FAST SIMULTANEOUS ORTHOGONAL REDUCTION TO TRIANGULAR MATRICES

I. V. OSELEDETS *, D. V. SAVOSTIANOV †, AND E. E. TYRTYSHNIKOV ‡

Abstract. A new algorithm is presented for simultaneous reduction of a given finite sequence of matrices to upper triangular matrices by means of orthogonal transformations. The reduction is performed through a series of deflation steps, where each step contains a *simultaneous eigenvalue problem* being a direct generalization of the generalized eigenvalue problem. To solve the latter, a fast variant of the Gauss–Newton algorithm is proposed with some results on its local convergence properties (quadratic for the exact and linear for the approximate reduction) and numerical examples of its effectiveness proving that our method is quite competitive with other known approaches.

Key words. Simultaneous reduction of matrices, fast algorithms, convergence estimates

1. Introduction. Numerical algorithms for simultaneous reduction of several matrices to a specific structured form (like triangular or diagonal) by means of a one and the same transformation applied to all given matrices is one of the most challenging problems in matrix analysis [3, 5, 1]. The transformations of interest can be similarity, congruence or equivalence with possible additional constraints. The problem we are going to tackle is the following problem of *approximate simultaneous reduction to triangular matrices*:

Problem. *Given $n \times n$ real matrices $A_1, ..., A_r$, find orthogonal $n \times n$ matrices Q and Z such that matrices*

$$B_k = QA_kZ$$

are as upper triangular as possible.

This problem arises, for example, during the computation of the Canonical Decomposition in tensor algebra, see [1] and references therein.

In the case of two matrices ($r = 2$) such a transformation is well studied, can be constructed explicitly, and often referred to as the *generalized Schur decomposition*. It justifies the name of *simultaneous generalized Schur decomposition*(SGSD) used for those decompositions in the case of $r > 2$ [1]. It is clear that arbitrary matrices A_k may not admit such a reduction with a good accuracy, so we impose on A_k the following *existence assumption*: real matrices A_k are such that orthogonal matrices Q and Z exist providing the equations

$$QA_kZ = T_k + E_k,$$

where T_k are upper triangular and the residue matrices E_k satisfy

$$(\sum_{k=1}^{r}(||E_k||_F)^2)^{1/2} = \varepsilon,$$

and ε is considered to be "small". When $\varepsilon = 0$ we will say that matrices A_1, ..., A_r possess an *exact SGSD*, otherwise an *approximate SGSD*, or *ε-SGSD*.

*Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow(ivan@bach.inm.ras.ru).

†Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow(draug@bach.inm.ras.ru).

‡Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow(tee@inm.ras.ru).

Numerical algorithms for simultaneous reduction of matrices are often obtained as generalizations of the corresponding algorithms for one matrix or a pair of matrices. We should mention two basic approaches for solving problems of simultaneous reduction of matrices. The first is a very general approach of [2], efficient numerical implementation of which requires the integration a of possibly very stiff ODE, and that is a very hard problem. The second approach is a Jacobi-type algorithm, where each step consists of two (or one) Jacobi rotations Q_i and Z_i that are sought to minimize the merit function of the form

$$\sum_{k=1}^{r} ||Q_i(QA_kZ)Z_i||_{LF}^2,$$

where $||\ ||_{LF}$ is a sum of squares of all elements in the strictly lower triangular part of the matrix, and Q and Z are orthogonal matrices that are already obtained at the previous steps. This algorithm was proposed in [1]; it was shown that at each step it requires rooting a polynomial of order 8.

In the case of $r = 2$ (a pair of matrices), a well-developed tool for the computation of the generalized Schur decomposition is the QZ-algorithm. Its generalization to the case of several matrices was developed in [4] where it was called an extended QZ-algorithm. However, the QZ-algorithm is efficient only when the shifts are used. It is not clear how the technique of shifts can be generalized to the case $r > 2$.

