Noname manuscript No.
(will be inserted by the editor)

Solving Weighted Least Squares (WLS) Problems on ARM-based Architectures

Jose A. Belloch · Balázs Bank · Francisco D. Igual · Enrique S. Quintana-Ortí · Antonio M. Vidal

Received: date / Accepted: date

Abstract The Weighted Least Squares algorithm (WLS) is applied to numerous optimization problems, but requires the use of high computational resources, especially when complex arithmetic is involved. This work aims to accelerate the resolution of a WLS problem by reducing the computational cost (relaying on BLAS/LAPACK routines) and the computational precision from double to single. As a test case, we design an IIR filter for a Graphic Equalizer, where the numerical errors due to single precision are easily visualized. In addition, given the importance of low power architectures for this kind of implementations, we evaluate the performance, scalability, and energy efficiency of each method on two different processors implementing the ARMv7 architecture, widely used in current mobile devices with power constraints. Results show that the method that exhibits a high theoretical computational cost overcomes in efficiency other methods with lower theoretical cost in architectures of this type.

Keywords WLS · Audio Processing · Low Power Processors · ARM$^{®}$ Cortex

Jose A. Belloch, Enrique S. Quintana-Ortí
Depto. de Ingeniería y Ciencia de Computadores
Universitat Jaume I de Castelló, Castelló de la Plana, Spain
E-mail: {jbelloch,quintana}@uji.es

Balázs Bank
Dept. of Measurements and Information Systems
Budapest University of Technology and Economics, Budapest, Hungary
E-mail: bank@mit.bme.hu

Francisco D. Igual
Depto. de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid, Madrid, Spain
E-mail: figual@pdi.ucm.es

Antonio M. Vidal
Depto. de Sistemas Informáticos y Computación
Universitat Politècnica de València, Valencia, Spain
E-mail: avidal@upv.es

1 Introduction

This paper aims at assessing the efficiency and power constraints of Weighted Least Squares (WLS) algorithms on ARMv7 architecture.

1.1 Weighted Least Squares Problems

Many optimization, parameter estimation, and approximation problems lead to the task of finding an optimal set of parameters $\mathbf{p}_{\text{opt}}$ (of size of $N \times 1$) that minimize the error between the model $\hat{\mathbf{y}} = f(\mathbf{p})$ and the reality (or observations) contained in $\mathbf{y}$, with $\hat{\mathbf{y}}$ and $\mathbf{y}$ both of dimensions of $M \times 1$. Mathematically, these problems can be expressed as

$$\mathbf{p}_{\text{opt}} = \arg\min_{\mathbf{p}} ||f(\mathbf{p}) - \mathbf{y}||. \tag{1}$$

The uniqueness of the solution and the complexity of the parameter estimation depends both on the type of error norm $||.||$ and the type of function $f(.)$. A particular case occurs when $\hat{\mathbf{y}} = f(\mathbf{p}) = \mathbf{M}\mathbf{p}$ is linear in $\mathbf{p}$, with $\mathbf{M}$ being the modeling matrix of size $M \times N$, and the L_2-norm is used. Then (1) becomes a Linear Least Squares (LLS) problem.

In this case, the error to be minimized, denoted as e_{LS}, is the sum of squares of the deviations:

$$e_{\text{LS}} = \sum_{k=1}^{K} |\hat{y}_k - y_k|^2 = ||\mathbf{M}\mathbf{p} - \mathbf{y}||_2^2 = (\mathbf{M}\mathbf{p} - \mathbf{y})^T(\mathbf{M}\mathbf{p} - \mathbf{y}).$$

We note that, for complex-valued problems, the Hermitian transpose $(.)^H$ (transpose and complex conjugate) should be used instead of $(.)^T$ in all equations throughout the paper.

Since e_{LS} is a second-order function of $\mathbf{p}$, it has a unique minimum, which can be found by solving the normal equations of the LLS [1]:

$$(\mathbf{M}^T\mathbf{M})\mathbf{p}_{\text{opt}} = \mathbf{M}^T\mathbf{y}. \tag{2}$$

This requires the solution of a symmetric positive definite linear system, and makes parameter estimation significantly simpler compared to general nonlinear optimization problems. An alternative method consists in computing the pseudo-inverse of $\mathbf{M}$, given by $\mathbf{M}^+ = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$, so that

$$\mathbf{p}_{\text{opt}} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y}. \tag{3}$$

Although both methods are mathematically equivalent, (2) usually yields numerically better-posed algorithms.

