ADAPTIVE APPROXIMATE ROTATIONS FOR COMPUTING THE EVD

Jürgen Götze
Technical University of Munich
Institute of Network Theory and Circuit Design
Arcisstr. 21, 80290 Munich, Germany

Gerben J. Hekstra
Delft University of Technology
Department of Electrical Engineering
Mekelweg 4, 2628 CD Delft, The Netherlands

ABSTRACT. In this paper, we present an algorithm for computing the eigenvalue decomposition (EVD) of a symmetric matrix. The EVD is computed using a Jacobi-type method, where the angle of the rotations is approximated by an angle α_k corresponding to an orthonormal μ–rotation. These CORDIC-like orthonormal μ–rotations share the property that performing the rotation requires a minimal number of shift-add operations, which can be executed on a floating-point CORDIC architecture. We present various methods of construction for such orthonormal μ–rotations of increasing complexity. Moreover, the computations to determine which angle α_k to use in the approximation of the angle, can itself be expressed purely in orthonormal μ–rotations on the matrix data. The algorithm is highly suitable for VLSI–implementation. The complexity of the rotations decreases during the diagonalization of the matrix by the Jacobi method and a significant reduction of required shift-add operations is achieved. Adapting the number of μ–rotations per rotation of the Jacobi method in order to regain quadratic convergence is also discussed.

KEYWORDS. Eigenvalue decomposition (EVD), Jacobi method, (adaptive) approximation of the rotation angle, orthonormal μ–rotations.

1 INTRODUCTION

Computing the EVD of a $n \times n$ symmetric matrix $\boldsymbol{A}$, i.e.,

$$\boldsymbol{A} = \boldsymbol{Q}^T \boldsymbol{\Lambda} \boldsymbol{Q},$$

where $\boldsymbol{Q}$ is an orthogonal matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix, is a frequently encountered problem in a great number of scientific applications, e.g. signal processing. In order to meet the real time requirements present in many of these applications, it is often necessary to compute the EVD on a parallel architecture.

The method of choice for the fast parallel computation of the EVD is the Jacobi method, since it offers a significantly higher degree of parallelism than the respective QR–method [1, 3]. One sweep of a cyclic Jacobi method (consisting of $n(n-1)/2$ rotation evaluations and their application to the matrix) can be implemented on an upper triangular array of processors with nearest neighbour interconnections in $O(n)$ time [11]. Usually, $O(\log_2 n)$ sweeps are required.

The complexity of the parallel or sequential implementations is mainly determined by the complexity of the rotation evaluation (angle computation). Therefore, different strategies for modifying the rotations have been presented:

- **M1:** approximate rotations [12],
- **M2:** factorized rotations [15, 12],
- **M3:** CORDIC [3, 4],
- **M4:** Combination of the modifications M1–M3:
  - **M1 + M2:** factorized approximate rotations, [7, 10]
  - **M1 + M3:** CORDIC–based approximate rotations (μ–rotations) [11, 17]

Which of these modified schemes yields the most efficient parallel implementation depends on the particular parallel architecture. With respect to an efficient VLSI–architecture a CORDIC–based method is favourable, whereby using CORDIC–based approximate rotations is advantageous compared to the use of the original (exact) CORDIC [11].

In this paper an algorithm for the computation of the EVD by a Jacobi–type method is presented, which can be mapped to an efficient, parallel VLSI architecture. The Jacobi–type method is implemented using different types of orthonormal μ–rotations. These orthonormal μ–rotations realize approximate rotations (i.e. the off–diagonal element is only reduced instead of annihilated as by an exact rotation). They can be executed on the floating point CORDIC architecture presented in [13]. The cost and complexity (number of required shift-add operations) of the different μ–rotations decreases as the rotation angle decreases. Since the required rotation angle decreases during the course of the Jacobi method, the type of μ–rotation used during the course of the algorithm becomes less complex, i.e., the type of μ–rotation is adapted to the stage of the diagonalization. Furthermore, it is also possible to adapt the accuracy of the rotation by increasing the number of μ–rotations used in the approximation of the angle in order to regain the quadratic convergence of the exact Jacobi method [11, 9].

