_r[n]\mathbf{R}_r^{-1}[n] \qquad r = 1, 2, \ldots P \tag{22}$$

By inserting Equations (22) and (13) in Equation (11) we obtain

$$\begin{aligned} \mathbf{w}[n] &= \mathbf{R}_{\mathbf{xx}}^{-1}[n]\mathbf{r}_{\mathbf{x}d}[n] = \mathbf{R}_{P+1}^{-1}[n]\mathbf{r}_{\mathbf{x}d}[n] \\ &= \lambda\mathbf{U}_r[n]\mathbf{R}_r^{-1}[n]\mathbf{r}_{\mathbf{xd}}[n-1] + \mathbf{U}_r[n]\mathbf{R}_r^{-1}[n]\sum_{p=1}^{P}\mathbf{x}_p[n]\mathbf{d}_p^*[n] \end{aligned} \tag{23}$$

By using $r = 1$ for the first addend and $r = p+1$ for the second one, Equation (23) can be rewritten as

$$\mathbf{w}[n] = \lambda\mathbf{U}_1[n]\mathbf{R}_1^{-1}[n]\mathbf{r}_{\mathbf{xd}}[n-1] + \sum_{p=1}^{P}\mathbf{U}_{p+1}[n]\mathbf{R}_{p+1}^{-1}[n]\mathbf{x}_p[n]\mathbf{d}_p^*[n] \tag{24}$$

[1]The Matrix Inversion Lemma [7] states that

$$(\mathbf{A} + \mathbf{X}\mathbf{B}\mathbf{X}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{X}(\mathbf{B}^{-1} + \mathbf{X}^T\mathbf{A}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{A}^{-1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are square and invertible matrices but need not be of the same dimension. Notice that the superscript $(.)^T$ can be substituted with the superscript $(.)^H$ when complex values are used.

Now, by recalling that $\mathbf{R}_1[n] = \lambda \mathbf{R}_{P+1}[n-1]$ it can be seen that

$$\lambda \mathbf{U}_1[n]\mathbf{R}_1^{-1}[n]\mathbf{r}_{\mathbf{xd}}[n-1] = \mathbf{U}_1[n]\mathbf{R}_{P+1}^{-1}[n-1]\mathbf{r}_{\mathbf{xd}}[n-1] = \mathbf{U}_1[n]\mathbf{w}[n-1] \tag{25}$$

Furthermore, thanks to Equation (17) we can write

$$\mathbf{U}_{p+1}[n]\mathbf{R}_{p+1}^{-1}[n]\mathbf{x}_p[n]\mathbf{d}_p^*[n] = \mathbf{U}_{p+1}[n]\mathbf{k}_p[n]\mathbf{d}_p^*[n] \tag{26}$$

At this point, by substituting equations Equation (25) and Equation (26) in Equation (24), one gets

$$\mathbf{w}[n] = \mathbf{U}_1[n]\mathbf{w}[n-1] + \sum_{p=1}^{P} \mathbf{U}_{p+1}[n]\mathbf{k}_p[n]\mathbf{d}_p^*[n] \tag{27}$$

Let us now rewrite the product

$$\begin{aligned} \mathbf{U}_1[n]\mathbf{w}[n-1] &= \mathbf{U}_2[n](\mathbf{I} - \mathbf{k}_1[n]\mathbf{x}_1^H[n])\mathbf{w}[n-1] = \\ &= \mathbf{U}_2[n]\mathbf{w}[n-1] - \mathbf{U}_2[n]\mathbf{k}_1[n]\tilde{y}_1^*[n] \end{aligned} \tag{28}$$

where

$$\tilde{y}_p[n] = \mathbf{w}^H[n-1]\mathbf{x}_p[n]. \tag{29}$$

Proceeding by induction Equation (28) can be rewritten as

$$\mathbf{U}_1[n]\mathbf{w}[n-1] = \mathbf{U}_{P+1}[n]\mathbf{w}[n-1] - \sum_{p=1}^{P} \mathbf{U}_{p+1}[n]\mathbf{k}_p[n]\tilde{y}_p^*[n] \tag{30}$$

Let us now define

$$\tilde{e}_p^*[n] = \tilde{d}_p^*[n] - \tilde{y}_p^*[n] \tag{31}$$

If we substitute Equation (30) in Equation (27), the final equation obtained is

$$\mathbf{w}[n] = \mathbf{w}[n-1] + \sum_{p=1}^{P} \mathbf{U}_{p+1}[n]\mathbf{k}_p[n]\tilde{e}_p^*[n] \tag{32}$$

The operations required to implement the standard Rank-P RLS algorithm are summarized in Algorithm (1).

2.2.2 A low-complexity solution

In [2] it is demonstrated that the standard Multi-rank RLS algorithm (1) can be implemented with lower computational effort, while preserving the same analytic solution.

This is possible because in every OFDM frame, $\mathbf{R}_{\mathbf{xx}}^{-\mathbf{1}}$, $\mathbf{r}_{\mathbf{xd}}$, and $\mathbf{w}$ are partially updated using the pilot signals.

The vector $\mathbf{w}$ updated at each pilot signal in 1 OFDM frame can be used to update $\mathbf{R}_{\mathbf{xx}}^{-\mathbf{1}}$, $\mathbf{r}_{\mathbf{xd}}$ and $\mathbf{k}$ for the next pilot signal of the same OFDM frame. This leads to have identical results but a lower computational effort compared to the one obtained in our solution where the $\mathbf{w}$ update is performed only at the last iteration on the pilot signals during the OFDM symbol.

2.2.3 QR decomposition-based Multi-rank Recursive Least Square solution

QR Decomposition (QRD) is known to be a useful approach to make Rank-1 RLS a numerically robust algorithm [1]. In order to develop the QR decomposition-based Recursive Least Squares solution to

Algorithm 1 Rank-P RLS Algorithm

Initialize: $n \leftarrow 0$
Initialize: δ, which must be a small positive constant
Initialize: $\mathbf{R}_1[0] = \delta^{-1}\mathbf{I}_{M,M}$
Initialize: $\mathbf{w}[0] = \mathbf{0}_{M,1}$
repeat
  $n \leftarrow n + 1$
  receive $\mathbf{x}_p[n], \quad p = 1,\ 2,\ ...\ P$
  receive $\mathbf{d}_p[n], \quad p = 1,\ 2,\ ...\ P$
  for $p = 1 : M$ **do**
    if $(p == 1)$ **then**
      $\mathbf{R}_p^{-1}[n] = \frac{1}{\lambda}\mathbf{R}_{p+1}^{-1}[n-1]$
    else
      $\mathbf{R}_p^{-1}[n] = R_{p-1}^{-1}[n]\Big[\mathbf{I} - \mathbf{k}_{p-1}[n]\mathbf{x}_p^H[n]\Big]$
    end if
    $\mathbf{u}_p[n] = \mathbf{R}_p^{-1}[n]\mathbf{x}_p[n]$
    $\mathbf{k}_p[n] = \frac{\mathbf{u}_p[n]}{\mathbf{I}+\mathbf{x}_p^H[n]\mathbf{u}_p[n]}$ which, after processing the *for* iteration, leads to Equation (17)
    $y_p[n] = \mathbf{w}^H[n-1]\mathbf{x}_p[n]$
    $\tilde{e}_p^*[n] = d_p^*[n] - y_p^*[n]$
  end for
  compute $\mathbf{w}[n]$ by solving the Equation (32)
  apply beamforming using $\mathbf{w}[n]$
until there are no further samples

the more general Rank-P case (*Multi-rank QRD-RLS algorithm*), it is necessary to define the following quantities:

$$\tilde{\mathbf{X}}[k] \triangleq [\mathbf{x}_1[k],\ \mathbf{x}_2[k],\ ...\ \mathbf{x}_P[k]] \quad \in \mathbb{C}^{M,P} \tag{33}$$

$$\mathbf{X}[k] \triangleq \Big[\tilde{\mathbf{X}}[1],\ \tilde{\mathbf{X}}[2],\ ...\ \tilde{\mathbf{X}}[n]\Big] \quad \in \mathbb{C}^{M,nP} \tag{34}$$

$$\tilde{\mathbf{y}}[k] \triangleq [y_1[k],\ y_2[k],\ ...\ y_P[k]] = \tilde{\mathbf{X}}^T[k]\mathbf{w}^*[n] \quad \in \mathbb{C}^{P,1} \tag{35}$$

$$\mathbf{y}[n] \triangleq \Big[\tilde{\mathbf{y}}[1],\ \tilde{\mathbf{y}}[2],\ ...\ \tilde{\mathbf{y}}[n]\Big] = \mathbf{X}^T[k]\mathbf{w}^*[n] \quad \in \mathbb{C}^{nP,1} \tag{36}$$

$$\tilde{\mathbf{d}}[k] \triangleq [d_1[k],\ d_2[k],\ ...\ d_P[k]]^T \quad \in \mathbb{C}^{P,1} \tag{37}$$

$$\mathbf{d}[n] \triangleq \Big[\tilde{\mathbf{d}}^T[1],\ \tilde{\mathbf{d}}^T[2],\ ...\ \tilde{\mathbf{d}}^T[n]\Big]^T \quad \in \mathbb{C}^{nP,1} \tag{38}$$

$$\tilde{\mathbf{e}}[k,n] \triangleq [e_1[k,n],\ e_2[k,n],\ ...\ e_P[k,n]]^T = \tilde{\mathbf{d}}[k] - \tilde{\mathbf{y}}[k] \quad \in \mathbb{C}^{P,1} \tag{39}$$

$$\mathbf{e}[n] \triangleq \Big[\tilde{\mathbf{e}}[1,n],\ \tilde{\mathbf{e}}[2,n],\ ...\ \tilde{\mathbf{e}}[n,n]\Big] = \mathbf{d}[n] - \mathbf{y}[n] \quad \in \mathbb{C}^{nP,1} \tag{40}$$

$$\mathbf{\Gamma}[n] \triangleq \mathbf{\Lambda}[n] \otimes \mathbf{I}_P \quad \in \mathbb{C}^{nP,nP} \tag{41}$$

where the symbol $\otimes$ stands for the Kronecker product and $\mathbf{I}_P$ is a $P \times P$ identity matrix. Adopting these definitions, the cost function in Equation (1) can be compactly written as

$$\begin{aligned} J_P(\mathbf{w}[n]) &= \mathbf{e}^T[n]\mathbf{\Gamma}[n]\mathbf{e}^*[n] = \Big\|\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{e}^*[n]\Big\|^2 = \\ &= \Big\|\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n] - \mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{X}^H[n]\mathbf{w}[n]\Big\|^2 \end{aligned} \tag{42}$$

where $\|.\|$ is the Euclidean norm of a vector. If we define the data matrix as

$$\mathbf{A}[n] \triangleq \mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{X}^H[n] \in \mathbb{C}^{nP,M} \tag{43}$$

its QR decomposition is given by

$$\mathbf{A}[n] \triangleq \bar{\mathbf{Q}}[n]\bar{\mathbf{R}}[n] = [\mathbf{Q}_1[n] \quad \mathbf{Q}_2[n]] \begin{bmatrix} \widehat{\mathbf{R}}[n] \\ \mathbf{0}_{nP-M,M} \end{bmatrix} \tag{44}$$

where $\bar{\mathbf{Q}}[n] \in \mathbb{C}^{nP,nP}$ is an orthogonal matrix, $\bar{\mathbf{R}}[n] \in \mathbb{C}^{nP,M}$ is an upper triangular matrix, $\mathbf{Q}_1[n] \in \mathbb{C}^{nP,M}$ and $\mathbf{Q}_2[n] \in \mathbb{C}^{nP,nP-M}$ represent a partition of the matrix $\bar{\mathbf{Q}}[n]$ and, finally, $\widehat{\mathbf{R}}[n] \in \mathbb{C}^{M,M}$ is the square upper triangular part of $\bar{\mathbf{R}}[n]$ while $\mathbf{0}_{nP-M,M} \in \mathbb{C}^{nP-M,M}$ is the part made by all zero entries. By substituting Equation (44) into Equation (42), the cost function becomes

$$J_P(\mathbf{w}[n]) = \left\| \mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n] - \bar{\mathbf{Q}}[n]\bar{\mathbf{R}}[n]\mathbf{w}[n] \right\|^2 \tag{45}$$