Not all optimization problems can be expressed as LLS problems, either because an output is not a linear function of the parameters or the L_2-norm is not an appropriate metric. However, many of these problems can still be approximated as a LLS problem with additional weights (Weighted Least Squares, or WLS). In this case, all deviations $(\hat{y}_k - y_k)$ are multiplied by a constant w_k before the L_2-norm is computed. Mathematically, this can be formulated as a multiplication with a diagonal matrix $\mathbf{W} = \mathbf{w}$, where $\mathbf{w}$ is the vector containing the elements w_k. In this case, the error is approximated as

$$e = ||f(\mathbf{p}) - \mathbf{y}||^2 \approx e_{\mathrm{WLS}} = ||\mathbf{W}(\mathbf{M}\mathbf{p} - \mathbf{y})||_2^2 = (\mathbf{M}\mathbf{p} - \mathbf{y})^T \mathbf{W}^T \mathbf{W}(\mathbf{M}\mathbf{p} - \mathbf{y}). \tag{4}$$

If the weight matrix $\mathbf{W}$ is fixed, (4) can be solved in one step, similarly to (3), by simply multiplying both $\mathbf{M}$ and $\mathbf{p}$ with $\mathbf{W}$:

$$\mathbf{p}_{\mathrm{opt}} = (\mathbf{M}^T \mathbf{W}^T \mathbf{W} \mathbf{M})^{-1} \mathbf{M}^T \mathbf{W}^T \mathbf{W} \mathbf{y}. \tag{5}$$

A special class of WLS algorithms is Iteratively Reweighted Least Squares [2], where (5) is solved iteratively a number of times in such a way that the modeling matrix $\mathbf{M}$ and the target vector $\mathbf{y}$ remain the same, but the weight matrix $\mathbf{W}$ changes. This will have an importance in Sec. 2.2, where Method 2 takes an advantage of recomputing (5) with $\mathbf{M}$ unchanged. The IRLS technique has been applied to numerous optimization problems. This includes FIR and IIR filter design in L_p sense [2,3], the frequency-domain Steiglitz-McBride algorithm [4], magnitude-priority filter design [5], and sparse recovery [6], to name a few.

A numerically similar problem arises when both $\mathbf{y}$ and $\mathbf{W}$ change, but the modeling matrix $\mathbf{M}$ still remain the same. This is the case when the same basis functions are used to approximate various observation vectors $\mathbf{y}_n$ with different weights $\mathbf{W}_n$. One example is the design of a Parallel Graphic Equalizer, where the target frequency response of the equalizer filter (corresponding to $\mathbf{y}_n$) can change in time instant n, but the response is modeled as the linear combination of the same elementary transfer functions [7]. This will again allow the use of Method 2 presented in Sec. 2.2. Since we have selected the Parallel Graphic Equalizer as a use case, a short overview is presented in the following.

1.2 The Parallel Graphic Equalizer

Equalizers correct or enhance the spectrum of a signal in order to meet a desired requirement. Equalizers are widely used in music production and in sound reproduction to control the timbral balance of music [8,9], as well as to reduce the effects of room acoustics on the sound quality [10]. In graphic equalizers, the user controls the gain of each frequency band using a set of sliders, which approximately modulate the desired magnitude response [7,11–14]. Common graphic equalizers control the gain at 31 frequencies spaced one third of an octave.

The basic idea of the Parallel Graphic Equalizer [7] is that based on the slider positions set by the user (top plot in Figure 1), a smooth frequency response (the target response $\mathbf{y}$) is computed. The actual equalization filtering

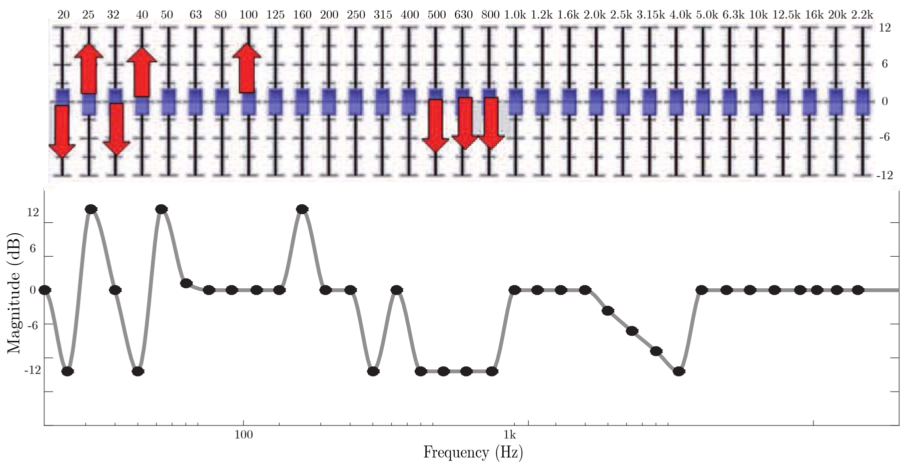

Fig. 1 At the top: an example of sliders positions. In the bottom: Magnitude response that must be produced at the graphic equalizer with these sliders positions.

is done by a parallel set of second-order filters, whose coefficients are computed by a WLS design, where the weights $\mathbf{W}$ are a function of the target response $\mathbf{y}$. That is, each time the sliders change, the target response $\mathbf{y}$ and weights $\mathbf{W}$ are updated based on the slider positions, and a new $\mathbf{p}_{\text{opt}}$ must be computed following (5). Note that the modeling matrix $\mathbf{M}$ remains fixed. The performance of a graphic equalizer can be evaluated by the maximum difference that the resulting frequency-presents with respect to the target response at the command points (black points at the bottom of Figure 1). For a high-quality equalizer, the differences must be lower than 1 dB at the command points [7, 15].