In section 2 we review the cyclic Jacobi algorithm for computing the EVD of a symmetric matrix and describe the idea of using approximate rotations. Section 3 gives the basic

definition of fast, orthonormal μ–rotations and presents several methods for unscaled orthonormal μ–rotations (method I, II, III), or for μ–rotations (method IV) requiring extra scaling iterations to make them orthonormal. It is also shown how to construct a set $\mathcal{A}$ of approximation angles using the above methods, and how to select the optimal approximation angle from this set, i.e. that one which effects in the greatest reduction of the off-diagonal norm. Section 4 demonstrates how the convergence speed of the algorithm can be steered by adaptively varying the number of μ–rotations used in the approximation of the rotation. Section 5 concludes the paper.

2 JACOBI'S METHOD

With respect to a parallel implementation of Jacobi's method for computing the EVD a cyclic Jacobi method is used[.1]

2.1 Cyclic Jacobi method

The cyclic Jacobi method computes the EVD of a $n \times n$ symmetric matrix by applying a sequence of orthonormal rotations to the left and right, of $\boldsymbol{A}$ as shown in the following algorithm

$h := 1$; $\boldsymbol{A}^{(1)} := \boldsymbol{A}$
for $s := 1$ **to** number of sweeps
  for $p := 1$ **to** $n-1$
    for $q := p+1$ **to** n
      $\boldsymbol{A}^{(h+1)} := \boldsymbol{J}_{pq}(\theta_h)\boldsymbol{A}^{(h)}\boldsymbol{J}_{pq}^{T}(\theta_h)$
      $h := h+1$
    end
  end
end

where $\boldsymbol{J}_{pq}(\varphi)$ is an orthonormal plane rotation over the angle φ in the (p, q) plane, and defined by $(\cos\varphi, -\sin\varphi, \sin\varphi, \cos\varphi)$ in the (pp, pq, qp, qq) positions of the $n \times n$ identity matrix.

The index pairs are choosen in the cyclic–by–row manner

$$(p, q) = (1, 2)(1, 3)(1, .n)(2, 3) .(2, .n) . (n-1, n) . \qquad (1)$$

The execution of all $N = n(n-1)/2$ index pairs (p, q) according to (1) is called a sweep. Since the matrices $\boldsymbol{A}^{(h)}$ remain symmetric for all h, it is sufficient to work with the upper triangular part of the matrix throughout the algorithm

Defining the off–diagonal quantity $S^{(h)}$ by

$$S^{(h)} = \sqrt{\frac{1}{2}\left[\left\|\boldsymbol{A}^{(h)}\right\|_F^2 - \sum_{i=1}^{n}\left(a_{ii}^{(h)}\right)^2\right]}, \qquad (2)$$

where $\|\cdot\|_F$ denotes the Frobenius norm, the execution of a similarity transformation $\boldsymbol{A}^{(h+1)} = \boldsymbol{J}_{pq}(\theta_h)\,\boldsymbol{A}^{(h)}\,\boldsymbol{J}_{pq}^T(\theta_h)$ yields:

$$\left[S^{(h+1)}\right]^2 = \left[S^{(h)}\right]^2 - \left[\left(a_{pq}^{(h)}\right)^2 - \left(a_{pq}^{(h+1)}\right)^2\right] \tag{3}$$

Obviously the maximal reduction of $S^{(h)}$ is obtained if $a_{pq}^{(h+1)} = 0$. Therefore,

$$\lim_{h\to\infty} S^{(h)} \to 0 \qquad \Leftrightarrow \qquad \lim_{h\to\infty} \boldsymbol{A}^{(h)} \to \mathrm{diag}(\lambda_1, \ldots, \lambda_n)\,. \tag{4}$$

Without loss of generality we will drop the index h and only consider the 2×2 symmetric EVD subproblem

$$\begin{bmatrix} a'_{pp} & a'_{pq} \\ a'_{pq} & a'_{qq} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T \tag{5}$$

in the sequel.