Furthermore, we know that the norm is invariant under multiplication of its argument by orthogonal matrices, and so let us multiply Equation (45) by $\bar{\mathbf{Q}}^H[n]$ as follows

$$\begin{aligned} J_P(\mathbf{w}[n]) &= \left\| \bar{\mathbf{Q}}^H[n]\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n] - \bar{\mathbf{R}}[n]\mathbf{w}[n] \right\|^2 = \\ &= \left\| \begin{bmatrix} \mathbf{Q}_1^H[n]\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n] - \widehat{\mathbf{R}}[n]\mathbf{w}[n] \\ \mathbf{Q}_2^H[n]\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n] \end{bmatrix} \right\|^2 \end{aligned} \tag{46}$$

in order to find the optimum weight vector $\mathbf{w}[n]$ which is the solution of the following system

$$\widehat{\mathbf{R}}[n]\mathbf{w}[n] = \mathbf{p}[n] \tag{47}$$

being

$$\mathbf{p}[n] \triangleq \mathbf{Q}_1^H[n]\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n] \in \mathbb{C}^{M,1} \tag{48}$$

The improvement from Equation (11) is related with the fact that Equation (47) represents an upper triangular system that can be easily solved by backward substitution with a reduced computational effort.

The problem now is finding an efficient way to compute $\widehat{\mathbf{R}}[n]$ and $\mathbf{p}[n]$ in terms of updates of the same quantities evaluated at the previous time instant.

To this purpose, let us re-write $\mathbf{A}[n]$ in terms of $\mathbf{A}[n-1]$, in order to explicitly build the update of the QR decomposition.

$$\begin{aligned} \mathbf{A}[n] &\triangleq \mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{X}^H[n] = \\ &= \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{\Gamma}^{\frac{1}{2}}[n-1] & \mathbf{0}_{(n-1)P,P} \\ \mathbf{0}_{P,(n-1)P} & \mathbf{I}_P \end{bmatrix} \begin{bmatrix} \mathbf{X}^H[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} = \\ &= \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{\Gamma}^{\frac{1}{2}}[n-1]\mathbf{X}^H[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} = \\ &= \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{A}[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} \end{aligned} \tag{49}$$

Now by recalling the QR decomposition of $\mathbf{A}[n]$ it is possible to write

$$\begin{aligned} \mathbf{A}[n] &= \bar{\mathbf{Q}}[n]\bar{\mathbf{R}}[n] = \\ &= \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{A}[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} = \\ &= \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{Q}[n-1]\mathbf{R}[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} = \\ &= \begin{bmatrix} \mathbf{Q}[n-1] & \mathbf{0}_{(n-1)P,P} \\ \mathbf{0}_{(P,n-1)P} & \mathbf{I}_P \end{bmatrix} \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{R}[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix}. \end{aligned} \tag{50}$$

If there exists an orthogonal matrix $\mathbf{T}[n]$ able to transform the matrix

$$\widetilde{\mathbf{R}}[n] \triangleq \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{R}[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} \quad \in \mathbb{C}^{nP,M} \tag{51}$$

into an upper triangular matrix, then the QR update is readily obtained as follows

$$\bar{\mathbf{Q}}[n] \triangleq \begin{bmatrix} \mathbf{Q}[n-1] & \mathbf{0}_{(n-1)P,P} \\ \mathbf{0}_{(n-1)P,P} & \mathbf{I}_P \end{bmatrix} \tag{52}$$

$$\bar{\mathbf{R}}[n] \triangleq \mathbf{T}[n] \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{R}[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} \tag{53}$$

There are different methods to find the orthogonal matrix $\mathbf{T}[n]$, based upon Givens rotations [8], Householder reflections [6, 8], or based upon scaled tangent rotations (STAR) [9]. The first methods are popular for their reduced computational complexity and the possibility to be efficiently implemented on systolic array structures.

A Givens rotation matrix is an orthogonal matrix able to zero out a specific element of the vector it multiplies. Thanks to a sequence of Givens rotations matrices, it could be possible to annihilate all the elements of the matrix $\tilde{\mathbf{X}}^H[n]$ contained in $\widetilde{\mathbf{R}}[n]$ in order to make the latter an upper triangular matrix. Let us define the Givens rotation matrix applied to a generic matrix $\mathbf{U}$

$$\mathbf{T}_{p,q,i}(\mathbf{U}) \triangleq \begin{bmatrix} \mathbf{I}_{P-1} & & & & \\ & c_{p,q} & & s_{p,q} & \\ & & \mathbf{I}_{q-p-1} & & \\ & -s^*_{p,q} & & c_{p,q} & \\ & & & & \mathbf{I}_{nP-q} \end{bmatrix} \tag{54}$$

observing that the entries of this matrix are all zeros except for the elements explicitly written. Let us define the c and s

$$c_{p,q} = \frac{|u_{pi}|}{\sqrt{|u_{pi}|^2 + |u_{qi}|^2}} \tag{55}$$

$$s_{p,q} = \frac{u^*_{qi}}{u^*_{pi}}\, c \tag{56}$$

where u_{pi} and u_{qi} are the p-th and q-th elements, respectively, of the i-th column of $\mathbf{U}$. The purpose of such a Givens matrix is to rotate the columns of $\mathbf{U}$ onto a plane defined by the p-th and q-th components of the i-th column of $\mathbf{U}$ in such a way that the q-th component u_{qi} is nulled out. In this sense, the matrix

$$\mathbf{T}_1[n] \triangleq \mathbf{T}_{1,nP-P+1,1}[n] \ \ldots\ \mathbf{T}_{1,nP-1,1}[n]\mathbf{T}_{1,nP,1}[n] \tag{57}$$

is designed to operate onto the matrix $\widetilde{\mathbf{R}}[n]$ in such a way that its first column is nulled out, except for the first element. Analogously, the matrix

$$\mathbf{T}_2[n] \triangleq \mathbf{T}_{2,nP-P+1,2}[n] \ \ldots\ \mathbf{T}_{2,nP-1,2}[n]\mathbf{T}_{2,nP,2}[n] \tag{58}$$

nulls out the second column, except for the first two elements. This process can be iterated obtaining the final orthogonal matrix

$$\mathbf{T}[n] \triangleq \mathbf{T}_M[n] \ \ldots\ \mathbf{T}_2[n]\mathbf{T}_1[n] \tag{59}$$

By defining $\mathbf{v}[n] \triangleq \mathbf{Q}_2^H[n]\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n]$, it is possible to write $\mathbf{p}[n]$ and $\mathbf{v}[n]$ as a function of $\mathbf{p}[n-1]$ and

$\mathbf{v}[n-1]$ as follows:

$$\begin{aligned}
\begin{bmatrix} \mathbf{p}[n] \\ \mathbf{v}[n] \end{bmatrix} &= \mathbf{Q}^H[n]\mathbf{\Gamma}^{\frac{1}{2}}[n]\mathbf{d}^*[n] = \\
&= \mathbf{T}[n] \begin{bmatrix} \mathbf{Q}^H[n-1] & \mathbf{0}_{(n-1)P,P} \\ \mathbf{0}_{P,(n-1)P} & \mathbf{I}_P \end{bmatrix} \cdot \\
&\cdot \quad \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{\Gamma}^{\frac{1}{2}}[n-1] & \mathbf{0}_{(n-1)P,P} \\ \mathbf{0}_{P,(n-1)P} & \mathbf{I}_P \end{bmatrix} \begin{bmatrix} \mathbf{d}^*[n-1] \\ \mathbf{d}^*[n] \end{bmatrix} = \\
&= \mathbf{T}[n] \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{Q}^H[n-1]\mathbf{\Gamma}^{\frac{1}{2}}[n-1]\mathbf{d}^*[n-1] \\ \mathbf{d}^*[n] \end{bmatrix} = \\
&= \mathbf{T}[n] \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{p}[n-1] \\ \lambda^{\frac{1}{2}}\mathbf{v}[n-1] \\ \mathbf{d}^*[n] \end{bmatrix} \qquad (60)
\end{aligned}$$

which represents the time update recursion for $\mathbf{p}[n]$ and $\mathbf{v}[n]$.

It is important to notice from Equation (52), Equation (53) and Equation (60) that the elements of the matrix $\bar{\mathbf{Q}}[n]$ are never required to compute $\mathbf{w}[n]$, since they do not appear into the time update equations of $\bar{\mathbf{R}}[n]$ (53) or $\mathbf{p}[n]$ (60). Furthermore, the particular structure of matrix $\bar{\mathbf{R}}[n]$ and matrix $\mathbf{T}[n]$ are such that time update can be computed directly for $\widehat{\mathbf{R}}[n]$ by operating onto the following matrix

$$\breve{\mathbf{R}}[n] \triangleq \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{R}_1[n-1] \\ \tilde{\mathbf{X}}^H[n] \end{bmatrix} \quad \in \mathbb{C}^{M+P,M} \qquad (61)$$

with a reduced version of $\mathbf{T}[n]$ referred to as $\breve{\mathbf{T}}[n]$.

Similar considerations can be applied to $\mathbf{p}[n]$ where the time update can be computed by operating onto

$$\breve{\mathbf{p}}[n] \triangleq \begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{p}[n-1] \\ \tilde{\mathbf{d}}^*[n] \end{bmatrix} \quad \in \mathbb{C}^{M+P,1} \qquad (62)$$

which means that the elements of $\mathbf{v}[n]$ are never required. This brings also to the fact that, even though the matrices and the vectors grow in dimension as they are time updated, parts of them can be totally neglected. Since this happens both in Equation (61), where some zeros rows are neglected and in Equation (62), where $\mathbf{v}[n]$ is neglected too, it follows that only $\breve{\mathbf{R}}[n]$ and $\breve{\mathbf{p}}[n]$ are taken into account and they are always of the same dimensions.