1.3 ARM-based architectures

Graphic equalizers are of special interest for mobile or hand-held audio devices. In general, this kind of appliances is severely constrained by energy consumption, which commonly results in limited performance rates. In this scenario, low power processors become mandatory. During the last years, ARM Cortex processors have become an appealing solution that combines a relatively high floating-point performance together with a restrained energy footprint, attracting the interest from the scientific community to leverage them in order to accelerate critical tasks.

ARM processors have been previously applied in different signal processing applications targeting multimedia applications. In [16], the authors propose different implementations for FIR and IIR filtering in the field of audio processing in the time-domain. Performance of a spatial audio application is reported in [17]. Various image processing algorithms have been accelerated using ARM architectures in [18].

The main goal of this paper is to evaluate how the computational time required to solve a WLS problem can be reduced. Two strategies for accelerating the resolution of a WLS problem are analyzed. On the one hand, we study the computational cost of three different methods when the WLS problem is a part of a real-time application. To this end, we leverage versions of software libraries for numerical linear algebra, such as LAPACK (*Linear Algebra Package*) [19] and BLAS (*Basic Linear Algebra Subprograms*) [20]. On the other hand, we accelerate the resolution of a WLS problem by reducing the computational precision from double to single. We evaluate the behavior of the three methods for both single and double arithmetic in order to assess their numerical stability for the particular case of designing a graphic equalizer. Finally, given the importance of low power architectures for this kind of implementations, we evaluate the performance, scalability, and energy efficiency of each method on two different processors implementing the ARMv7 architecture, widely used in current mobile devices with power constraints.

The rest of the paper is structured as follows. Section 2 proposes three different methods to tackle an IRLS problem, and casts the underlying operations in terms of the LAPACK/BLAS routines that allow a tuned implementation. Section 3 reports the performance, scalability, and energy efficiency of the proposed methods. Finally, Section 4 closes the paper with a few concluding remarks.

2 Methods for solving an IRLS problem

In order to tackle (5), we evaluate three different methods:

- Method 1: solving the normal equations.
- Method 2: computing the QR decomposition to the matrix before weighting.
- Method 3: computing the QR decomposition to an extended matrix constructed by matrix $\mathbf{M}$ and vector $\mathbf{y}$, both of them weighted by matrix $\mathbf{W}$.

Note that Method 2 takes advantage of the fact that the modeling matrix $\mathbf{M}$ stays fixed, thus some parts of the solution can be pre-computed. This is the case for IRLS problems and also for the Parallel Graphic Equalizer used as a test case. Methods 1 and 3 are more general in the sense that they do not make this assumption, so they can be used for a wider set of WLS problems. Hereafter, the notation $\mathbf{A}\backslash\mathbf{b}$ means solving the linear system $\mathbf{Ax}=\mathbf{b}$.

2.1 Method 1

The first method consists in solving the normal equations. To this end, we need to carry out the operations described in Table 1 in sequential order. The fourth column of Table 1 shows the BLAS/LAPACK routines that are used

Table 1 Operations carried out by Method 1. Costs are given in floating-point arithmetic operations (flops)

Operations	Description	Computational Cost (flops)	Routine
$\mathbf{M} \leftarrow \mathbf{WM}$	Weighting of matrix $\mathbf{M}$. $\mathbf{W}$ is a diagonal matrix	MN	–
$\mathbf{y} \leftarrow \mathbf{Wy}$	Element-wise Multiplication	M	–
$\mathbf{A} \leftarrow \mathbf{M}^T\mathbf{M}$	Matrix-Matrix Multiplication	$MN(M + (M-1))$	`xgemm`
$\mathbf{b} \leftarrow \mathbf{M}^T\mathbf{y}$	Vector-Matrix Multiplication	$1N(M + (M-1))$	`xgemv`
$\mathbf{p}_{\text{opt}} \leftarrow \mathbf{A}\backslash\mathbf{b}$	Solving the linear system	$(1/3)N^3 + 2N^2$	`xgesv`

for the implementation[1]. Note that the computation of the product $\mathbf{M}^T\mathbf{M}$ and the solution of the linear system is often a source of numerical rounding errors in practice [21].