Rewriting a'_{pq} gives us:

$$a'_{pq} = \frac{1}{2}\left[(2a_{pq})\cos 2\theta - (a_{qq} - a_{pp})\sin 2\theta\right] \tag{6}$$

Solving $a'_{pq} = 0$ for the above gives us the optimal angle of rotation θ_{opt} for which maximal reduction is achieved.

$$\theta_{opt} = \frac{1}{2}\arctan(\tau) \tag{7}$$

where $\tau = \frac{2a_{pq}}{a_{qq}-a_{pp}}$, and where the range of θ_{opt} is limited to $|\theta_{opt}| \leq \frac{\pi}{4}$.

2.2 Approximate rotations

Using approximate rotations enables the reduction of the complexity of the vectorization and the rotation mode [14, 2, 10]. For a reduction of the off–diagonal quantity S it is not necessary to meet $a'_{pq} = 0$, but it is sufficient, when using an approximate angle θ_{app}, that

$$\left|a'_{pq}\right| = |d| \cdot |a_{pq}| \text{ with } 0 \leq |d| < 1\,, \tag{8}$$

where the reduction d is a function of the approximative angle θ_{app} and the matrix data, given by:

$$d(\theta_{app}, \tau) = \cos 2\theta_{app} - \frac{1}{\tau}\sin 2\theta_{app}\,. \tag{9}$$

This is the basis for the design of approximate rotations, i.e. rotations that meet (8).

The approach used is to construct a set $\mathcal{F} = \{\boldsymbol{F}_0, \boldsymbol{F}_{-1}, \ldots\}$ of fast orthonormal μ–rotations, where $\boldsymbol{F}_k$ implements the rotation over an angle α_k in a minimal number of shift-add operations. The angles form the ordered set of approximation angles $\mathcal{A} = \{\alpha_0, \alpha_{-1}, \ldots\}$. Both angle α_k and orthonormal μ–rotation are selected by the integer angle index k, satisfying $k \leq 0$. In section 3, we will present methods to construct such fast orthonormal μ–rotations and how to determine the optimal angle index k.

Using approximate rotations must be paid by an increase in the number of required sweeps. It has been shown in [2, 10, 11], however, that the overall cost, in terms of the number of shift-add operations, obtained by using approximate rotations is significantly lower as compared to the exact Jacobi method.

3 METHODS FOR ORTHONORMAL ROTATIONS

We define the close-to-orthonormal μ-rotational $\boldsymbol{F}$ given by the 2×2 rotation matrix

$$\boldsymbol{F} = \begin{bmatrix} \hat{c} & -\sigma\hat{s} \\ \sigma\hat{s} & \hat{c} \end{bmatrix} = \hat{m} \cdot \boldsymbol{J}(\theta) \tag{10}$$

with $\hat{m}$ being the scaling factor of the matrix, given by

$$\hat{m} = \sqrt{\hat{c}^2 + \hat{s}^2} = 1 + \varepsilon\,. \tag{11}$$

The $\hat{c}, \hat{s}$ are pairwise approximations of a sine / cosine pair, satisfying $0 \leq \hat{s} \leq \hat{c} \leq 1$, and chosen such that

1. the multiplication with $\hat{c}$ and $\hat{s}$ is cheap to compute, or that the combined evaluation of the rotation has a cheap implementation, basically only a small number of shift and add operations.
2. the error in scaling ε is smaller than the required accuracy. When this is the case, the effect of the error in the scaling (orthonormality) is overshadowed by the rounding error in the computation. Hence the term "close-to-orthonormal" applies for this type of rotations.

The angle of rotation $\theta = \sigma \cdot \alpha$ is determined by the direction of rotation $\sigma \in \{-1, +1\}$, signifying clockwise or counterclockwise rotation, and by the absolute angle of rotation α, with $\alpha < \pi/2$, which is fixed through the choice of the $\hat{c}, \hat{s}$ pair as:

$$\alpha = \arctan\left(\frac{\hat{s}}{\hat{c}}\right) \tag{12}$$

We have at our disposal a number of methods to systematically arrive at $\hat{c}, \hat{s}$ pairs of varying accuracy and for varying angles of rotation (= varying angle index k). We will present two classes of methods for orthonormal μ-rotations, and select four methods from these, taking into account the cost involved for performing such rotations.

