



HAL
open science

An algorithm for extremal eigenvectors computation of Hermitian matrices and its FPGA implementation

Guillaume Lucius, Frédéric Le Roy, Denis Aulagnier, Stéphane Azou

► **To cite this version:**

Guillaume Lucius, Frédéric Le Roy, Denis Aulagnier, Stéphane Azou. An algorithm for extremal eigenvectors computation of Hermitian matrices and its FPGA implementation. IEEE International Midwest Symposium on Circuits and Systems (MWSCAS 2013), Aug 2013, Columbus, OH, United States. 10.1109/MWSCAS.2013.6674920 . hal-01508692

HAL Id: hal-01508692

<https://hal.science/hal-01508692>

Submitted on 14 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An algorithm for extremal eigenvectors computation of Hermitian matrices and its FPGA implementation

G. Lucius

Thales Systèmes Aéroportés
SAS
Brest, France
guillaume.lucius@fr.thalesgroup.com

F. Le Roy

CNRS, UMR 6285 Lab-STICC, ENSTA Bretagne
Brest, France
frederic.le_roy@ensta-bretagne.fr

D. Aulagnier

Thales Systèmes Aéroportés
SAS
Brest, France
denis.aulagnier@fr.thalesgroup.com

S. Azou

CNRS, UMR 6285 Lab-STICC, ENIB
Brest, France
azou@enib.fr

Abstract—We consider the problem of implementing an algorithm for the extraction of leading eigenvectors of a small Hermitian matrix on field-programmable gate array (FPGA). The evolution of FPGAs can now handle increasingly bandwidth problems or larger in size. Jacobi algorithms are usually implemented in FPGA for real matrix size not exceeding 20×20 . The increase in size or complex number problem may lead to use other algorithms such as Lanczos, which are rarely implemented on FPGA. Recently, it has been pointed out that the Lanczos method can efficiently address the extreme eigenvalues computation problem on FPGA, for medium size real matrices. This paper presents an algorithm for the extraction of extremal eigenvalues and corresponding eigenvectors for small Hermitian matrix using a high-level approach for the architecture synthesis.

Keywords—Field Programmable Gate Arrays; High-level Synthesis; Hermitian eigenpairs.

I. INTRODUCTION

Estimation of direction of arrival (DOA) parameters is involved in many fields such as telecommunication, radar, sonar [1] or seismic [2]. DOA estimation relies on subspace methods that typically consist in achieving a spectral decomposition of the covariance matrix of the measurements. The decomposition usually requires the computation of a few eigenvalues and their corresponding eigenvectors of a complex Hermitian matrix. Depending on the application and the instantaneous bandwidth, real-time constraints may require the use of FPGA to take advantage of parallel computing. Implementation of the Lanczos algorithm is very uncommon on FPGAs and appears only recently. In [3], it is shown that FPGAs are suitable for medium size matrices (a few hundreds) and high-bandwidth applications, but the case of Hermitian matrices is not considered. We propose to investigate the Hermitian case and also to compute the eigenvectors; moreover, we focus on the particular scenario where the measurement process leads to small covariance matrices. The proposed algorithm is implemented on a FPGA using a High-Level Synthesis (HLS) approach. The paper is organized as follows: section II gives a short overview of existing methods for solving the extremal eigenvalues problem. Then, section III details a particular algorithm and its variants are discussed.

Finally, before some conclusions, we describe the proposed architecture resulting from use of some HLS tools.

II. MATHEMATICAL BACKGROUND

The extremal eigenvalues/eigenvectors problem can be solved in different ways. The choice of the algorithm depends on the matrix properties, its size, and the number of desired eigenvalues. As part of a wideband receiver, high sampling frequencies lead to a large amount of dataflow that requires maximizing the computation efficiency. In the present paper, we consider the case of a $n \times n$ complex Hermitian matrix denoted by \mathbf{A} , with n relatively small (a few tens), for which the first two largest eigenvalues and associated eigenvectors are computed. Then, the algorithm is chosen to avoid unnecessary calculations and take part of the Hermitian properties of the matrix.