2.2 Method 2

An alternative method consists in using the $\mathbf{QR}$ decomposition of the matrix $\mathbf{M} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper triangular matrix. Thus, the pseudo-inverse is computed as

$$\begin{aligned}(\mathbf{WM})^+ &= (\mathbf{M}^T\mathbf{W}^T\mathbf{WM})^{-1}(\mathbf{W}^T\mathbf{M}^T)\\ &= (\mathbf{R}^T\mathbf{Q}^T\mathbf{W}^T\mathbf{WQR})^{-1}(\mathbf{R}^T\mathbf{Q}^T\mathbf{W}^T)\\ &= (\mathbf{R}_1^T\mathbf{E}^T\mathbf{E}\mathbf{R}_1)^{-1}(\mathbf{R}_1^T\mathbf{E}^T)\\ &= \mathbf{R}_1^{-1}(\mathbf{E}^T\mathbf{E})^{-1}\mathbf{R}_1^{-T}\mathbf{R}_1^T\mathbf{E}^T\\ &= \mathbf{R}_1^{-1}(\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T,\end{aligned}$$

where $\mathbf{E} = \mathbf{WQ}_1$. Note that $\mathbf{R} = [\mathbf{R}_1^T \quad \mathbf{0}]^T$ and $\mathbf{Q} = [\mathbf{Q}_1 \quad \mathbf{Q}_2]$. Thus, we can discard those elements of $\mathbf{Q}$ that are multiplied by zeros. In order to obtain $\mathbf{E}^T\mathbf{E}$, we need to obtain a second $\mathbf{QR}$ decomposition:

$$\mathbf{E}^T\mathbf{E} = \mathbf{R}_E^T\mathbf{Q}_E^T\mathbf{Q}_E\mathbf{R}_E = \mathbf{R}_{E1}^T\mathbf{R}_{E1},$$

where $\mathbf{R}_E = [\mathbf{R}_{E1}^T \quad \mathbf{0}]^T$. Now we can compute $\mathbf{p}_{\text{opt}}$ as

$$\begin{aligned}\mathbf{p}_{\text{opt}} &= (\mathbf{WM})^+(\mathbf{Wy})\\ \mathbf{p}_{\text{opt}} &= \mathbf{R}_1^{-1}(\mathbf{R}_{E1}^T\mathbf{R}_{E1})^{-1}\mathbf{E}^T(\mathbf{Wy})\\ \mathbf{R}_{E1}^T\mathbf{R}_{E1}\mathbf{R}_1\mathbf{p}_{\text{opt}} &= \mathbf{E}^T(\mathbf{Wy}).\end{aligned}$$

[1] The character(`x`) in the routine names should be replaced by `s`, `d`, `c`, `z` to indicate operations with single or double precision arithmetic on real or complex values.

Table 2 Operations carried out by Method 2. Costs are given in floating-point arithmetic operations (flops)

Operations	Description	Computational Cost (flops)	Routine
$\mathbf{E} \leftarrow \mathbf{W}\mathbf{Q}_1$	Weighting of matrix $\mathbf{Q}_1$. $\mathbf{W}$ is a diagonal matrix	MN	–
$\mathbf{y} \leftarrow \mathbf{W}\mathbf{y}$	Element-wise multiplication $\mathbf{W}$ is a diagonal matrix	M	–
$\mathbf{c} \leftarrow \mathbf{E}^T\mathbf{y}$	Vector-Matrix Multiplication	$N(M + (M-1))$	`xgemv`
$\mathbf{R}_{E1} \leftarrow \mathbf{W}\mathbf{Q}_1$	Computation of the $\mathbf{R}$ matrix from the $\mathbf{QR}$ decomposition of E	$(2/3)(N^2)(3M-N)$	`xgeqrf`
$\mathbf{u} \leftarrow \mathbf{R}_{E1}^T \backslash \mathbf{c}$ $\mathbf{v} \leftarrow \mathbf{R}_{E1} \backslash \mathbf{u}$ $\mathbf{p}_{\text{opt}} \leftarrow \mathbf{R}_1 \backslash \mathbf{v}$	Solving three triangular linear systems	$3N^2$	`xtrsm`

Denoting $\mathbf{c} = \mathbf{E}^T\mathbf{W}\mathbf{y}$, $\mathbf{v} = \mathbf{R}_1\mathbf{p}_{\text{opt}}$, and $\mathbf{u} = \mathbf{R}_{E1}\mathbf{v}$, we have to solve three triangular linear systems to obtain $\mathbf{p}_{\text{opt}}$:

$$\begin{aligned} \mathbf{R}_{E1}^T\mathbf{u} &= \mathbf{c}, \\ \mathbf{R}_{E1}\mathbf{v} &= \mathbf{u}, \text{ and} \\ \mathbf{R}_1\mathbf{p}_{\text{opt}} &= \mathbf{v}. \end{aligned} \tag{6}$$

With this approach, the pseudo-inverse does not need to be fully computed every time the sliders change; besides, the first **QR** decomposition can be precomputed. This new method only requires the computation of vector **c** and the solution of the three triangular in (6), which can be obtained via the corresponding invocations to the BLAS routine `xtrsm`. The computational algorithm that must be executed when the sliders of the graphic equalizer change is summarized, step by step, in Table 2.