3.1 Unscaled orthonormal μ-rotations

The simplest of the methods is the method I rotation, which is the same as the CORDIC μ–rotation, with

$$\begin{aligned} \hat{c} &= 1 \\ \hat{s} &= 2^k \\ \hat{m} &= (1 + 2^{2k})^{\frac{1}{2}} \end{aligned} \tag{13}$$

where k, with $k \leq 0$ and integer, is the angle index. The smaller the value of k, the smaller the angle, and the more the scaling factor $\hat{m}$ approaches unity.

For (floating-point) computation, with the mantissa size n_m, the maximum round-off error is equal to $\frac{1}{2}2^{-n_m} = 2^{-(n_m+1)}$.

Hence we define the working limit G_I for the angle index such that if $k \leq G_I$ then $\hat{m}$ satisfies:

$$1 - 2^{-(n_m+1)} < \hat{m} < 1 + 2^{-(n_m+1)} \tag{14}$$

and the scaling effect of the rotation can be neglected. Solving (14) for the working limit G_I by substituting the actual value of $\hat{m}$ gives us:

$$G_I = \left\lfloor \frac{-n_m}{2} \right\rfloor \tag{15}$$

For example, for $n_m = 32$, we can use this method I for $k = -16, \ldots, -32$ without scaling.

The more accurate method II rotations are given by

$$\begin{aligned} \hat{c} &= 1 - 2^{2k-1} \\ \hat{s} &= 2^k \\ \hat{m} &= (1 + 2^{4k-2})^{\frac{1}{2}} \end{aligned} \tag{16}$$

while the even more accurate method III rotations are given by

$$\begin{aligned} \hat{c} &= 1 - 2^{2k-1} \\ \hat{s} &= 2^k - 2^{3k-3} \\ \hat{m} &= (1 + 2^{6k-6})^{\frac{1}{2}} \end{aligned} \tag{17}$$

These methods have working limits G_{II} and G_{III} respectively, derived in a similar manner to G_I, given by

$$\begin{aligned} G_{II} &= \left\lfloor \frac{-n_m + 2}{4} \right\rfloor \\ G_{III} &= \left\lfloor \frac{-n_m + 6}{6} \right\rfloor \end{aligned} \tag{18}$$

As each of the previous three methods is limited to a certain working limit, new methods must be developed for more and more accurate rotations. An alternative is to use a fixed method of given accuracy and apply additional scaling iterations to achieve the required accuracy.

3.2 Orthonormal μ-rotations with extra scaling iterations

The rotation used is a double rotation of method I, rotating twice over an angle defined by the angle index of $k-1$, resulting in:

$$\begin{aligned} \hat{c} &= (1)^2 - (2^{k-1})^2 &&= 1 - 2^{2k-2} \\ \hat{s} &= 2(2^{k-1}) &&= 2^k \\ \hat{m} &= \sqrt{1 + 2^{2(k-1)}}^2 &&= 1 + 2^{2(k-1)} \end{aligned} \tag{19}$$

Note that the scaling is no longer a square root function. We exploit this fact in the following fast converging scaling sequence.

We define the additional scaling $K_m = \prod_{i=0}^{m} v_i$, where $m \geq 0$ is the number of scaling steps v_i, with

$$\begin{aligned} v_0 &= 1 \\ v_1 &= (1 - 2^{2(k-1)}) \\ v_i &= (1 + 2^{2^i(k-1)}) \quad \text{for } i \geq 2 \end{aligned} \tag{20}$$

These scaling steps appear in the right-hand side of the well-known factorization

$$1 - x^{2^m} = (1-x)(1+x)(1+x^2)(1+x^4) \cdot \cdot (1 + x^{2^{(m-1)}}) \tag{21}$$

after substitution of $x = 2^{(k-1)}$. It is then trivial to show that the overall scaling, $\hat{m}, K_m$ satisfies

$$K_m \cdot \hat{m} = 1 \pm 2^{-2^{(m+1)}(k-1)}. \tag{22}$$

Hence the method IV scaled orthonormal μ-rotation is given by $\hat{s}$ of (19) and the scaling procedure according to (20).