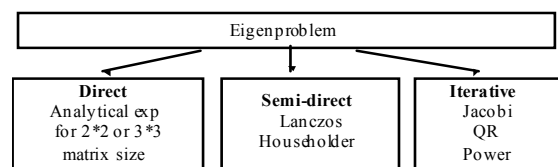


Fig. 1. Classification of Eigen methods

The main numerical methods to solve the eigenproblem can be classified in three categories (fig 1). Direct methods are not often used because they are limited to the case of a very small matrix size ($n < 4$), by using the analytical expressions for the eigenpairs. For larger problems, there are semi-directs and iterative methods. Semi-direct methods use an intermediate matrix and continue the diagonalization with an iterative method, whereas iterative methods such as the Jacobi method are completely iterative. Generally, conventional methods such as Jacobi, compute the eigenvalues and eventually the eigenvectors with a complexity $O(n^3)$. Their most common use is when all eigenvalues are required. However, when only a few eigenvalues are needed, it is sometimes preferable to use appropriate methods such as the Power method [4]. It is also possible to use semi-direct methods that combine a matrix

transformation with Lanczos or Householder methods and then compute a part of eigenpairs with an iterative algorithm. The Power method can be used to quickly obtain the leading eigenvectors of the matrix \mathbf{A} . However the convergence depends on the eigenvalues distribution and a very large number of iterations can be required when the eigenvalues are close, which prevents obtaining a good approximation in a reasonable time in the context of a high dataflow. It is well known that a Hermitian matrix has some particular properties that simplify the eigendecomposition. Thus, a Hermitian matrix is a square self-adjoint matrix, diagonalizable, with real eigenvalues [5]. In our case, it is more interesting to go through an intermediate tridiagonal matrix as shown in Fig 2. The remaining calculations are then easier thanks to the reduced number of data to be manipulated.

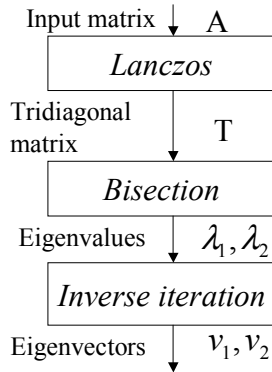


Fig. 2. Three stages proposed algorithm

We propose a three stage semi-direct algorithm, whose particularities are detailed in section 3. It consists in a tridiagonalisation via Lanczos, followed by the bisection method, to obtain the eigenvalues; then the inverse iteration is performed to get corresponding eigenvectors.

III. CHOICE OF ALGORITHMIC VARIANTS

A. Background

DOA estimation methods generally use the noise or signal subspaces [6]. These subspaces being orthogonal, the complete eigendecomposition is not really necessary and represents a superfluous computational cost. Moreover, the pseudo-spectrum projection does not really need the exact eigenvector but only the subspace spanned. Subsequently, we therefore seek the subspace spanned by the first two eigenvectors.

There are methods to perform a partial diagonalization such as Lanczos or Householder transformation methods. Applied to a Hermitian matrix, they produce a real symmetric tridiagonal matrix. This particular structure simplifies the rest of the algorithm. The bisection method is used to compute the eigenvalues by dichotomy. Once this first step is terminated, it remains to find the two corresponding eigenvectors using the inverse iteration. This one is a variation of the Power method applied to the inverse matrix. Starting with a correct

approximation of the eigenvalues, this method will usually find the corresponding eigenvectors in one iteration.

B. The Lanczos method

The Lanczos method is an iterative method based on Krylov space. It allows, from an initial vector q_0 , to construct a subspace $\mathbf{K}_n(\mathbf{A}, q_0) = [q_0, \mathbf{A}q_0, \dots, \mathbf{A}^{n-1}q_0]$. At the k th iteration, a $k \times k$ tridiagonal matrix \mathbf{T}_k is created from the original matrix \mathbf{A} . \mathbf{T}_k is composed of α_k and β_k , diagonal and sub-diagonal elements respectively (see Fig 3). In the end, the matrix \mathbf{T} is similar to the matrix \mathbf{A} and therefore has the same eigenvalues in the generated subspace \mathbf{K}_n . In the case of a finite representation, a loss of orthogonality may be observed after a few cycles. The current Lanczos vector loses its orthogonality compared to the previous vectors and reorthogonalization is necessary to keep an orthonormal basis and avoid spurious eigenvalues.