The operations required to implement the Rank-P QRD-RLS Algorithm are summarized in Algorithm (2).

2.3 Simulation results obtained with the Multi-rank RLS algorithm

In infinite precision arithmetic the performance of the standard Multi-rank RLS algorithm (1) and of the Multi-rank QRD-RLS algorithm (2) are the same, since they attain the same analytic solution. Therefore the behavior of the Multi-rank QRD-RLS algorithm is analyzed in this paragraph, simulated in infinite precision arithmetic. Different values of Doppler shift frequency f_d are taken into account in order to evaluate the capability of the algorithm to recover Doppler shifts (see also Deliverable D17, [1]).

We consider a set of simulation parameters taken from IEEE 802.16a standard: $N_d = 200$ data sub-carriers, $N_p = 8$ pilot sub-carriers and $N_z = 56$ zeros sub-carriers, for a total of $N_{\mathrm{FFT}} = 256$ samples per OFDM symbol, without cyclic prefix. The signal bandwidth is 25 MHz, transmitted on a carrier frequency of 28 GHz. Furthermore, the propagation characteristics are such that they can be properly modeled as an AWGN channel with Doppler shift effect (we impose SNR = 0 dB in the simulations below) [10]. At the receiver, a standard Uniform Linear Array (ULA) array composed by $M = 8$ antennas is adopted.

As a first performance metric, we evaluate the relative residual error between the normalized beamformer weight vector $\frac{\mathbf{w}[n_0]}{\mathbf{w_1}[n_0]}$ (where $\mathbf{w_1}[n_0]$ is the first element of the vector) and its theoretical optimum value, that, in the case of one impinging signal in AWGN, is the steering vector $\mathbf{a}[\theta_0, \phi_0]$ in the DOA

λ	Rank	Doppler $[kHz]$	No. of Symb. for steady state	$\varepsilon_{\mathbf{w}}[n_0]$ $[\%]$	$\varepsilon_{\mathbf{w}}[n_0]$ [dB]
0.99	1	± 0	400	6.5	-11.82
0.99	1	± 1	400	214.2	3.31
0.99	4	± 0	46	1.4	-18.27
0.99	4	± 1	400	120.8	0.82
0.99	16	± 0	14	0.5	-22.86
0.99	16	± 1	200	12.4	-9.07
0.99	16	± 3	300	117.8	0.71
0.8	1	± 1	400	161.4	2.08
0.8	4	± 1	25	50.4	-2.97
0.8	16	± 1	15	10.6	-9.76
0.8	16	± 8	35	38.8	-4.11
0.8	16	± 9	36	57.4	-2.41

Table 1: Performance comparison for different ranks and forgetting factors of the Multi-rank QRD-RLS algorithms, in different Doppler shift conditions.

(θ_0, ϕ_0) of the HAP signal. The residual error is defined as

$$\varepsilon_{\mathbf{w}}[n_0] = \frac{\left|\left| \frac{\mathbf{w}[n_0]}{\mathbf{w_1}[n_0]} - \mathbf{a}[\theta_0, \phi_0] \right|\right|^2}{||\mathbf{a}[\theta_0, \phi_0]||^2} \tag{63}$$

and it is shown in Table 1, averaged over 100 Monte Carlo simulation runs with $n_0 = 1000$ iterations, and expressed both in logarithmic scale and percentage, for different Doppler rates, algorithm ranks and forgetting factors. Rank variation is obtained by simply using a different number of known sequences as reference signals.

The second performance metric is given by number of OFDM symbols needed to reach the steady state condition for the standard RLS algorithm. Fig. 2 shows how the residual error behaves in time using different rank updates.

It is evident that as the rank increases,

- the beamformer approaches more closely the optimum solution,
- the transient behavior is shortened,
- the residual error in steady state becomes lower and lower.

Furthermore, it is possible to observe from Table 1 that the Doppler shift becomes critical above a certain threshold (e.g., ± 9 kHz) and makes the beamformer unable to mimic the steering vector, whereas, below that threshold, a proper choice of the algorithm parameters allows compensation of Doppler shift.

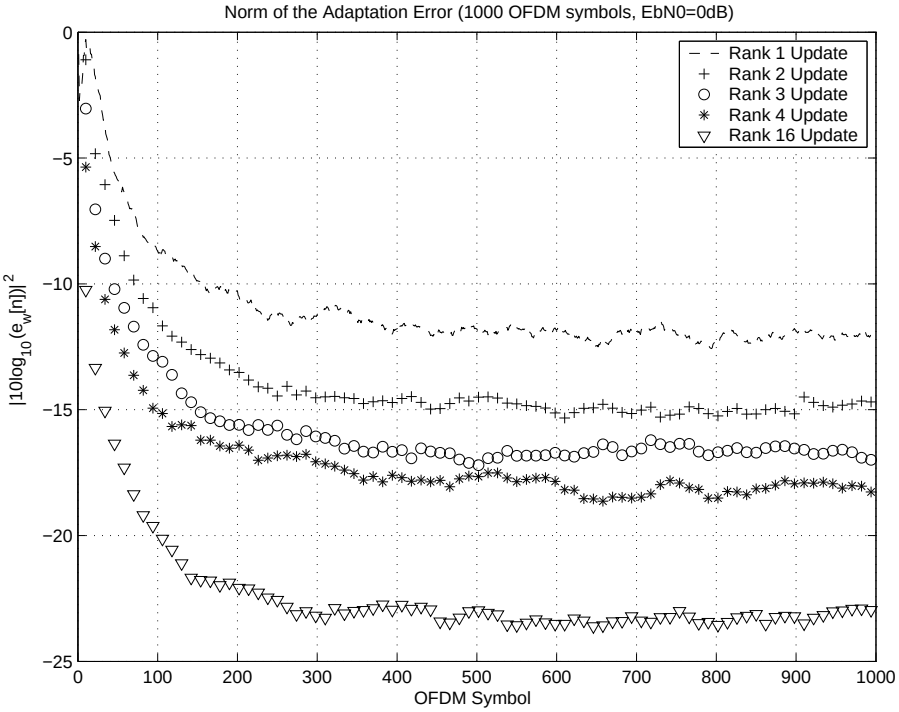


Figure 2: Residual error for different ranks, $f_d = 0.2$ kHz and forgetting factor $\lambda = 0.9$.

Algorithm 2 Rank-P QRD-RLS Algorithm

Initialize: $n \leftarrow 0$

Initialize: $\mathbf{R}_1[0] = \mathbf{0}_{M,M}$

Initialize: $\mathbf{p}[0] = \mathbf{0}_{M,1}$

repeat

$n \leftarrow n + 1$

receive $\mathbf{x}_p[n], \quad p = 1,\ 2,\ ...\ P$

receive $\mathbf{d}_p[n], \quad p = 1,\ 2,\ ...\ P$

build $\breve{\mathbf{R}}[n]$ according to Equation (61)

build $\breve{\mathbf{p}}[n]$ according to Equation (62)

for $p = 1 : M$ **do**

for $q = M + P : -1 : M + 1$ **do**

compute $c_{p,q}$ and $s_{p,q}$ according to Equation (55) and Equation (56) on the basis of $\left[\breve{\mathbf{R}}[n]\right]_{p,p}$ and $\left[\breve{\mathbf{R}}[n]\right]_{q,p}$

$\mathbf{u} \leftarrow \left[\breve{\mathbf{R}}[n]\right]_{p,:}$

$\left[\breve{\mathbf{R}}[n]\right]_{p,:} \leftarrow c_{p,q}\mathbf{u} + s_{p,q}\left[\breve{\mathbf{R}}[n]\right]_{q,:}$

$\left[\breve{\mathbf{R}}[n]\right]_{q,1:p} \leftarrow 0_{1,p}$

$\left[\breve{\mathbf{R}}[n]\right]_{q,p+1:\mathrm{end}} \leftarrow -s^*_{p,q}[\mathbf{u}]_{1,p+1:\mathrm{end}} + c_{p,q}\left[\breve{\mathbf{R}}[n]\right]_{q,p+1:\mathrm{end}}$

$\mathbf{u} \leftarrow [\breve{\mathbf{p}}[n]]_{p,1}$

$[\breve{\mathbf{p}}[n]]_{p,1} \leftarrow c_{p,q}\mathbf{u} + s_{p,q}[\breve{\mathbf{p}}[n]]_{q,1}$

$[\breve{\mathbf{p}}[n]]_{q,1} \leftarrow -s^*_{p,q}\mathbf{u} + c_{p,q}[\breve{\mathbf{p}}[n]]_{q,1}$

end for

end for

$\widehat{\mathbf{R}}[n] \leftarrow \left[\breve{\mathbf{R}}[n]\right]_{1:M,1:M}$

$\mathbf{p}[n] \leftarrow [\breve{\mathbf{p}}[n]]_{1:M,1}$

compute $\mathbf{w}[n]$ by solving the system (47) via backward substitution

apply beamforming using $\mathbf{w}[n]$

until there are no further samples

3 Analysis of multi-rank RLS algorithms in finite precision arithmetic

While there is no difference in the algebraic solution of the standard Multi-rank RLS algorithm shown in Algorithm (1), the Multi-rank RLS presented in [2], and the Multirank QRD-RLS algorithm shown in Algorithm (2), as well as in their ideal performance discussed in subsection 2.3, the situation is dramatically different, whenever a real-time implementation with finite precision arithmetic is addressed.

In this section we investigate the suitability of the algorithms discussed in Section 2 to the implementation on a finite precision device. Our investigation will address the following main points:

1. **Computational complexity of the algorithms**
2. **Choice of the machine wordlength**
3. **Algorithms performance in finite precision arithmetic**

The simulation of finite precision arithmetic has been made by using a floating point 32-bit version of the algorithms, written in Matlab® language.

As for the Multi-rank RLS algorithm, we mainly refer hereafter to the low-complexity implementation proposed in [2] (see Section 2.2.2). However, we can anticipate that the implementation in [2] will result in instability in some conditions, while the QRD approach is always stable.

3.1 Analysis of computational complexity

In Tables 2–4 the computational complexity of each algorithm discussed in Section 2 is summarized, for rank P, M beamformer weights (i.e., M antenna sensors) and N_{bit} bits assigned as numerical representation wordlength.

Standard Multi-rank RLS Algorithm (1)				
Step no.	Operation	No. of real products	No. of real sums	No. of real divisions
1	$\mathbf{k}[n]$	$8M^2+6M$	$8M^2-1$	$2MN_{bit}$
2	$\mathbf{R_{xx}}[n]$	$10M^2$	$8M^2-2M$	
3	$\mathbf{e}[n]$	$4M$	$4M$	
4	(1+2+3)*P			
5	$\mathbf{w}[n]$	$32M^2+24M$	$32M^2$	
Tot.		$(18P+32)M^2+(10P+24)M$	$(16P+32)M^2+2MP-P$	$2PMN_{bit}$

Table 2: Computational effort for the standard Multi-rank RLS algorithm (1).