As for the unscaled methods, we can derive the working limit G_{IV}, which is a function of both the accuracy n_m and the number of scaling steps m, as being:

$$G_{IV} = \left\lceil \frac{-(n_m + 1)}{2^{(m+1)}} \right\rceil \tag{23}$$

In a similar way, we can compute the dual form, which is the limit M, being the minimum number of scaling steps m necessary to reach the required accuracy. Without proof, we state that the limit M is given by

$$M = \left\lfloor \log_2 \left(\frac{-(n_m + 1)}{k - 1} \right) \right\rfloor \tag{24}$$

Using this rotation method, and taking the minimum number of scaling steps necessary, as dictated by M, we can provide orthonormal μ-rotations for rotating over large angles and for high accuracy. Note that the rotation itself costs 4 shift-add operations, or two *pairs* of shift-add operations, while the scaling costs M *pairs* of shift-add operations. In general, we can say that the cost of a method IV rotation is given by

$$C_{IV} = (2 + M) \text{ cycles} \tag{25}$$

where one *cycle* is equivalent to a pair of shift-add operations.

For example: for $n_m = 32$, we can construct the rotation for $k = -2$, with the minimum number of scaling steps, $m = M = 3$, as

$$\boldsymbol{F}_{-2}(\sigma) = (1 - 2^{-6})(1 + 2^{-12})(1 + 2^{-24}) \begin{bmatrix} 1 - 2^{-6} & -\sigma(2^{-2}) \\ \sigma(2^{-2}) & 1 - 2^{-6} \end{bmatrix}. \tag{26}$$

3.3 Constructing the set of μ–rotations

The ordered set of approximation angles $\mathcal{A}$, for a given accuracy n_m, is constructed using the aforementioned methods. Table 1 shows when to select which method, depending on the value of the angle index k, as well as the angle, and the cost for rotation and possible scaling.

The orthonormal μ–rotations are chosen such that they satisfy the accuracy condition (14) using the cheapest possible method.

angle index k	method	angle α_k	cost rot.	cost scl.
$-\infty < k \leq G_I$	I	$\arctan\left(2^k\right)$	1	0
$G_I < k \leq G_{II}$	II	$\arctan\left(\frac{2^k}{1-2^{2k-1}}\right)$	2	0
$G_{II} < k \leq G_{III}$	III	$\arctan\left(\frac{2^k-2^{3k-3}}{1-2^{2k-1}}\right)$	3	0
$G_{III} < k \leq 0$	IV	$\arctan\left(\frac{2^k}{1-2^{2k-2}}\right)$	2	M

Table 1: The selection of μ–rotations methods depending on the angle index k, also showing the angle and overall cost

3.4 Determining the optimal approximate rotation

The crucial point for approximate rotations is to find the approximative angle $\theta_{app} = \sigma \cdot \alpha_k$, where $\sigma \in \{-1, +1\}$ is the direction of the rotation and the angle $\alpha_k \in \mathcal{A}$, with the angle index k, is chosen such that $|d(\theta_{app}, \tau)|$ is minimal.

The direction of rotation σ follows from the sign of the optimal angle θ_{opt} and is given by:

$$\sigma = \text{sign}(\tau) = \text{sign}(a_{pq}) \cdot \text{sign}(a_{qq} - a_{pp}) \tag{27}$$

In order to determine the angle index k, we introduce the working domain limit g_k and define the working condition for the angle α_k as being

$$g_{k-1} < |\tau| \leq g_k \tag{28}$$

where g_k is determined such that, when τ satisfies the above then $|d(\tau_k)|\sigma$ is minimal over the set of angles $\mathcal{A}$. The limit g_k follows from the solution of

$$d(g_k, \alpha_k) = -d(g_k, \alpha_{k+1}) \tag{29}$$

which results in

$$g_k = \tan(\alpha_k + \alpha_{k+1}) = \tan\gamma_k \tag{30}$$

where $\gamma_k = \alpha_k + \alpha_{k+1}$ is the working limit in the angle domain.

In [11], for this particular set of approximative angles, and through the use of floating-point arithmetic the search is narrowed down to checking which of three consecutive angles gives us the maximal reduction (minimal $|d|$). Here we show that the selection between two consecutive angles can be done using μ–rotations as follows.