In this paper we propose a new algorithm to compute the simultaneous generalized Schur decomposition. We reach the purpose through a sequence of *deflation steps*. At each step we solve an optimization problem in $2n + r$ variables which is a direct generalization of the generalized eigenvalue problem (we call it *simultaneous eigenvalue problem*). The main ingredient of the algorithm is our fast version for the Gauss-Newton method applied to the latter problem. We suggest an inexpensive update technique for the matrices at successive iterative steps.

We prove the local quadratic convergence in the case of exact SGSD for the given A_1, ..., A_r, and linear convergence in the case of ε-SGSD, the convergence factor being proportional to ε. Numerical experiments confirm the effectiveness of our approach. For example, the computation of the SGSD for 128 matrices of size 128×128 in our examples takes less than a minute.

2. Simultaneous eigenvalue problem and its solution.

2.1. Transformation of the initial problem. Given real matrices A_1, ..., A_r, we want to find orthogonal matrices Q and Z making matrices QA_1Z, ..., QA_rZ as upper triangular as possible. In order to do this, first consider a related problem of finding orthogonal matrices Q and Z such that the matrices QA_iZ are reduced to the following block triangular form:

$$QA_kZ \approx \begin{pmatrix} \lambda_k & v_k^\top \\ 0 & B_k \end{pmatrix}, \tag{2.1}$$

where B_k is a $(n-1) \times (n-1)$ matrix, λ_k is a scalar and v_k is a vector of size $n-1$. As soon as (2.1) is found, we accomplish a *deflation step* reducing the problem to the same but smaller problem for the matrices B_k. Proceeding in the same way with B_k we finally reduce A_k to the upper triangular form. Thus, our main goal is the solution of (2.1).

The approximate equations (2.1) are equivalent to

$$QA_kZe_1 \approx \lambda_k e_1,$$

where e_1 is the first column of the identity matrix. Since Q is orthogonal, we can multiply these equations from the left by $Q^\top$ without changing the residue:

$$A_k Z e_1 \approx \lambda_k Q^\top e_1.$$

Now, introducing two vectors $x = Ze_1$ and $y = Q^\top e_1$, we come up with the following overdetermined system of equations to be solved in the least squares sense:

$$A_k x = \lambda_k y. \tag{2.2}$$

If $r = 2$ then this is a well-known generalized eigenvalue problem for a pair of matrices A_1, A_2. So we may refer to the problem (2.2) as *simultaneous generalized eigenvalue problem*, or simply *simultaneous eigenvalue problem* (SEP). Since x, y, and λ_k are defined up to the multiplication by a constant, we use the following normalizing condition:

$$||x||_2 = 1.$$

The simultaneous eigenvalue problem (2.2) is the key component of our algorithm. How it can be solved will be described in the next section. If we consider this solver as a "black box", then the algorithm for calculation of the SGSD reads:

ALGORITHM 2.1.

Given r real matrices A_1, ..., A_r of size $n \times n$, find orthogonal matrices Q and Z such that the matrices QA_kZ are as upper triangular as possible:

1. *Set*

$$m = n,\ B_i = A_i,\ i = 1, ..., r,\ Q = Z = I.$$

2. *If $m = 1$ then stop.*
3. *Solve the simultaneous eigenvalue problem*

$$B_k x = \lambda_k y,\ k = 1, ..., r.$$

4. *Find $m \times m$ Householder matrices Q_m, Z_m such that*

$$x = \alpha_1 Q_m^\top e_1,\ y = \alpha_2 Z_m e_1.$$

5. *Calculate C_k as $(m-1) \times (m-1)$ submatrices of matrices $\widehat{B}_k$ defined as follows:*

$$\widehat{B}_k = QB_kZ = \begin{pmatrix} \alpha_k & v_k^\top \\ \varepsilon_k & C_k \end{pmatrix}.$$