Multi-rank RLS Algorithm in [2]				
Step no.	Operation	No. of real products	No. of real sums	No. of real divisions
1	$\mathbf{k}[n]$	$8M^2+6M$	$8M^2-1$	$2MN_{bit}$
2	$\mathbf{e}[n]$	$4M$	$4M$	
3	$\mathbf{R_{xx}}[n]$	$10M^2$	$8M^2-2M$	
4	$\mathbf{w}[n]$	$4M$	$4M$	
5	(1+2+3+4)*P			
Tot.		$P(18M^2+14M)$	$P(16M^2+6M-1)$	$2PMN_{bit}$

Table 3: Computational effort for the Multi-rank RLS algorithm proposed in [2].

In order to give an idea of the number of operations performed in a second, we can choose:

- $M=8$, number of sensors
- rank $P=4$

Multi-rank QRD-RLS Algorithm (2)					
Step no.	Operation	No. of real products	No. of real sums	No. of real divisions	No. of SQRT
1	$\mathbf{R}[n]\sqrt{\lambda}$	M^2+M			
2	$\mathbf{p}[n]\sqrt{\lambda}$	$2M$			
3	Givens rot.	$PM(30+6M)$	$PM(18+4M)$	PM	PMN_{bit}
4	$\mathbf{w}[n]$ via backward subst.	$M^2/2+11M/2$	$M^2/2+5M/2$	$2M$	
Tot.		$(\frac{3}{2}+6P)M^2+$ $(\frac{17}{2}+30P)M$	$(\frac{1}{2}+4P)M^2+$ $(\frac{5}{2}+18P)M$	$(P+2)M$	PMN_{bit}

Table 4: Computational effort for Multi-rank QRD-RLS algorithm (2).

- wordlength $N_{bit}=16$
- OFDM symbol duration equal to $8\ \mu$s

Concerning the mathematical operations performed by the device, let us suppose that:

- The device performs a real sum and a real product for each cycle
- The device performs a real division in a number of cycles equal to the wordlength

In these conditions, the device that implements the algorithm (1) should perform 7168 operations for each OFDM symbol, which requires a clock frequency of $f=1.79$ GHz. In the algorithm [2], the device should perform 5056 operations for each OFDM symbol, which requires a clock frequency of $f=1.26$ GHz. Finally, the device that implements the Multirank QRD-RLS algorithm is requested to compute just 2660 operations for each OFDM symbol, which requires a clock frequency of $f=665$ MHz, which is almost half the one obtained with the algorithm in [2].

On the other hand, if a rank-1 implementation is selected for a Single-Carrier communication [1], i.e. $P=1$, the number of operations and clock frequency requested by the three algorithms are summarized in the table below:

	Standard rank-1 RLS (1)	Rank-1 RLS in [2]	Rank-1 QRD-RLS (2)
No. of operations per OFDM symbol	3472	1264	788
Clock frequency [MHz]	434	158	98.5

3.2 Wordlength analysis

When we approach the implementation of an algorithm on a finite precision arithmetic device, we have to deal with the problem of determining the number of bits to represent each data value in the algorithm (N_{bit}).

After having determined the minimum number of bits to accurately represent the data, the device must be arranged in order to avoid overflow and truncation effects. Evidently, the device must have at disposal the necessary number of bits per dataword.

The choice of the correct wordlength can be determined either analytically, by studying the data input range and the critical paths, or by an empirical way, simulating a quantized version of the algorithm, in order to directly find the correct number of bits which occur to describe the integer and the fractional part of each value. Both these strategies have been used to find the best value N_{bit}.

If we are dealing with data with modulus strictly lower than 1, we can decide to process the signals taking into account only their fractional part. It must be multiplied by $2^{(N_{bit}-1)}$ and then represented on a congruous number of bits $(N_{bit}-1)$, actually getting rid of the integer part.

If, on the other hand, we are dealing with signals which are not limited in the range $(-1,+1)$, it is important to evaluate the maximum dynamics to be described in finite precision arithmetic, either

to rescale all the values so as to obtain data with modulus lower than 1, as before, or to provide a description with a congruous number of bits also for the integer part of the data.

These preliminary considerations do not take into account possible overflow problems, discussed in the following.

3.2.1 Analytical study

First, it is important to choose a method to represent the data: it could be a modulus and sign representation or 2's complement description, for the fixed point methods, or a floating point family method. We choose a 2's complement description, which is the most usual way to deal with such problems. Integer and fractional parts of the data will be quantized separately. This description is compatible with the use of the signed library in the VHDL language. In this description, having $N_{bit} = b+1$ bits at one's disposal, 1 bit can be used for the sign and the other b bits for the data quantization.

Furthermore, the statistical investigation of the values assumed by the data allows convenient modeling of description range, to prevent the overflow problem.

Then, the critical path must be identified and computed. The critical path is the sequence of operations that must be completed on schedule for the entire calculation to be completed on schedule. It is the longest duration path through the workplan.

Recalling that each multiplication causes a truncation or a rounding off of the LSBs (Less Significant Bits), it is necessary to know the number of multiplications in order to implement a well-conditioned algorithm. Besides, since each sum potentially causes an overflow, it is necessary to know the number of sums in order to best scale the input data and to prevent overflow.

For the algorithms under investigation, the weight vector is the output data value which passes through the highest number of operations at each iteration. For the Multi-rank RLS proposed in [2], it is possible to compute the number of bits necessary to account for the operation of the critical path from Table 3, as

$$b_{op} = \lceil \log_2(P(16M^2 + 6M - 1)) \rceil = \lceil 12.013 \rceil = 13 \,\text{bits} \tag{64}$$

Besides, the smallest input data is given by the adaptation error which can reach values near to 10^{-11}. This means that, to correctly represent these values, $b_{data} = \lceil \log_2 \left(\frac{1}{10^{-11}} \right) \rceil = \lceil 36.54 \rceil = 37$ bits are needed. Therefore, the system that implements the Multi-rank RLS [2] should represent the data with $N_{bit} = b_{op} + b_{data} = 13 + 37 = 50$ bits, which is a quite demanding requirement.

For implementation of the Multi-rank QRD-RLS algorithm, we note the following:

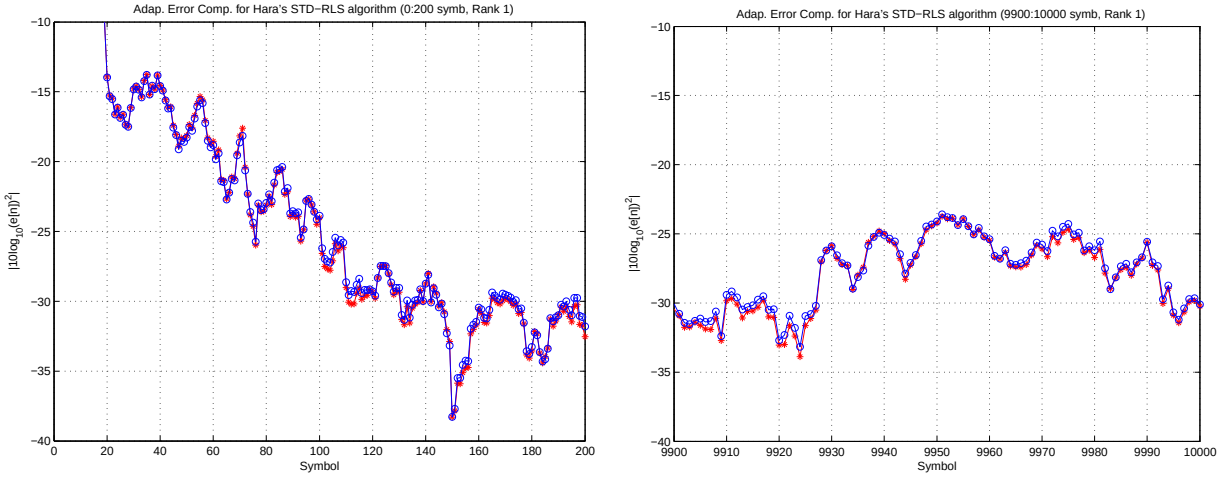
1. The QRD-RLS algorithm does not deal with the adaptation error, since it exploits other quantities to evaluate the weight vector.
2. It has been demonstrated in [11] that the quantization error which propagates in the algorithm is exponentially stable and has terms which decay proportionally to the time index.
3. The calculation which is most sensitive to quantization is the evaluation of the parameters c and s in the Givens rotations. Since these operations need a greater number of bits to represent the integer part of the data, during the simulation and then in the VHDL implementation, the integer part of the data elaborated inside the Givens rotation block are described using double wordlength.

An analytical way to compute the wordlength for the finite precision implementation of an algorithm such as the Multi-rank QRD-RLS one has been proposed in [11]. The data most likely to be subject to overflow problems are those of the elements contained in matrix $\mathbf{R}_{\mathbf{xx}}[n]$ and in vector $\mathbf{r}_{\mathbf{x}d}[n]$. Let x_{MAX} be the maximum magnitude of the signal $x_p[n]$ and d_{MAX} the maximum magnitude of the desired signal $d_p[n]$; it has been demonstrated in [11] that

$$\left| \mathbf{R}_{\mathbf{xx}}[n] \right| < \frac{|x_{\text{MAX}}|}{\sqrt{1-\lambda}} \tag{65}$$

and

$$\left| \mathbf{r}_{\mathbf{x}d}[n] \right| < \frac{|d_{\text{MAX}}|}{\sqrt{1-\lambda}}, \tag{66}$$



(a) First symbols (b) Last symbols

Figure 3: Residual error of the classic Multirank RLS, with infinite precision implementation (asterisks) and quantized implementation over 50 bits (circles)

where $|\mathbf{A}|$ is defined as the norm-1 of matrix $\mathbf{A}$. These upper bounds allow evaluation of the wordlength necessary to keep the quantization error below an acceptable threshold. To evaluate d_{MAX} and x_{MAX}, $d_p[n]$ is modeled as a random variable with zero mean and variance $\sigma_d = 1$, so that it is reasonable to assume $|d_{\text{MAX}}| = 3\sigma_d = 3$ and $|\mathbf{r}_{\mathbf{x}d}[n]| < 30$, with $\lambda = 0.99$. As a consequence, for the simulated system, $\sigma_x = \sqrt{2}$, $|x_{\text{MAX}}| = 3\sqrt{2}$ and $|\mathbf{R}_{\mathbf{xx}}[n]| < 42.5$.

Therefore, we choose to describe the integer part of the data with 7 bits (1 for the sign and 6 for the absolute value), so that we correctly describe numbers in the range $[-(2^6-1),(2^6-1)]$, whereas it is known that this algorithm does not suffer from unbounded output problems. However, owing to the critical step represented by the computation of the square roots, data involved in these operations are represented with double wordlength values.

In order to set the wordlength for the fractional part of the data so as to guarantee the stability of the algorithm, it is possible to proceed via an empirical evaluation as proposed in [12]. It has been proved in [12] that the two methods (analytical and empirical) bring the same result.