Construct the vector v given by

$$v = \begin{bmatrix} a_{qq} - a_{pp} \\ 2a_{pq} \end{bmatrix}. \tag{31}$$

The angle between v and the x-axis is equal to the required angle $\arctan(\tau) = 2\theta_{opt}$. To test whether $|\tau|$ is greater or smaller than the kth limit, is equivalent to checking whether this angle is resp. greater or smaller in absolute value to the kth limit angle γ_k.

Hence we rotate v over the angle $-\sigma_k = -\sigma(\cdot\ \alpha_k + \alpha_{k+1}\ )$, to obtain $v' = [\,v'_x\ v'_y\,]^T$. When the y-component v'_y is of the *same* sign as v_y, then the angle between v and the x-axis is larger than γ_k, and we will select α_{k+1} for the approximation angle. If, however, it is of *opposite* sign to v_y, then the angle is smaller, and we select α_k. The rotation over $-\sigma_k$ is performed as two consecutive rotations over α_k and α_{k+1}, implemented as fast orthonormal μ–rotations

$$v' = \boldsymbol{F}_{k+1}(-\sigma) \cdot \boldsymbol{F}_k(-\sigma) \cdot v \tag{32}$$

The same method can be adapted to testing which one of three consecutive angles to use. For this we compute v' and v'' according to

$$\begin{aligned} v' &= \boldsymbol{F}_{k+1}(-\sigma) \cdot \boldsymbol{F}_k(-\sigma) \cdot v \\ v'' &= \boldsymbol{F}_{k-1}(-\sigma) \cdot \boldsymbol{F}_k(-\sigma) \cdot v \end{aligned} \tag{33}$$

and choose the correct angle of rotation, using the selection tree, based on the resulting signs t_y, t''_y

$$\begin{array}{ll} \alpha_{k+1} & \text{if } \mathrm{sign}(v_y) = \mathrm{sign}(v'_y) \\ \alpha_{k-1} & \text{if } \mathrm{sign}(v_y) \neq \mathrm{sign}(v''_y) \\ \alpha_k & \text{otherwise} \end{array} \tag{34}$$

If the intermediate results of the rotations are re-used, this selection mechanism costs at most three orthonormal μ–rotations.

The resulting total reduction $|\,d_{app}(\tau)\,|\,\theta$, using the above method for determining the angle index k is shown in figure 1 for an example set of angles $\mathcal{A}$ with 32-bit accuracy.

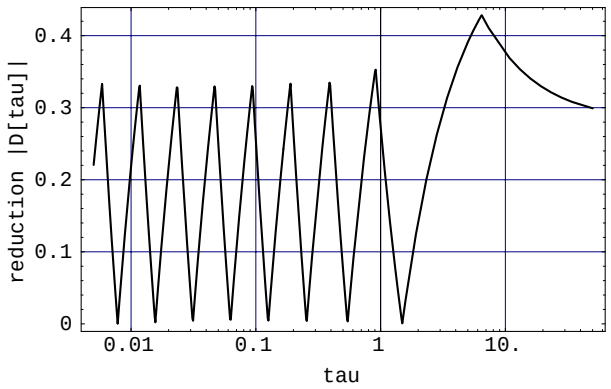

Figure 1: The total reduction as a function of τ, for the set of angles with 32-bit accuracy

4 ADAPTATION OF THE ROTATION METHODS

The nature of the Jacobi method guarantees that the less complex μ–rotation methods are used as the matrix becomes more and more diagonal dominant during the course of the algorithm, i.e. ultimately only method I (the least complex method) is required. Therefore, a Jacobi method realized as a wavefront processor array automatically speeds up the computations if the μ–rotation method is adapted according to table 1 as the number of sweeps increases. This behaviour is illustrated for a random matrix of dimension $n = 6$. The values of k computed during the algorithm are shown in figure 2. Each of the 11 required sweeps contains $N = 15$ rotations (i.e. $N = 15$ values of k). Obviously, the optimal angle index k decreases (i.e. absolute value increases) during the algorithm. Only the first 4 sweeps require method IV. The last 3 sweeps only require method I, i.e. the least complex orthonormal μ–rotation. In figure 3 ($r = 1$, solid line) the reduction of the off–diagonal norm is shown vs. the sweep number.