6. *Set*

$$Q \leftarrow \begin{pmatrix} I_{(n-m)\times(n-m)} & 0 \\ 0 & Q_m \end{pmatrix} Q,\ Z \leftarrow Z \begin{pmatrix} I_{(n-m)\times(n-m)} & 0 \\ 0 & Z_m \end{pmatrix}.$$

7. *Set $m = m - 1$, $B_k = C_k$ and proceed to the step 2.*

2.2. Gauss-Newton algorithm for the simultaneous eigenvalue problem. In this section we present an algorithm for the SEP (2.2). To begin with, let us write (2.2) element-wise as follows:

$$\sum_{j=1}^{n} A_{ij}^{k} x_j = \lambda_k y_i. \tag{2.3}$$

Now introduce $r \times n$ matrices

$$(a_j)_{ki} = A_{ij}^{k}, k = 1, ..., r, \ , i = 1, ..., n \ , j = 1, ..., n,$$

and a column vector $\lambda = [\lambda_1, \ ..., \ \lambda_r]^\top$. Then (2.3) becomes

$$\sum_{j=1}^{n} x_j a_j = \lambda y^\top. \tag{2.4}$$

The set of equations (2.4) can be considered as an overdetermined system of nonlinear equations. To solve it, we derive a variant of the Gauss-Newton method and propose a fast scheme to implement it. Then we obtain some convergence estimates.

The idea behind the Gauss-Newton method is to linearize the system (as in the standard Newton method) producing an overdetermined linear system and then solve it in the least squares sense.

On the linearization of (2.4) at some point (x, λ, y), we obtain the following overdetermined system:

$$\sum_{j=1}^{n} \widehat{x}_j a_j = \triangle\lambda y^\top + \lambda \triangle y^\top, \quad \widehat{x} = x + \triangle, \quad ||\widehat{x}||_2 = 1. \tag{2.5}$$

At each iterative step, the system (2.5) has to be solved in the least squares sense. To cope with this problem, let us observe, first of all, that the unknowns $\triangle y$ and $\triangle\lambda_k$ can be easily excluded in the following way. To this end, find a $n \times n$ Householder matrix H such that

$$Hy = he_1,$$

and a $r \times r$ Householder matrix C such that

$$C\lambda = ce_1.$$

(In the above equations, e_1 denotes the first column in the indentity matrices of different sizes.) Premultiplying (2.5) by C and postmultiplying it by $H^\top$, we obtain

$$\sum_{j=1}^{n} \widehat{x}_j \widehat{a}_j = ce_1 \triangle\widehat{y}^\top + h\triangle\widehat{\lambda} e_1^\top, \tag{2.6}$$

where

$$\widehat{a}_j = Ca_jH^\top, \quad \triangle\widehat{y} = H\triangle y, \quad \triangle\widehat{\lambda} = C\triangle\lambda.$$

The equivalence of (2.6) and (2.5) follows from the orthogonality of H and C. In particular, the coefficients $\widehat{x}_j$ are the same in the both problems.

Now, the problem (2.6) is split into two independent problems. To find $\widehat{x}$, one should minimize

$$||\sum_{j=1}^{n} b_j x_j||_F^2, \quad ||x|| = 1, \tag{2.7}$$

where the matrices b_j are obtained from $\widehat{a}_j$ by replacing the elements in the first row and column by zeroes. Once $\widehat{x}$ is found, $\triangle\widehat{y}$ and $\triangle\widehat{\lambda}$ can be determined from the equations

$$(\sum_{j=1}^{n} \widehat{x}_j \widehat{a}_j)_{k1} = h\triangle\widehat{\lambda}_k, \quad k = 2, ..., r, \qquad (\sum_{j=1}^{n} \widehat{x}_j \widehat{a}_j)_{1i} = c\triangle\widehat{y}_i, \quad i = 2, ..., n.$$