3.2.2 Empirical validation: performance in finite precision arithmetic

It is possible to show that the finite precision implementation of the Multirank RLS algorithm in [2] with such a number of bits has approximately the same performance in its infinite precision implementation. This is shown in fig. (3), with the normalized residual error $\varepsilon_{\mathbf{w}}[n]$ plotted on a logarithmic scale. The propagation conditions set for the simulations discussed in this subsection are $f_d = 0$ and SNR = 0 dB.

Nonetheless, the *difference between the residual errors* for the infinite and finite precision implementation is -65.8 dB at the 100-th symbol, increasing to -57.2 dB at the 10000-th symbol. This means that, despite the long wordlength, the RLS implementation [2] exhibits instability.

In the following, the wordlengths of the integer part and fractional part will be separately studied:

- β_I bits are dedicated to the integer part,
- β_F bits are dedicated to the fractional part.

On the basis of the last result discussed in the previous subsection, a finite precision implementation of the Multi-rank RLS algorithm [2] has been evaluated, representing the data over N_{bit} bits, composed of:

- 1 bit for the sign,

Rank	Mean difference	Max difference
1	-55.39	-46.71
3	-55.36	-46.01
5	-54.456	-43.80

Table 5: Multi-rank RLS algorithm [2]: difference between the residual errors for infinite and finite precision implementation.

β_I	**classic RLS**		**QRD-RLS**	
No. of bits	Mean difference	Max. difference	Mean difference	Max. difference
$1+1$	-18.425	-17.081	-6.2233	1.0304
$2+1$	-35.491	-31.392	-9.9268	-5.1551
$3+1$	-46.322	-40.208	-14.044	-11.228
$5+1$	-49.732	-44.214	-80.366	-77.179
$6+1$	-47.495	-44.211	-83.681	-79.498
$7+1$	-51.608	-42.601	-87.528	-82.78

Table 6: Differences between residual errors in infinite and finite implementation as a function of β_I, with $\beta_F = 15$.

- $\beta_I = 6$ bits,
- $\beta_F = 15$ bits.

In these conditions the *difference between the residual errors* for the infinite and finite precision implementation, averaged over 10 Monte Carlo simulations, has been reported in Table 5, as a function of the rank. These results, though quite good, show a worsening behavior when the rank increases. With $N_{bit} = 22$ bits, the algorithm suffers from quantization error and tends to become unstable.

In order to observe the differences between the performances of the two algorithms Multi-rank RLS [2] and Multi-rank QRD-RLS, in Figure 4 we represent the normalized residual error in dB obtained for the two algorithms as a function of time. The algorithms work with $\lambda = 0.8$, $SNR = 0$ dB, rank $P = 16$ and 150 simulated symbols, using $\beta_F = 15$ bits for the fractional part and β_I variable for the integer part. The Multirank RLS algorithm [2] is indicated as *STD-RLS*, while the Multirank QRD-RLS algorithm is indicated as *QRD-RLS*.

Table 6 summarizes the differences between infinite and finite implementation obtained with the two algorithms using $\beta_I = 1$ to 7 bits for the integer part.

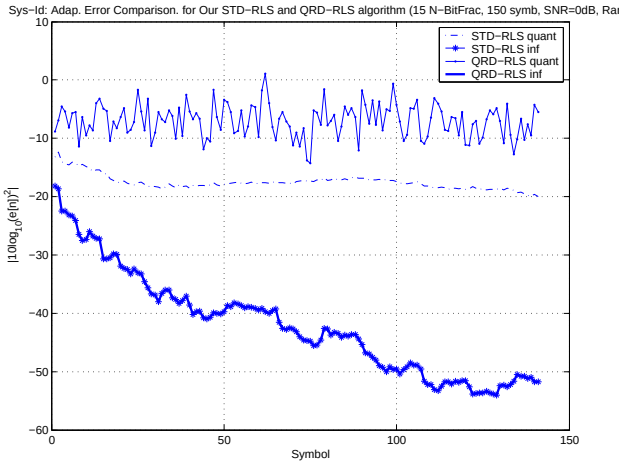
From Table 6 and Figure 4 it can be noticed that, for a low number of bits representing the integer part of the data, the Multirank QRD-RLS algorithm, although suffering the saturation, is *stable*, since the most critical values, i.e., the output of the square roots in the Givens rotations, are treated with a double number of bits for the integer part. The Multi-rank RLS algorithm [2], instead, suffers less the saturation problems for the first samples, but, as time goes on, the involved matrices tend to rapidly expand in numerical value and the algorithm becomes *unstable*.

With $\beta_I = 6$ bits representing the integer part, the Multirank QRD-RLS algorithm performs well and does not suffers saturation problems.

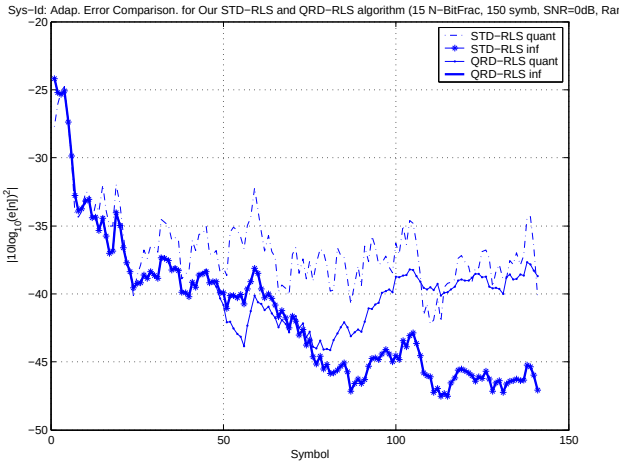
For the the partial wordlength for the fractional part, β_F, in Table 7 we show the differences between infinite and quantized implementation of the algorithms, obtained with $\beta_I = 6$ bits and variable β_F. It can be observed that the quantized implementation of the Multirank RLS algorithm [2] is stable with such a low number of symbols, but, as long as the number of simulated symbols increase, it becomes unstable.[2]

Finally, Figure 5 shows the normalized residual errors of the two algorithms, obtained for different wordlengths; it is observed that the quantized version of the QRD-RLS algorithm differs less from its infinite implementation than the Multi-rank RLS algorithm [2].

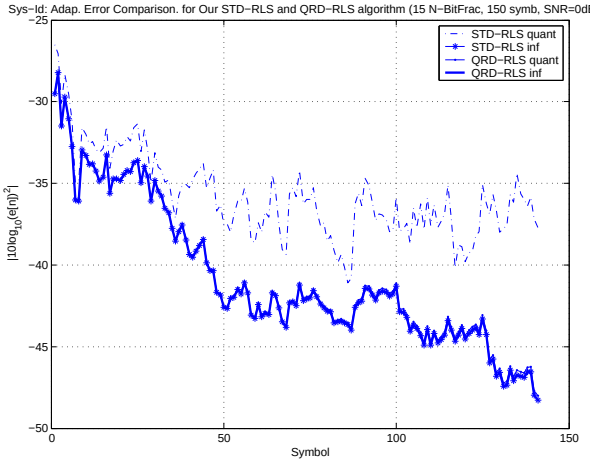
[2] A common solution to such a problem is to periodically reset the memory matrices.



(a) Bits for integer part: 1



(b) Bits for integer part: 5

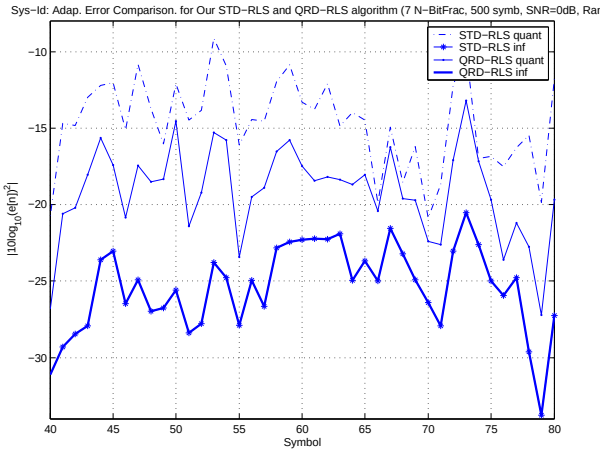


(c) Bits for integer part: 6

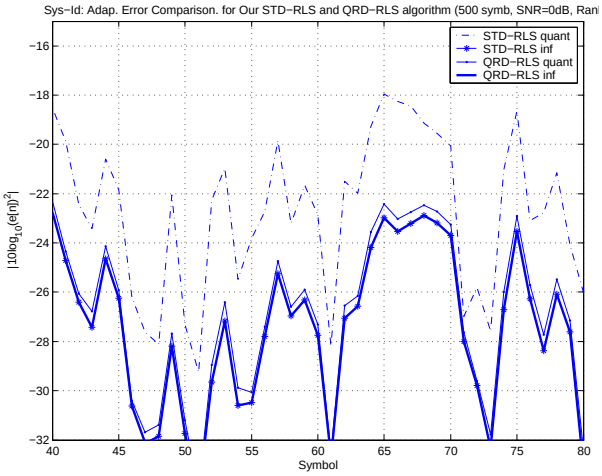
Figure 4: Normalized residual error for different implementations of the Multi-rank RLS algorithm, for different number of quantization bits for the integer part of the data, β_I.

β_F	**Multi-rank RLS [2]**		**Multi-rank QRD-RLS**	
No. of bits	Mean difference	Max. difference	Mean difference	Max. difference
7	-15.645	-8.9706	-23.402	-16.785
8	-18.461	-10.245	-30.779	-22.062
10	-24.753	-17.485	-43.416	-34.988
11	-28.586	-21.433	-48.789	-41.779
12	-32.175	-21.219	-55.305	-44.173
13	-37.042	-30.812	-61.936	-55.377
15	-46.833	-38.553	-72.478	-65.041
18	-64.959	-56.349	-91.456	-81.621

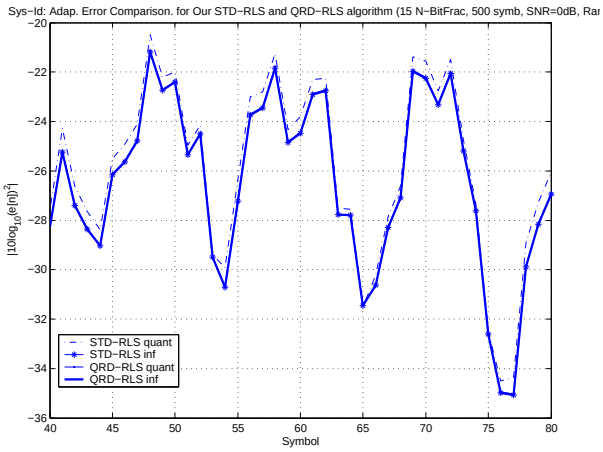
Table 7: Differences between residual errors in infinite and quantized implementation of the algorithms, as a function of β_F, with $\beta_I = 6$ bits.