However, if we are only interested in extreme eigenvalues, the loss of orthogonality does not affect them [7,8] and the reorthogonalization is not necessary. The overall computational cost can be computed easily using Fig. 3. Cost is then reduced to $8n^3 + 28n^2 - 2n$, compared to the Lanczos method with orthonormalization.

```

Input :  $\mathbf{A}$    Outputs :  $\alpha_k, \beta_k$ 
for  $k = 1..n$ 
  if  $k > 1$ 
     $t = w; w = v/\beta_{k-1};$  (11)
     $v = -\beta_{k-1} * t;$  (12)
  endif
   $v = v + \mathbf{A} * w;$  (13)
   $\alpha_k = w^H * v;$  (14)
   $v = v - \alpha_k * w;$  (15)
   $\beta_k = \|v\|;$  (16)
end for
  
```

Fig. 3. Lanczos algorithm, two vectors working version without reorthogonalization

For large matrix, the Lanczos method without reorthogonalization is more interesting in terms of computational cost [4], its implementation will be discussed in the next section. Compared to the Householder method, both of them have the advantage of making the tridiagonal transformation into a known number of iterations, but the presence of a matrix product penalizes Householder with a $16n^4 - 48n^3 + 7n^2 + 10n - 93$ cost. Nevertheless, comparison (Fig 4) shows an intersection of computational costs, which reveals that the Householder method may be preferable in case of very small matrices.

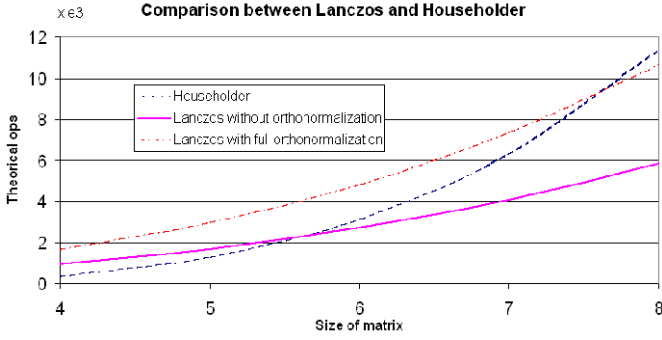


Fig. 4. Comparison of computational costs for Lanczos and Householder.

C. Bisection method

The bisection method allows to get the eigenvalues of the tridiagonal matrix \mathbf{T} . Compared to other methods such as the QR algorithm that achieves the full diagonalization, the bisection method is used here to compute one eigenvalue at a time and avoid unnecessary calculation. It uses the simplified expression of the characteristic polynomial of \mathbf{T} to find the roots. The approach consists in using the Sturm sequence evaluated at a point c , $p_k(c) = \det(\mathbf{T}_k - c\mathbf{I})$. Thus, the number of sign changes in the sequence is equal to the number of eigenvalues of \mathbf{T} smaller than c . Then, the searched eigenvalue is estimated by dichotomy until a given precision is reached. The Gerschgorin theorem [9] is used to obtain a rough approximation of the eigenvalues, and then to identify the first interval bisection.

D. Inverse iteration

The inverse iteration method is very similar to the Power method, except that it applies to the inverse of the \mathbf{A} matrix. As we don't keep Lanczos vectors due to non-orthogonalization, use of eigenvalues is our unique choice and this method can compute the eigenvector corresponding to the minimum eigenvalue. From an initial vector x^1 , we form a vector sequence x^k , $k = 1, 2, \dots$ each one being computed as $x^{k+1} = \mathbf{A}^{-1} x^k$; this sequence is known to converge towards the minimal eigenvectors.

Due to the large computational burden involved by the matrix inverse calculation, the inverse iteration approach is not applied in this way. Multiplying the last equation by \mathbf{A} we obtain $\mathbf{A}x^{k+1} = x^k$. As the matrix \mathbf{A} is constant, we have every reason to make its LU decomposition to bring this system into two triangular systems, which is easier to solve. As

$$\text{a result, the system to solve takes the form } \begin{cases} \mathbf{L}^* y = x^k \\ \mathbf{U}^* x^{k+1} = y \end{cases}.$$

Thus, we solve the first system to find y and replace it in the second system to find x^{k+1} . This technique is also known as forward and backward substitution. If u is an approximated eigenvalue of λ resulting from the bisection method, then $\mathbf{A} - u\mathbf{I}$ has an eigenvalue closed to zero and the inverse

iteration converges to the eigenvector associated with λ . We can therefore calculate any eigenvector, if we have an approximation of its corresponding eigenvalue.