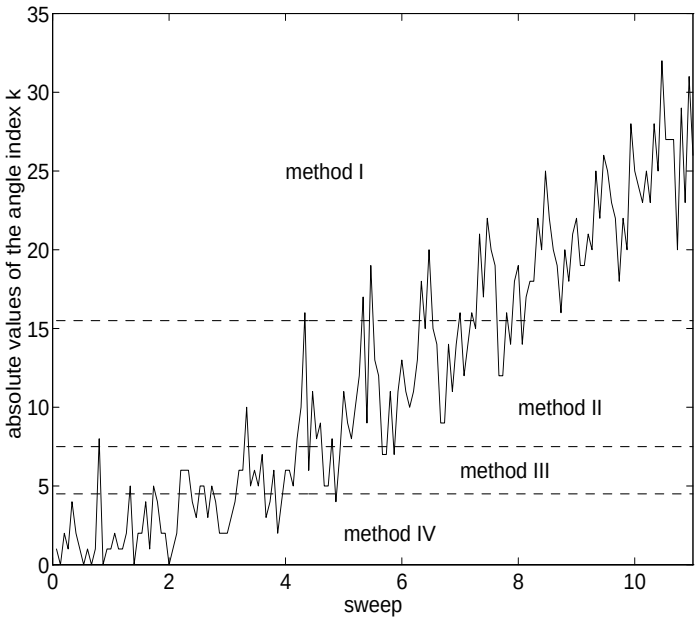

Figure 2: Values of optimal angle index k during the course (sweeps) of the Jacobi method (random matrix $n = 6$, i.e. each sweep consists of $N = 15$ rotations/angle indices)

Obviously, using only one μ–rotation per approximate rotation shows no ultimate quadratic convergence (see figure 3, solid line) as does the Jacobi method with exact rotations. As shown in [11] the quadratic convergence of the Jacobi method can be regained by adapting (increasing) the number of orthonormal μ–rotations executed per plane rotation $\boldsymbol{J}_{pq}$. Increasing the number of orthonormal μ–rotations executed, i.e., executing r μ–rotations with optimal angle index per plane rotation, increases the accuracy of the the approximate rotation. Therefore, the conditions for quadratic convergence [10] can be met as the diagonalization advances. Figure 3 shows the off–diagonal norm vs. sweeps. The Jacobi method

is executed with exact rotations (dashed line), one μ–rotation per similarity transformation (solid line) and an adaptive number r of μ–rotations per similarity transformation (dash–dotted line). The adaptation scheme used for this figure was $r = \lfloor |k_{mean}| / 10 \rfloor$, where k_{mean} is the mean value of the optimal angle indices in the respective sweep. Note that the increase of r comprises a decreasing cost for the μ–rotations as the simpler rotation methods are used for smaller k.