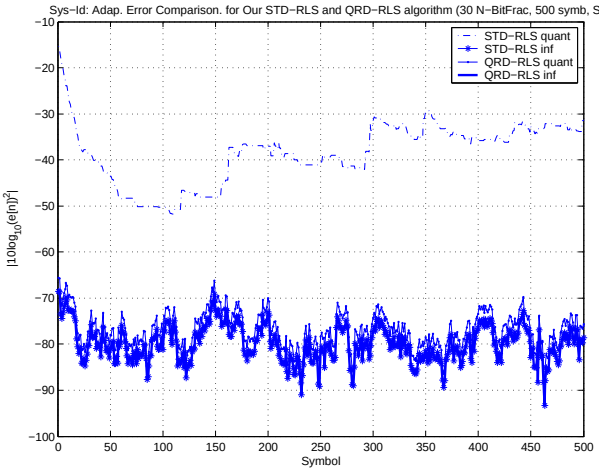
(a) Bit for integer part: 11 (10+sign), bit for fractional part: 7



(b) Bit for integer part: 11 (10+sign), bit for fractional part: 11



(c) Bit for integer part: 11 (10+sign), bit for fractional part: 15



(d) Bit for integer part: 4 (3+sign), bit for fractional part: 30

Figure 5: Normalized residual errors of the two algorithms, obtained for different wordlengths $N_{bit} = 1 + \beta_I + \beta_F$.

Performances of Multirank QRD-RLS in High Noise Conditions				
Rank	No. of operations	Clock frequency [MHz]	Residual error floor 1500-th symbol [dB]	No. of symb. for rank 1 steady-state
1	788	98.5	-27.29	270
2	1412	176.5	-33.16	105
3	2036	254.5	-36.62	70
4	2660	332.5	-38.95	50
5	3284	410.5	-41.21	41
6	3908	488.5	-42.61	34
7	4532	566.5	-43.88	30
8	5156	644.5	-45.09	26
9	5780	722.5	-46.26	24
10	6404	800.5	-47.05	22
11	7028	878.5	-48.04	21
12	7652	956.5	-48.79	19
13	8276	1034.5	-49.61	17
14	8900	1114.5	-50.03	16
15	9524	1195.5	-50.93	15
16	10148	1468.5	-51.38	14

Table 8: Costs/benefits of QRD-RLS Multirank Algorithm

To conclude this section, we propose in Table 8 a performance comparison for different ranks of the Multirank QRD-RLS algorithm, in terms of

- number of operations needed,
- clock frequency,
- normalized residual error in steady-state,
- number of transient symbols to reach the steady-state error level attained by the rank 1 version of the algorithm.

The choice of the device parameters (i.e., number of quantization bits, clock frequency) derives from the trade-off between system performance (i.e., residual error and number of transient symbols) and implementation costs (i.e., number of operations and clock frequency).

A possible solution, that gives a good trade-off between costs and benefits and that we consider in our next development, is a device that implements a rank-4 QRD-RLS algorithm, quantized on 7 bits for the integer part ($\beta_I = 6 + 1$ is for the sign) and $\beta_F = 13$ bits for the fractional one. This solution can be implemented on a device working on $N_{bit} = 20$ bits wordlength at least, with a clock frequency of 665 MHz.

3.2.3 Wordlength in the computation of the Givens rotations

For the computation of the Givens rotations, a specific block is dedicated to compute parameters $c_{p,q}$ and $s_{p,q}$. Inside this block a longer wordlength is necessary, because of the presence of square modules in the denominator of Equation (55); this leads the represented values to have a double dynamic compared to that used for other computations. Thus, the number of bits is increased to $\beta'_I = 12$ bits for the integer part, while the fractional part may not to be changed.

Note that this increased wordlength is used just within the hardware block described in §4.2.5, which computes Givens rotations, where we need $N'_{bit} = 26$ bits totally. Indeed, after the computation of the square modules, there is a square root operation that bring the values back to the previous dynamic. Consequently, outside the Gives rotation's block, the wordlength can become again $N_{bit} = 20$ bits.

4 VHDL implementation

This chapter describes a general implementation of the Multi-rank QRD-RLS algorithm in VHDL language. The main aims are

- a description of the data in finite precision arithmetic, which leads to quantization errors,
- a description of the blocks that must be implemented for computing the algorithm, together with those necessary for computing complex values.

The last subsection will shows that the results obtained in VHDL are coherent with those simulated with Matlab®.

4.1 Data description in VHDL components

Recall that the input data are quantized on N_{bit} bits and represented in 2's complement, so the MSB (Most Significant Bit) is used for the sign. Then the value is represented with $N_{bit} - 1$ bits, where this number can be divided into β_F bits for the fractional part and β_I bits for the integer part. The β_F and β_I parameters have been derived from Matlab® simulations, shown in Section 3.

It is important to notice that the data must be integer values, converted into bit vectors. Each real input must be shifted to become an integer, so each value is multiplied by 2^{β_F}. The real and imaginary part of each complex value are treated separately, as two independent real numbers.

Figure 6 shows the quantization procedure applied to a complex, fractional number $X = a + jb$, where some precision is lost.

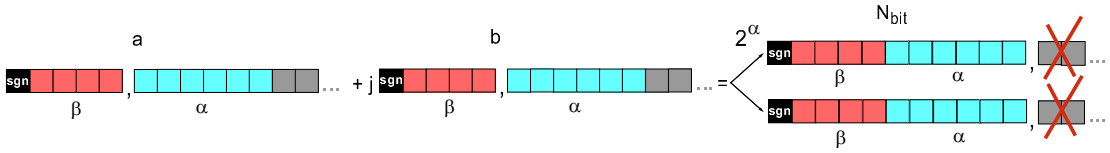


Figure 6: Shifting by $\alpha = \beta_F$ bits and truncation. $\beta = \beta_I$.

4.2 Developed blocks for the Multi-rank QRD-RLS algorithm

The overall architecture has been designed to **maximize the throughput of the device**, despite the cost of a larger starting delay time. For this reason, *everything is pipelined,* resulting in a *flat architecture* with a great number of registers, useful to reduce the critical paths, correctly synchronize the data and reset the components if necessary.

4.2.1 Multirank_QR.vhd

This is the main block of the algorithm, it connects the blocks that compute the QR update seen in Equation (52) and (53) and the blocks that compute the backward substitution for solving the system in Equation (47), giving out the weight vector calculated by the algorithm (see Figure 7).

This block receives the general parameters of the algorithm:

- `NBIT`: wordlength N_{bit}, used for the description of the numbers in 2's complement.
- `ALPHA`: number of bit dedicated to the fractional part (β_F).
- `NSENS`: number of antennas M.
- `NRANK`: rank P.
- `TPD, TPD1, TPD2`: time delay of a register, an addition, a multiplication, respectively.

The blocks *QR_upd* and *BackSubs* are described respectively in subsections 4.2.2 and 4.2.4.

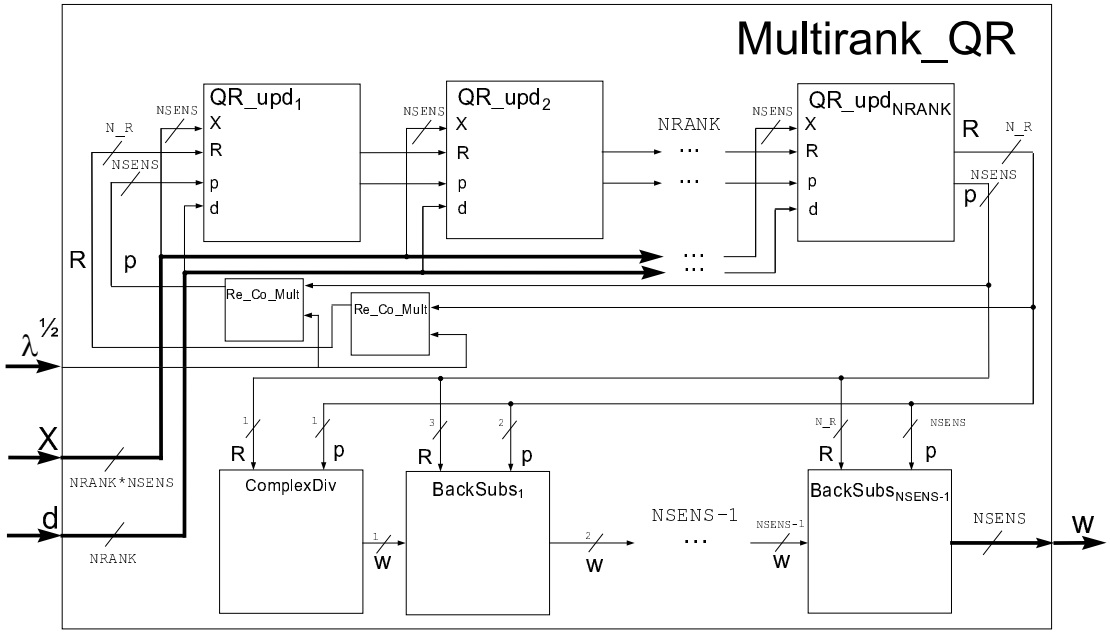


Figure 7: Scheme of Multirank_QR.vhd

4.2.2 QR_upd.vhd

This component performs the QR update seen in Equation (61) and (62). The parameter N_R is the number of elements of the upper triangular matrix $\check{\mathbf{R}}[n]$. This number depends on M, that is the sensor number (NSENS), and it can be determined by the numerical series:

$$\mathtt{N\_R} = \sum_{k=1}^{M} k = \frac{M(M+1)}{2}. \tag{67}$$

The scheme of the block is represented in Figure 8, it contains M *Giv_Rot* blocks (Section 4.2.3). These blocks work in series.

The inputs of the block are:

- $\mathbf{X}$: the data received by the system
- $\mathbf{d}$: the desired signal
- $\mathbf{R}$: the matrix $\check{\mathbf{R}}[n-1]$ from the previous step
- $\mathbf{p}$: the vector $\check{\mathbf{p}}[n-1]$ from the previous step

The outputs of the block are:

- $\mathbf{R}$: the matrix $\check{\mathbf{R}}[n]$ necessary for the next update
- $\mathbf{p}$: the vector $\check{\mathbf{p}}[n]$ necessary for the next update

4.2.3 Giv_Rot.vhd

This component performs the Givens rotations on the i-th row of matrix $\check{\mathbf{R}}[n]$, taking into account the corresponding elements of $\check{\mathbf{p}}[n]$, $\mathbf{x}[n]$ and $d[n]$.

This block works with one row of $\check{\mathbf{R}}[n]$, so it receives the parameter NROT that indicates the number of rotations that must be performed; this number corresponds to the number of elements of the selected row of $\check{\mathbf{R}}[n]$, that is the same size as $\mathbf{x}[n]$.