2.3 Method 3

A variation of Method 2 consists in carrying out the **QR** decomposition of the whole system and solving the linear equation as suggested in [21]. To this end, a matrix **A** is formed by concatenating matrix **M** with vector **y** as

$$\mathbf{A} = [\mathbf{W}\mathbf{M} \quad \mathbf{W}\mathbf{y}] \tag{7}$$

Thus, matrix **A** is of size $M \times (N+1)$. Afterwards, the **QR** decomposition of matrix **A** is computed, obtaining two matrices $\mathbf{Q}_A$ and $\mathbf{R}_A$. Extracting the main triangular part of $\mathbf{R}_A$ (in Matlab notation),

$$\mathbf{R}'_A = \texttt{triu}(\mathbf{R}_A(1:N, 1:N)),$$

and the first N rows of column $(N+1)$ of $\mathbf{R}_A$,

$$\mathbf{r}'_A = \mathbf{R}_A(1:N, N+1),$$

the solution can be obtained by solving the triangular linear system:

$$\mathbf{p}_{\text{opt}} = \mathbf{R}'_A \backslash \mathbf{r}'_A \tag{8}$$

The solution of (8) together with the two **QR** decompositions, performed by the LAPACK routine `xgels`, present a computational cost of $2(N^2)(M - N/3) + N^2$ flops. To this cost, we have to add $MN+M$ from the multiplications of **WM** and **Wy** in (7).

3 Performance of the IRLS algorithms

We analyze next the behavior of the IRLS algorithms in order to assess the numerical precision of each method (for the particular case of designing a graphic equalizer), as well as their computational performance and energy efficiency on an ARMv7 architecture.

3.1 Numerical Precision

In order to evaluate the numerical stability of the three methods, we compute the frequency-response of the graphic equalizer considering single and double precision arithmetic.

When using single precision, the curve obtained via Method 1, depicted in Figure 2 (left), barely fits the target response at the graphic equalizer. From the acoustic point of view, the maximum difference between the computed frequency-response at the target command points (slider positions) approaches 6.3058 dB, which significantly exceeds the desired 1 dB threshold error for designing a graphic equalizer [7]. This phenomenon occurs because the condition of the matrix **M** is very high, and therefore the matrix **M** is close to singular or badly scaled.

Figure 2 (right) illustrates the curves obtained from Methods 2 and 3, which practically coincide. We can appreciate that the frequency-response matches the target response quite accurately. From the acoustic point of view, the maximum difference from the computed frequency-response, at target command points (slider positions), is 0.4193 dB for Method 2 and 0.4195 dB for Method 3. These values are within the tolerance range of 1 dB.

In summary, we can conclude that Methods 2 and 3 meet the required accuracy for a graphic equalizer even if executed in single precision; Method 1, however, renders large numerical errors that exceed the target threshold.

Regarding double precision, all three methods obtain a frequency-response of a graphic equalizer that fits the target response accurately. All methods achieve a maximum difference from the computed frequency-response at target command points of less than 1 dB, with a maximum difference of 0.6564 dB.

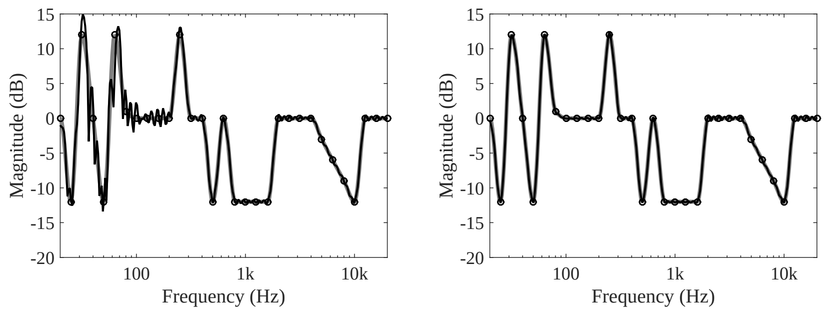

Fig. 2 Left: Target response and frequency-response obtained using Method 1. Right: Target response and frequency-response obtained using the QR-based methods (Methods 2 and 3).

3.2 Computational Performance

In this section, we evaluate the computational performance on a low power system. To this end, we have implemented the three methods on an Exynos 5422 ARM-based System-on-Chip equipped with big.LITTLE technology. We have performed these measurements considering a small matrix $\mathbf{M}$ with dimensions 244×125, that is a typical size for an audio equalizer; and a larger one, with dimensions 8000×4000, that is common in image-related signal processing applications.