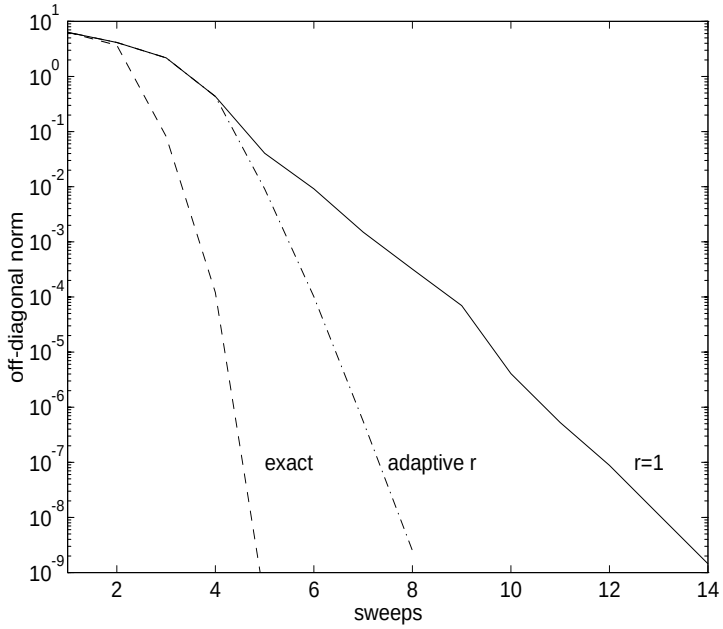

Figure 3: Off–diagonal norm vs. sweeps

5 CONCLUSIONS

In this paper a Jacobi–type algorithm for computing the EVD of a symmetric matrix was presented. It uses different types of orthonormal μ–rotations as approximate rotations. These orthonormal μ–rotations are distinguished by decreasing complexity for decreasing angles of rotation. The evaluation of the angle index k, which determines the approximate rotation angle and the used type of orthonormal μ–rotation, can also be executed by at most three (unscaled) μ–rotations. Therefore, the entire Jacobi–type method can be performed by the execution of μ–rotations (pairs of shift-add operations, respectively), i.e., it can completely be executed on a floating point CORDIC architecture. Thus, it is highly suitable for a VLSI implementation.

Furthermore, the nature of the Jacobi method (i.e. increasing diagonalization with decreasing rotation angles) supports the use of different types of orthonormal μ–rotations as well as an adaptation of the accuracy of the approximate rotation, since in both cases the simpler types of μ–rotations are used as the required rotation angle (angle index k) decreases.

Finally, note that the presented methods also apply to the SVD of a rectangular matrix if the SVD problem is mapped to a symmetric EVD problem with increased dimension [6, 8].

ACKNOWLEDGEMENTS

The work of the first author was performed while on leave at the Delft University of Technology. The research fellowship of the Delft University of Technology is acknowledged. Thanks are also to the Network Theory Group (Prof. P. Dewilde, Prof. E. Deprettere) of the Delft University of Technology for the hospitality and the great time in Delft.

REFERENCES

[1] R.P. Brent and F.T. Luk. The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays. *SIAM J. Sci. Stat. Comput.*, 6:69–84, 1985.

[2] J.P. Charlier, M. Vanbegin, and P. van Dooren. On efficient implementations of Kogbetliantz's algorithm for computing the singular value decomposition. *Numer. Math.*, 52:279–300, 1988.

[3] J.-M. Delosme. Bit-level systolic algorithm for the symmetric eigenvalue problem. In *Proc. Int. Conf. on Application Specific Array Processors*, pages 771–781, Princeton (USA), 1990.

[4] M.D. Ercegovac and T. Lang. Redundant and on–line CORDIC: Application to matrix triangularization and SVD. *IEEE Trans. on Computers*, 39:725–740, 1990.

[5] W.M. Gentleman. Least squares computations by Givens rotations without square roots. *J. Inst. Maths Applics*, 12:329–336, 1973.

[6] G.H. Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, second edition, 1989.

[7] J. Götze. Parallel methods for iterative matrix computations. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 233–236, Singapore, 1991.

[8] J. Götze. CORDIC–based approximate rotations for SVD and QRD. In *Proc. European Signal Processing Conference*, Edinburgh (Scotland), 1994.

[9] J. Götze. Monitoring the stage of diagonalization in Jacobi–type methods. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, page to appear, Adelaide (Australia), 1994.

[10] J. Götze. On the parallel implementation of Jacobi's and Kogbetliantz's algorithm. *to appear SIAM J. Sci. & Stat. Comput.*, 1994.

[11] J. Götze, S. Paul, and M. Sauer. An efficient Jacobi–like algorithm for parallel eigenvalue computation. *to appear IEEE Trans. on Computers*, 1993.

[12] J. Götze and U. Schwiegelshohn. A square root and division free Givens rotation for solving least squares problems on systolic arrays. *SIAM J. Sci Stat. Comput.*, 12:800–807, 1991.

[13] G.J. Hekstra and E.F. Deprettere. Floating point CORDIC. In *11th Symp. on Computer Arithmetic*, Winsor (Canada), 1993.

[14] J.J. Modi and J.D. Pryce. Efficient implementation of Jacobi's diagonalization method on the DAP. *Numer. Math.*, 46:443–454, 1985.

[15] W. Rath. Fast Givens rotations for orthogonal similarity transformations. *Numer. Math.*, 40:47–56, 1982.