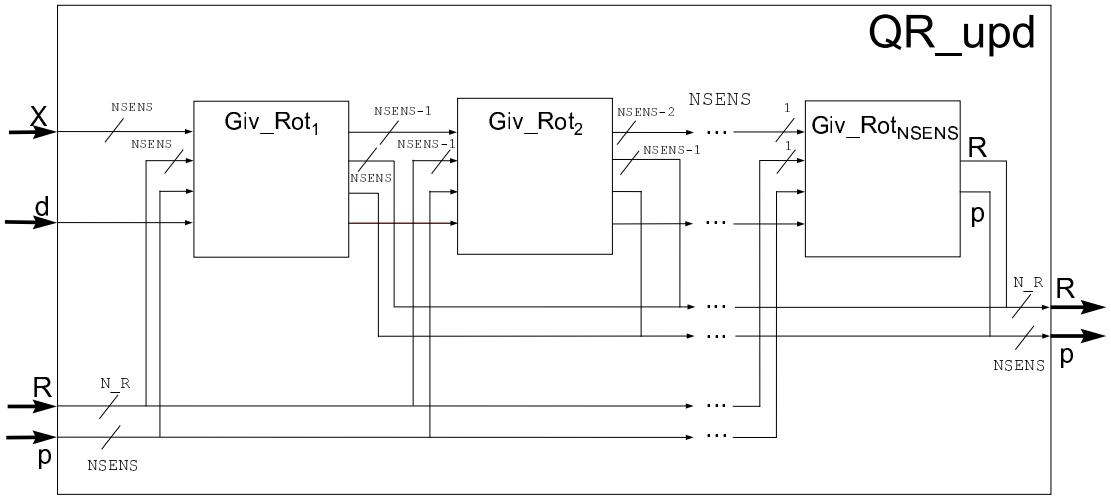


Figure 8: Scheme of QR_upd.vhd

As the data arrive, the block *C_S_Calc*, described in Section 4.2.5, calculates the parameters c and s as shown in Equation (73) and Equation (74). The operations performed on every element are done simultaneously, as it can be seen in Figure 9.

The inputs of the block are:

- $\mathbf{X}$: the data received by the system
- $\mathbf{d}$: the desired signal
- $\mathbf{R}$: the corresponding row of the matrix $\check{\mathbf{R}}[n]$
- $\mathbf{p}$: the corresponding element of the vector $\check{\mathbf{p}}[n]$

The outputs of the block are:

- $\mathbf{X}$: the modified data received by the system
- $\mathbf{d}$: the modified desired signal
- $\mathbf{R}$: the modified row of the matrix $\check{\mathbf{R}}[n]$
- $\mathbf{p}$: the modified element of the vector $\check{\mathbf{p}}[n]$

4.2.4 BackSubs.vhd

This block operates the backward substitution. The backward substitution is the way of resolving the system in Equation (47) with less computational effort. Let us define the system:

$$\begin{cases} \mathbf{R}_{1,1}[n]\mathbf{w}_1[n] + \mathbf{R}_{1,2}[n]\mathbf{w}_2[n] + \ldots + \mathbf{R}_{M,M}[n]\mathbf{w}_M[n] &= \mathbf{p}_1[n] \\ \mathbf{R}_{2,2}[n]\mathbf{w}_2[n] + \ldots + \mathbf{R}_{M,M}[n]\mathbf{w}_M[n] &= \mathbf{p}_2[n] \\ \vdots & \\ \mathbf{R}_{M,M}[n]\mathbf{w}_M[n] &= \mathbf{p}_M[n] \end{cases} \tag{68}$$

The first step of the backward substitution is

$$\mathbf{w}_M[n] = \frac{\mathbf{p}_M[n]}{\mathbf{R}_{M,M}[n]}, \tag{69}$$

computed by the *ComplexDiv* block present in the *Multirank_QR* block, as showm in Figure 7.

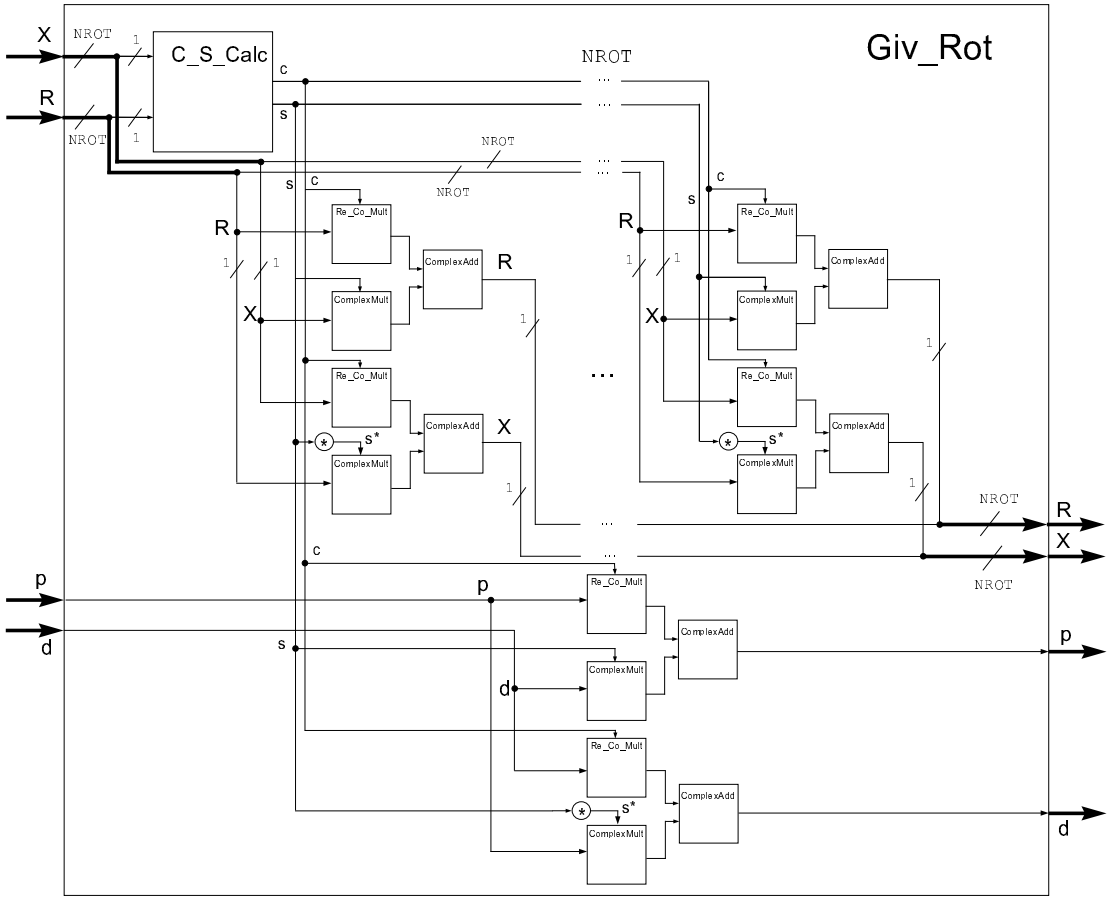


Figure 9: Scheme of Giv_Rot.vhd

The computed weight vector $\mathbf{w_M}[n]$ is given as input to the first *BackSubs* block, that implements the operation:

$$\mathbf{w}_{M-1}[n] = \frac{\mathbf{p}_{M-1}[n] - \mathbf{R}_{M-1,M}[n]\mathbf{w_M}[\mathbf{n}]}{\mathbf{R}_{M-1,M-1}[n]}, \tag{70}$$

obtained from

$$\mathbf{R}_{M-1,M-1}[n]\mathbf{w}_{M-1}[n] + \mathbf{R}_{M-1,M}[n]\mathbf{w}_M[n] = \mathbf{p}_{M-1}[n] \tag{71}$$

The output of each block is the weight vector for to the next block, so that the equation solved in the next block is:

$$\mathbf{R}_{M-2,M-2}[n]\mathbf{w}_{M-2}[n] + \mathbf{R}_{M-2,M-1}[n]\mathbf{w}_{M-1}[n] + \mathbf{R}_{M-2,M}[n]\mathbf{w}_M[n] = \mathbf{p}_{M-2}[n]. \tag{72}$$

4.2.5 C_S_Calc.vhd

This component calculates the parameters c and s that are used in the Givens rotations.

The expressions calculated by this block are the ones reported in Equation (55) and Equation (56), but the way these operations are computed is different from the theoretical case. $\mathbf{R}[n]$ has always the same dimensions ($M \times M$) because the zeros are not computed and $\mathbf{x}[n]$ is not considered as a row of $\mathbf{R}[n]$, but as a separate vector.

Thus, the equations employed in the practical implementation are:

$$c[i] = \frac{|R[i,i]|}{\sqrt{|R[i,i]|^2 + |x[i]|^2}} \tag{73}$$

$$s[i] = \frac{R[i,i]}{|R[i,i]|} \frac{x^*[i]}{\sqrt{|R[i,i]|^2 + |x[i]|^2}} \tag{74}$$

These operations need a higher number of bits dedicated to the integer part of the data. That is due to the squares calculated in the denominators of $c[i]$ and $s[i]$. The square root takes the values back to the usual range, so the output data can have the same number of bits used before.

The choice of the number of bits used for the integer part of the data inside this block is the double of the one used outside the block. The number of bits dedicated to the fractional part remains the same.

4.2.6 Blocks for complex operations

The blocks listed hereafter have been developed for the operations with complex numbers. They are general and can be used for any complex calculations in VHDL.

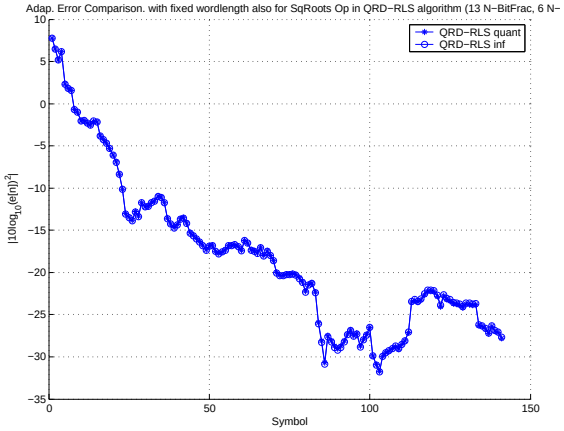
The real part and the imaginary part of each complex number are treated separately through parallel vectors and connected to the block used for the operation.

However, while most blocks subdivide complex operations in elementary additions and multiplications, this cannot be done with the division and the square root operations. The implementation of the blocks that calculate the division or the square root requires the knowledge of the device that will be used for the synthesis of the VHDL code. The blocks that implement the divisions use the type of variable `real`, which could be used only by components of higher level, like a DSP. The presence of this type of variable does not permit synthesis at a lower level. The square root operation has been divided in simple operations through the Newton-Raphson method, but it contains a division anyway, so this component is subjected to the same problem explained before. The problem of the square root can be solved using a Look Up Table (LUT), where all possible results could be memorized; this way is practicable when the wordlength used for representing the input number is small enough to permit the creation of a medium-size LUT.