We have obtained the processing time for each method, varying the number of cores used in the computation, for two different microarchitectures: ARM Cortex-A7 and Cortex-A15. During those tests, both types of cores operated at a frequency of 1.4 GHz. We have used a reference LAPACK implementation linked with an architecture-tuned BLAS implementation (BLIS version 0.2.0 [1]) with multi-thread support.

Figure 3 shows that the double-precision codes require twice the execution time of the single precision solution. In the four subfigures it becomes clear that the use of the Cortex-A15 cores reduce the processing time by a factor of 4 compared to the Cortex-A7. Regarding the scalability and focusing on the size (8000×4000), all methods show a significant speed-up when the number of cores increases from one to two cores, but the acceleration decreases with three and four cores. The speed-up using the Cortex-A15 cores achieves a highest value of 2, and a maximum value of 3 for the Cortex-A7.

Regarding the small problem (244×125), the multicore processing barely influences the processing time. For the Cortex-A15 and Cortex-A7 cores, the speedup is practically negligible. The processing time of Method 2 is insignificantly larger than that of Method 3. This could be due to the LAPACK routine `degsl/segsl`. Note that we have discarded the performance of Method 1 for the single-precision execution, since it offers invalid results for the specific application of the graphic equalizer. Despite Method 1 presents a higher theo-

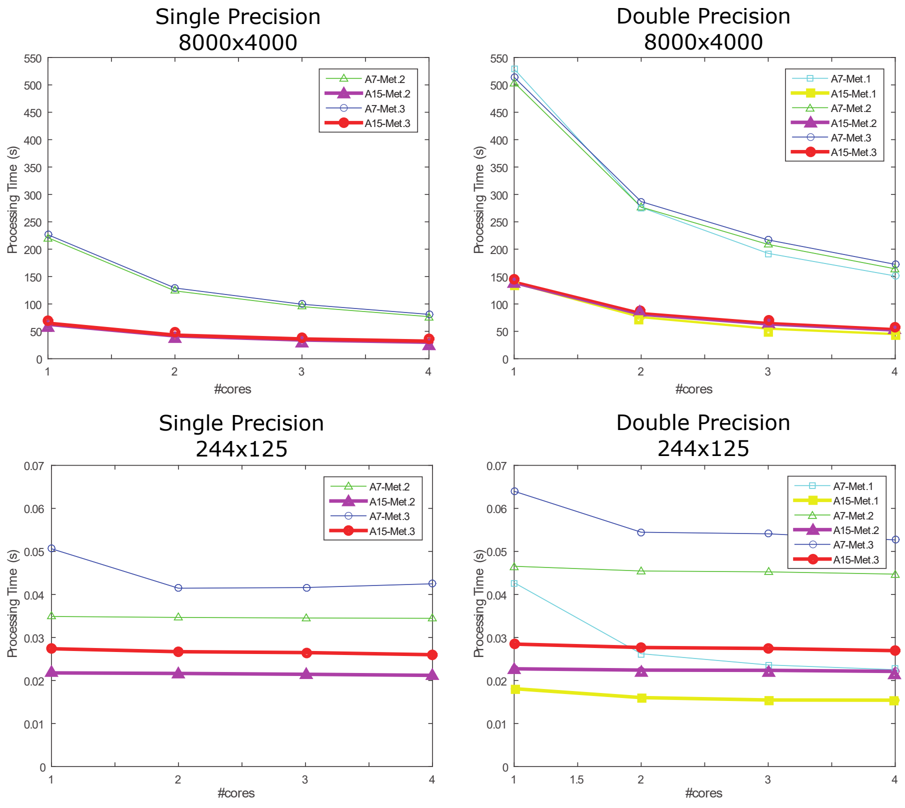

Fig. 3 Processing Time for each proposed method for sizes of matrix $\mathbf{M}$ 8000×4000 and 244×125 considering operations with single and double numerical precisions. $\mathbf{M}$.

retical computational cost, it offers the best computational times compared to the other methods, improving even the results obtained with the Cortex-A15.

3.3 Energy efficiency

An interesting aspect to analyze the energy efficiency of the methods. To this end, we have measured the power dissipation with the `pmlib` framework [22] for each one of the configurations evaluated in the previous subsection. Specifically, we have collected instantaneous power readings in order to obtain the average power dissipated during each execution. Afterwards, we obtained energy efficiency metric that considers: computational cost relaying on BLAS/LAPACK routines, average power rate and processing time. Specifically, we show at Figure 4 the GFlops/Watt for each one of the configurations.