IV. EXPERIMENTAL RESULTS

A. Computational cost analysis

Taking into consideration that two real operations are required for a complex addition, six real operations for a complex multiplication and thirteen for a complex division, the exact computational cost of each method can be deduced and compared to algorithms for real matrices. One of the main advantages of using Lanczos without orthonormalization is that the computational cost does not depend on the data but only on the size of the matrix. In the case of Lanczos, the ratio of computational costs for complex/real cases stabilizes at a value of four, which shows that a single iteration consists mainly of multiplications accumulations. The Lanczos process is mainly composed of a matrix-vector product (13, Fig 3), representing 85% of the computational cost of one iteration for $n = 20$; this ratio increases asymptotically with n .

B. High-Level Synthesis tools

HLS is the automated design process design that converts a high-level design specification into a Register Transfer Level (RTL) implementation. This type of tools emerged in the later 80's, but their adoption has increased these last years with third generation tools, which commonly uses ANSI C/C++/System C as input languages description [10]. During this work, we used the Xilinx Vivado HLS tool that enables to generate an adapted implementation for Xilinx components. In the case of floating-point operations; they will be implemented by the appropriate floating-point Xilinx Core Generator cores [11].

Vivado HLS tool allows architecture design exploration using directives in the form of pragma directly in the GUI. Within functions, C/C++ descriptions are generally implemented as series of loops. Much of design exploration goes through loop directives. The most significant ones are pipelining or unrolling for loops that share or parallelize resources. As we are potentially dealing with a large amount of data, memory directives are crucial too. Partitioning or reshaping data in the memory can avoid bottlenecks. These features allow generating various architecture designs from a single input code and therefore the architecture exploration is faster than rewriting the VHDL code.

C. Architecture exploration of the Lanczos method

As the Lanczos computational cost is essentially composed by the matrix-vector multiplication (13), the implementation result depends mostly on this operation. The strong dependence of data between 11, 12, 14, 15 and 16 in one iteration of Lanczos leads to a parallelization potential only due to the matrix-vector multiplication. Pipelining is used to maximize the efficiency of resources and the matrix-vector multiplication is implemented as n parallel dot vector products.

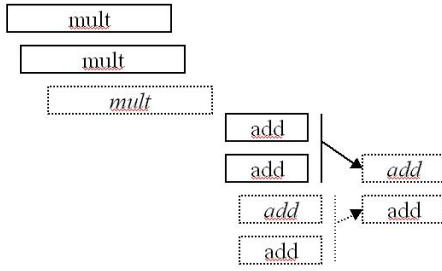


Fig. 5. Implementation of the dot vector product

Flexibility is obtained by instantiating one or more dot vector products. Expressing as much as possible operations as loops provided architecture flexibility. Dot-vector product is implemented using only one pipelined multiplier and two adders (real and imaginary parts) as a reduction tree (fig 5).

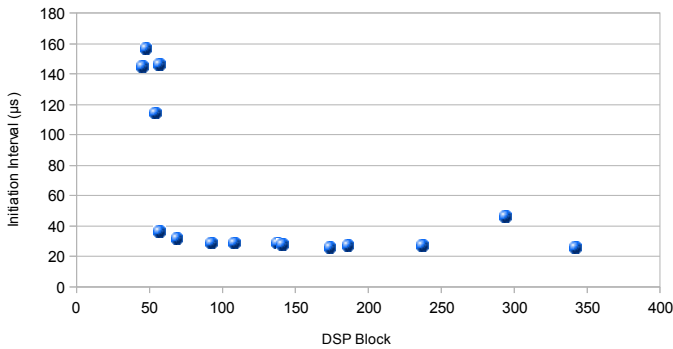


Fig. 6. Architecture exploration of the Lanczos method for $n=16$

The architecture exploration can quickly converge to an efficient solution by analyzing the performance data provided by the tool. Fig. 6 shows an example of architecture exploration; the inflexion point from the asymptotic curve of candidate solutions represents the most effective one. Setting the number of dot-vector products in parallel or the number of adders/multipliers in this function carries out most of the exploration. Artifacts with high initiation interval reflect bottlenecks; as a result, memory access and parallelism must match.