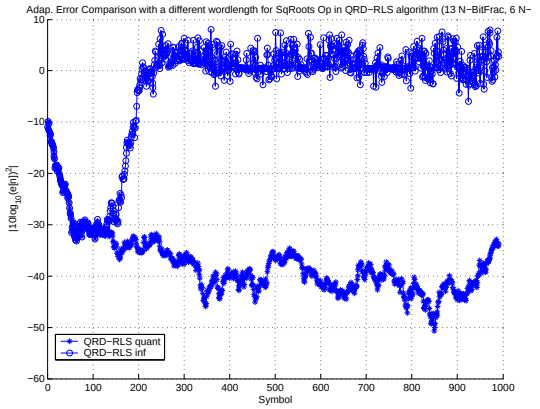
Our empirical study of the data wordlength has shown that, while 13 bits have to be used for the fractional parts of the generic data, just one bit is necessary to represent the integer parts of the square root outputs. This is due to the fact that this quantization error becomes negligible. The wordlength for the integer part has been found to be 6 bits, but for the radicand data it can be reduced to 5 bits, as shown in Figure 10. We believe that a good trade-off between performance and space saving on the device is to use $5+1$ bits to represent the integer part and 2 bits for the fractional part of the LUT. Thus, we must implement a LUT containing $2^8 = 256$ words, each 8 bits long.

The blocks implemented for complex operations are the following:

- *Reg.vhd* - Register
- *Reg_N.vhd* - Register with a parametric delay
- *ComplexAdd.vhd* - Adder for two complex numbers
- *ComplexMult.vhd* - Multiplier for two complex numbers
- *Re_Co_Mult.vhd* - Multiplier of a real number by a complex number
- *SqAbs.vhd* - It calculates the square absolute value of a complex number
- *ComplexDiv.vhd* - Divider for two complex numbers
- *C_on_R_Div.vhd* - Divider of a complex number by a real number
- *RealDiv.vhd* - Divider for two real numbers
- *Sq_Root.vhd* - It calculates the square root of a real number
- *Reg.vhd* - Register
- *Reg.vhd* - Register
- *Reg.vhd* - Register



(a) Total bit: (1+6+13), bit for the radicands: (1+6+13)



(b) Total bit: (1+6+13), bit for the radicands: (1+4+13)

Figure 10: Differences between residual errors for different wordlength for the radicand data.

4.3 VHDL validation

The number of bits used for representing the values has been chosen according to the results discussed in Section 3. We found out that a good trade-off between computational effort and quality of the results was $N_{bit} = 20$, subdivided in:

- 1 bit dedicated to the sign
- 6 bits dedicated to the integer part (β_I)
- 13 bits dedicated to the fractional part (β_F)

The results of the hardware implementation follow the simulated ones, as shown in Figure 11, thus demonstrating that the hardware implementation works correctly.

Another important issue is to estimate the delay of the outputs. The simulations use the following parameters: rank $P = 4$ (number of pilots), $M = 8$ (sensors number). The total delay measured is $D_{TOT} = 302 T_{CK}$, where T_{CK} is the clock beat and the delay introduced by the single operation is null because it depends on the specific device used. Since the OFDM symbol duration assumed in our system is $8\,\mu s$, it follows that the maximum clock beat for the device is

$$T_{CK} = \frac{T_{OFDM}}{D_{TOT}} = \frac{8\,\mu s}{302} = 26.48\,\mathrm{ns} \tag{75}$$

and the minimum clock frequency necessary to compute all the steps of the algorithm within a symbol interval is

$$f_{CK;min} = 1/T_{CK} = 37.765\,\mathrm{MHz}. \tag{76}$$

Nonetheless, since this result has neglected the operation delay, it must be intended as a *lower bound* on the necessary clock frequency.

The parallelization of the calculations lead us to have a required frequency lower than the one seen in Table 8, where the operations are supposed to be performed in series.

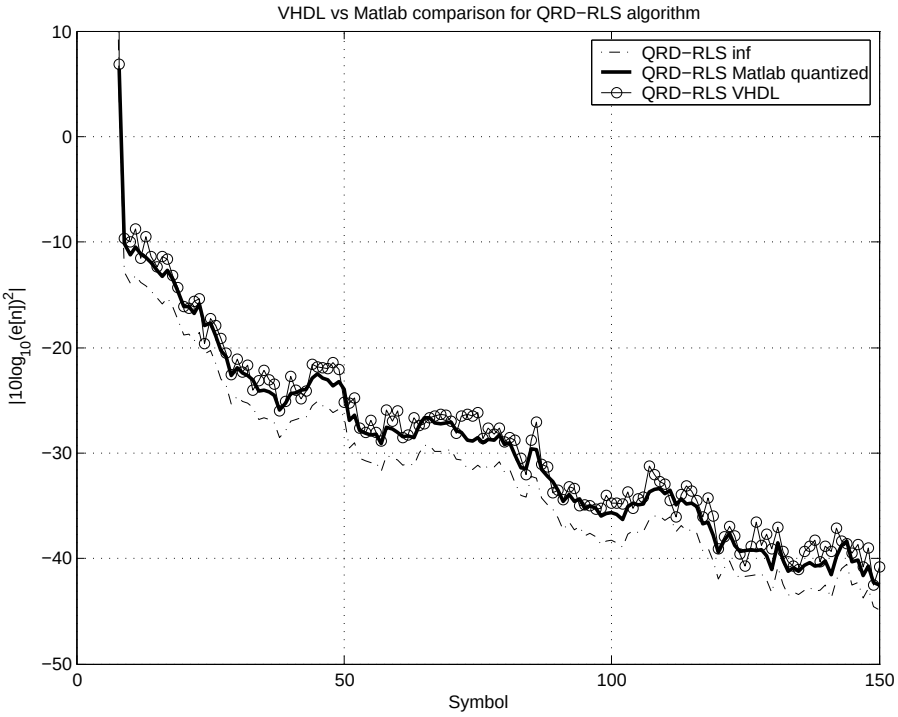


Figure 11: Comparison of the normalized residual errors as a function of the time achieved by: VHDL implementation (crossed line), Matlab® simulated quantization (circled line) and infinite precision implementation (continuous line) of the Rank-4 QRD-RLS algorithm. Quantization is performed over $N_{bit} = 20$ bits $(6 + 1 + 13)$.

Performances of Multirank QRD-RLS in High Noise Conditions				
Rank	No. of operations	Clock frequency [MHz]	Residual error floor 1500-th symbol [dB]	No. of symb. for rank 1 steady-state
4	2660	332.5	-38.95	50

Table 9: Costs/benefits of QRD-RLS Multirank Algorithm, rank 4

5 Conclusions

In this report, an efficient implementation of the Multi-rank QRD-RLS algorithm has been analyzed to overcome Doppler shift problems of a HAP-to-train transmission channel.

Choosing a Multirank QRD-RLS algorithm where the rank is 4 the performances are summarized in Table 9.

The suitability of the algorithm to be implemented in programmable logic on an electronic device with quantization errors has been demonstrated, since it overcomes the instability problems of classic Multi-rank RLS formulations and has low computational effort.

The number of bits needed for representing the data is $N_{bit} = 20$ bits, independently of the rank, subdivided in:

- 1 bit for the sign,
- $\beta_I = 6$ bits,
- $\beta_F = 15$ bits.

The logic block that computes Givens rotations necessitates of a longer wordlength, composed of $N'_{bit} =$

β_I	β_F	**Multi-rank QRD-RLS**	
No. of bits	No. of bits	Mean difference	Max. difference
$6+1$	13	-61.936	-55.377

Table 10: Differences between residual errors in infinite and quantized implementation of the algorithms, as a function of β_F and β_I.

26 bits, $\beta'_I = 12$ and $\beta'_F = \beta_F$ bits. However, the data at the output of this block are represented again over $N_{bit} = 20$ bits, without loss of precision.

The highest quantization error is due to the fractional part and can be seen in Table 10.

The implementation analysis discussed in this report takes into account an OFDM signal scenario, where the rank of the beamforming algorithm is lower than or equal to the number of pilot subcarriers in the transmitted signal. Nonetheless, it can be easily viewed as a direct extension of the algorithm already discussed in the Capanina Deliverable D17 [1] for an IEEE 802.16 Single-Carrier system, therefore the algorithm performance is, in some way, scalable with the number of (sub)carriers, and the results of this reports are directly suitable to account for the Single-Carrier case.

The results of the VHDL synthesis demonstrate that the algorithm can efficiently work on a specific programmable device (e.g., an FPGA), using the VHDL code developed. Considering a null operation delay, the *lower bound* on the necessary clock frequency was found as $f_{CK;min} = 1/T_{CK} = 37.765$ MHz.

References

[1] CAPANINA Deliverable D17, “Beamforming algorithms and implementation aspects for ground terminals and aerial platforms specified,” *Tech. rep. CAP-D17-WP33-UOY-PUB-01*, February 2006.

[2] S. Hara, S. Hane, Y. Hara, “Simple null-steering OFDM adaptive array antenna for Doppler-shifted signal suppression,” *IEEE Transactions on Vehicular Technology*, vol. 54, pp. 91–99, January 2005.

[3] CAPANINA Deliverable D24, “Report on steerable antenna architectures and critical rf circuits performance,” *Tech. rep. CAP-D24-WP32-UOY-PUB-01*, August 2006.

[4] E. Falletti, F. Sellone, C. Spillard, and D. Grace, “A transmit and receive multi-antenna channel model and simulator for communications from high altitude platforms,” *Int. J. on Wireless Information Networks – Spec. Issue on HAP Technology and Trials*, vol. 13, no.1, January 2006.

[5] J. G. Proakis, C. M. Rader, F. Ling, C. L. Nikias, M. Moonen, and I. K. Proudler, *Algorithms for Statistical Signal Processing*. Upper Saddle River, New Jersey, USA: Prentice-Hall, 2002.

[6] K. R. Liu, S. F. Hsieh, K. Yao, “Systolic block householder transformation for rls algorithm with two-level pipelined implementation,” *IEEE Transactions on Signal Processing*, vol. 40, pp. 946–958, April 1992.

[7] S. Roweis, “Matrix Identities,” *http://www.cs.toronto.edu/˜roweis/notes.html*, pp. 1–4, June 1999.

[8] G. H. Golub, C. F. Van Loan, *Matrix computations*. The Johns Hopkins University Press, third ed., 1996.

[9] K. J. Raghunath, K. K. Parhi, “Fixed and floating point error analysis of QRD-RLS and STAR-RLS adaptive filters,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. III,81–III,84, 19-22 April 1994.

[10] CAPANINA Deliverable D14, “Mobile link propagation aspects, channel model and impairment mitigation techniques,” *Tech. rep. CAP-D14-WP22-UOY-PUB-07*, February 2005.

[11] H. Dedieu, M. Hasler, “Error propagation in recursive QRD LS filter,” *International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1841–1844, 14-17 April 1991.

[12] P. S. R. Diniz, M. G. Siqueira, “Fixed-point error analysis of the QR-recursive least square algorithm,” *IEEE Transactions on Circuits and Systems* II*: Analog and Digital Signal Processing*, vol. 42, pp. 334–348, May 1995.