As we can appreciate, Method 1 offers maximum efficiency compared to Method 2 and 3 for double precision executions. For this case, the decisive factor is the low processing time that Method 1 requires for the execution.

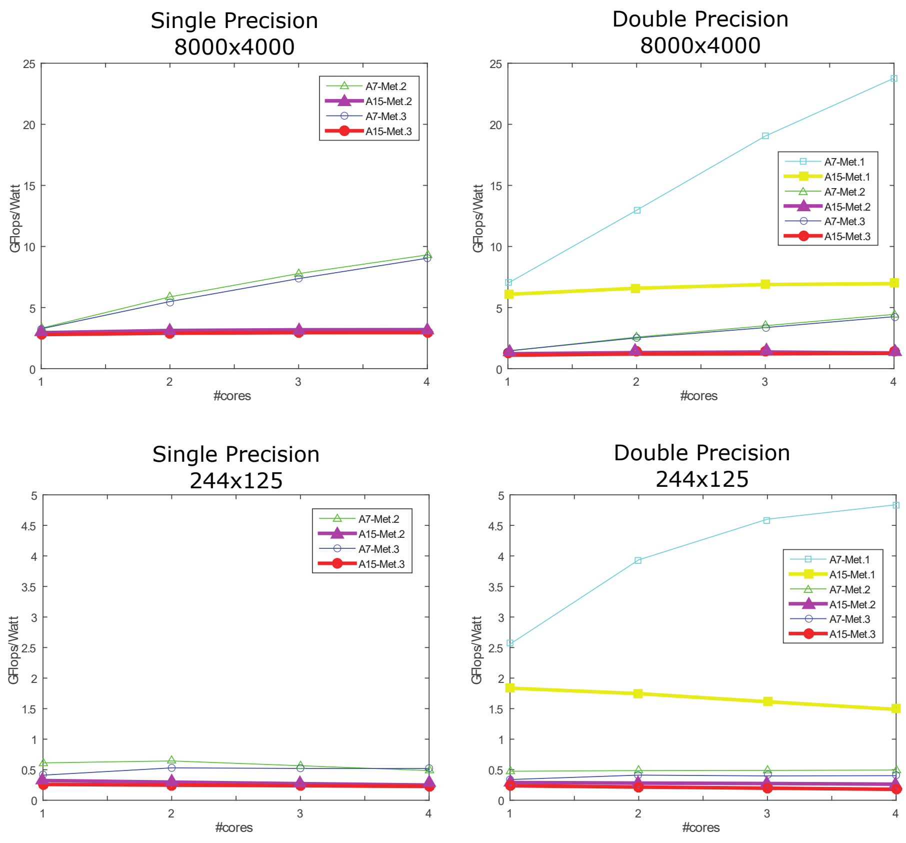

Fig. 4 GFlops/Watt for each proposed method for sizes of matrix $\mathbf{M}$ 8000×4000 and 244×125 considering operations with single and double numerical precisions.

This factor together with the low consumption of the Cortex-A7 makes that the combination of Method 1 with double precision and executed on Cortex-A7 being the most efficient implementation achieving 24 GFloops/Watt for a large size and 5 GFloops/Watt for a small size. For simple precision, Method 2 and Method 3 achieves similar efficiency being Method 2 slightly higher than Method 3. In any case, both Methods obtain a third of efficiency that is achieved by Method 1.

4 Conclusions

In this paper, we have analyzed QR-based methods for improving the performance of applications that rely on the IRLS algorithm. Two of these methods are especially appealing, from the perspective of numerical accuracy, and enable the use of single precision. Concretely, for the specific application of the graphic equalizer, the QR-based methods result in more accurate frequency-responses than Method 1.

We have executed these methods for the WLS problem both in single and double precision on an ARM-based architecture composed of four Cortex-A7 cores and four Cortex-A15 cores. We have assessed both their computational performance and energy efficiency for different configurations. We have to highlight the efficiency of Method 1 when it is implemented on Cortex-A7 in double precision. This efficiency overcomes in three times the rest of the configurations in double precision.

Acknowledgements This work started in spring 2016 when Jose A. Belloch was a visiting postdoctoral researcher at Budapest University of Technology and Economics thanks to the European Network COST Action IC1305 inside the program Short Term Scientific Mission with the following reference: COST-SPASM-ECOST-STSM-IC1305-020416-072431. Dr. Jose A. Belloch is supported by GVA contract APOSTD/2016/069. The researchers from Universitat Jaume I are supported by the CICYT projects TIN2014-53495-R of MINECO and FEDER. The authors from the Universitat Politècnica de València are supported by MINECO projects TEC2015-67387-C4-1-R, PROMETEOII/2014/003 and CAPAP-H5 network TIN2014-53522-REDT. The researcher from UCM is supported by the EU (FEDER) and the Spanish MINECO, under grants TIN 2015-65277-R and TIN2012-32180.