D. Architecture of the bisection method

As for the Lanczos method, bisection is essentially composed of mulacc and potential parallelism is limited due to the iterative nature of the algorithm. Mismatch between throughput and latency during the Sturm sequence leads to underutilization of the pipeline. To completely fill the pipeline, multiple eigenvalues are solved simultaneously. Thereby, up to four eigenvalues can be computed without increasing resources. However, HLS tool finds its limits for this kind of function; optimization and architecture exploration are difficult without explicitly writing the architecture in the C code.

E. Architecture exploration of the inverse iteration

Inverse iteration revolves around the LU decomposition in terms of complexity. Even if the LU decomposition using Cholesky's algorithm is optimized for Hermitian matrices we

firstly use the Crout's algorithm to perform the decomposition because reordering loops is easier in this way. The LU decomposition uses a lot of mulacc, which leads to many bottlenecks. To avoid mulacc architecture, reordering loops in the C code is used to compute one monomial term at a time sweeping the whole matrix before performing accumulation. We present the results for $n=16$ (Tab I), p eigenvectors and q bisection iterations; in our case q has never exceeded 13.

TABLE I. IMPLEMENTATION RESULTS FOR $N=16$

Algorithm	Theoretical ops	DSP48	Throughput
Lanczos	39904	69	31,6 μ s
Bisection	$p*1664$	6	$q*3,2\mu$ s
Inverse iteration	$p*10790$	$p*295$	370 ns

^a. Result of implementation on XC7VX485T device @300MHz for p eigenvalues

V. CONCLUSION & FUTURE WORK

We presented a three-stage semi-direct algorithm for the computation of eigenvectors of a Hermitian matrix implemented as a pipelined architecture using a HLS approach instead of VHDL. It has been shown that this approach allows to quickly and easily explore several designs for getting efficient solutions. However, handover for this kind of tool requires a particular way to write C/C++ code and adaptation time is required. Several additional improvements have been identified: we intend to develop the Cholesky's algorithm for the LU decomposition using the Hermitian symmetry of the matrix. In the case of a lower dynamic input, it would be interesting to work with fixed-point numbers that are more suited to FPGA's and their DSP blocks. And finally, some algorithmic variants have been identified for smaller matrices, keeping in mind the known and fixed execution time constraint.

REFERENCES

- [1] G. Bienvu, "Influence of the spatial coherence of the background noise on high resolution passive methods," *IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 306-309, 1979.
- [2] R.L. Kirilin, W. J. Done, *Covariance analysis for seismic signal processing*. Soc. of Exploration Geophysicists, 1999.
- [3] A. Raffique, N. Kapre, G.A Constantinides, "A High Throughput FPGA-Based Implementation of the Lanczos Method for the Symmetric Extremal Eigenvalue Problem." *Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, Vol. 7199*, pp 239-250, 2012.
- [4] G. H. Golub, C. F. Van Loan, *Matrix computations*. Johns Hopkins University Press, 1996.
- [5] S. Axler, *Linear algebra done right*. Springer, 1997.
- [6] Z. Chen, G. Gokeda, Y. Yu, *Introduction to Direction-of-arrival Estimation*. Artech House Publishers, 2010.
- [7] J.W. Demmel, *Applied numerical linear algebra*. Society for Industrial Mathematics, 1997.
- [8] J.K Cullum, R.A Willoughby, *Lanczos algorithms for large symmetric eigenvalue computations*. Society for Industrial Mathematics, 2002.
- [9] R. S. Varga, *Geršgorin and his circles*. Springer, 2004.
- [10] G. Martin, G. Smith, "High-level synthesis: Past, present, and future," *Design & Test of Computers, IEEE*, vol. 26, no. 4, p. 18-25, 2009.
- [11] Xilinx, *Vivado Design Suite User Guide : High-Level Synthesis*. http://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug902-vivado-high-level-synthesis.pdf, 2012