References

1. T. M. Smith, R. A. van de Geijn, M. Smelyanskiy, J. R. Hammond, and F. G. Van Zee, "Anatomy of high-performance many-threaded matrix multiplication," in *28th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2014)*, 2014.
2. C. S. Burrus, "Iterative reweighted least squares," *OpenStax-CNC document*, May 2012, module m45285, http://cnx.org/content/m45285/1.12.
3. S. W. Khang, "Best L_p approximation," *Mathematics of Computation*, vol. 26, no. 118, pp. 505–508, July 1972.
4. L. B. Jackson, "Frequency-domain Steiglitz-McBride method for least-squares filter design, ARMA modeling, and periodogram smoothing," *IEEE Signal Process. Lett.*, vol. 15, pp. 49–52, 2008.
5. B. Bank, "Magnitude-priority filter design for audio applications," in *Proc. 132*$^{\text{nd}}$ *AES Conv., Preprint No. 8591*, Budapest, Hungary, May 2012.
6. I. Daubechies, R. Devire, M. Fornasier, and C. S. Gntrk, "Iteratively reweighted least squares minimization for sparse recovery," *Computer Music J.*, vol. 23, no. 2, pp. 52–69, Jan. 2010.
7. J. Rämö, V. Välimäki, and B. Bank, "High-precision parallel graphic equalizer," *IEEE/ACM Trans. Audio Speech Language Processing*, vol. 22, no. 12, pp. 1894–1904, Dec. 2014.
8. E. Perez Gonzales and J. Reiss, "Automatic equalization of multi-channel audio using cross-adaptive methods," in *Proc. AES 127th Conv.*, New York, Oct. 2009.
9. J. Rämö and V. Välimäki, "Live sound equalization and attenuation with a headset." in *Proc. AES 51st Int. Conf.*, Helsinki, Finland, Aug. 2013.
10. A. Mäkivirta, P. Antsalo, M. Karjalainen, and V. Välimäki, "Modal equalization of loudspeaker-room responses at low frequencies," *J. Audio Eng. Soc.*, vol. 51, no. 5, pp. 324–343, May 2003.
11. M. Holters and U. Zölzer, "Graphic equalizer design using higher-order recursive filters," in *Proc. Int. Conf. Digital Audio Effects*, Montreal, QC, 2006, pp. 37–40.
12. S. Tassart, "Graphical equalization using interpolated filter banks," *J. Audio Eng. Soc.*, vol. 61, no. 5, pp. 263–279, May 2013.

13. Z. Chen, G. S. Geng, F. L. Yin, and J. Hao, "A pre-distortion based design method for digital audio graphic equalizer," *Digital Signal Process.*, vol. 25, pp. 296–302, Feb. 2014.
14. V. Välimäki and J. Reiss, "All about audio equalization: Solutions and frontiers," *Applied Sciences*, vol. 6, no. 5, pp. 129–145, 2016.
15. J. A. Belloch and V. Välimäki, "Efficient target-response interpolation for a graphic equalizer," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 564–568.
16. J. A. Belloch, F. J. Alventosa, P. Alonso, E. S. Quintana-Ortí, and A. M. Vidal, "Accelerating multi-channel filtering of audio signal on arm processors," *The Journal of Supercomputing*, pp. 1–12, 2016.
17. J. A. Belloch, A. Gonzalez, F. D. Igual, R. Mayo, and E. S. Quintana-Ortí, "Vectorization of binaural sound virtualization on the ARM cortex-A15 architecture," in *Proceedings of 23rd European Signal Processing conference, (EUSIPCO)*, Nize, France, September 2015.
18. G. Mitra, B. Johnston, A. Rendell, E. McCreath, and J. Zhou, "Use of simd vector operations to accelerate application code performance on low-powered arm and intel platforms," in *IEEE 27th International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, May 2013, pp. 1107–1116.
19. S. Tomov, J. Dongarra, and M. Baboulin, "Towards dense linear algebra for hybrid gpu accelerated manycore systems." LAPACK Working Note, Tech. Rep. 210, Oct. 2008. [Online]. Available: http://www.netlib.org/lapack/lawnspdf/lawn210.pdf
20. J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, "A proposal for an extended set of Fortran basic linear algebra subprograms," *ACM Signum Newsletter*, pp. 2–18, 1985.
21. G. H. Golub and C. F. V. Loan, *Matrix Computations*, 4th ed. Baltimore, Maryland: The John Hopkins University Press, 2013.
22. P. Alonso, R. M. Badia, J. Labarta, M. Barreda, M. F. Dolz, R. Mayo, E. S. Quintana-Ortí, and R. Reyes, "Tools for power-energy modelling and analysis of parallel scientific applications," in *41st Int. Conf. on Parallel Processing – ICPP*, 2012, pp. 420–429.