For the two unknowns $\triangle\widehat{y}_1$ and $\triangle\widehat{\lambda}_1$, we have only one equation, so one of these unknowns can be chosen arbitrary. However, this approach requires an explicit computation of the Householder matrices and evaluation of $\widehat{a}_j$. It will be shown later that $\widehat{x}$ can be computed without any reference to the Householder matrices. Therefore, we need a way to find the new λ and y directly from the known $\widehat{x}$. Having obtained the new $\widehat{x}$, we propose to evaluate y and λ by the power method as follows:

$$\widetilde{\lambda} = by, \quad \widetilde{y} = b^\top \lambda, \tag{2.8}$$

where

$$b = \sum_{j=1}^{n} \widehat{x}_j a_j.$$

The problem (2.7) is, in fact, a problem of finding the minimal singular value of a matrix

$$B = [\mathrm{vec}(b_1), ..., \mathrm{vec}(b_n)],$$

where the operator vec transforms a matrix into a vector taking the elements column-by-column.

Therefore, $\widehat{x}$ is an eigenvector (normalized to have a unit norm) corresponding to the minimal eigenvalue of the $n \times n$ matrix $\Gamma = B^\top B$:

$$\Gamma\widehat{x} = \gamma_{\min}\widehat{x}.$$

This matrix Γ plays the key role in the solution process. Its elements are given by

$$\Gamma_{sl} = (b_s, b_l)_F$$

where $(\cdot, \cdot)_F$ is the Frobenius (Euclidian) scalar product of matrices. To calculate the new vector $\widehat{x}$, we need to find the minimal eigenvalue and the corresponding eigenvector of the matrix Γ.

The solution of the problem (2.5) consists of the two parts:

1. Calculation of the matrix Γ.
2. Finding the minimal eigenvalue and the corresponding eigenvector of the matrix Γ.

Since only one eigenvector for Γ is to be found, we propose to use the shifted inverse iteration using x from the previous iteration as an initial guess. The complexity is then $\mathcal{O}(n^3)$.

Let us estimate the number of arithmetic operations required for the step 1. The straitforward implementation of this step includes $\mathcal{O}(n^2r + nr^2)$ (calculation of b_j) + $\mathcal{O}(n^2rn)$(calculation of the $B^\top B$) arithmetic operations. The total cost of the step 1 is

$$\mathcal{O}(n^3r + n^2r + nr^2).$$

However, Γ can be computed a way more efficiently without the explicit computation of the Householder matrices.

2.2.1. Calculation of the matrix Γ. In this section we suggest an efficient method to acquire the entries of Γ. Recall that

$$\Gamma_{sl} = (b_s, b_l)_F, \quad i, j = 1, ..., n.$$

Thus, we need the scalar products $(b_s, b_l)_F$. From the definition of b_j it follows that b_j and $\widehat{a}_j$ are connected in the following way:

$$b_j = \widehat{a}_j - \widehat{a}_j e_1 e_1^\top - e_1 e_1^\top \widehat{a}_j + (\widehat{a}_j)_{11} e_1 e_1^\top.$$

Hence, the required scalar products are expressed as

$$(b_s, b_l)_F = (\widehat{a}_s, \widehat{a}_l)_F - (\widehat{a}_s e_1, \widehat{a}_l e_1) - (\widehat{a}_s^\top e_1, \widehat{a}_l^\top e_1) + (\widehat{a}_s)_{11}(\widehat{a}_l)_{11}.$$

Taking into account that $\widehat{a}_j = C a_j H^\top$, we have

$$\Gamma_{sl} = (b_s, b_l)_F = (a_s, a_l)_F - \frac{(a_s y, a_l y)}{||y||^2} - \frac{(a_s^\top \lambda, a_l^\top \lambda)}{||\lambda||^2} + \frac{(a_s y, \lambda)(a_l y, \lambda)}{||y||^2 ||\lambda||^2}. \qquad (2.9)$$

A fast computation of Γ can be based on (2.9). Note also that

$$(a_s, a_l)_F = \sum_{ki} (a_s)_{ki}(a_l)_{ki} = \sum_{ki} (A_k)_{is}(A_k)_{il} = (\sum_{k=1}^{n} A_k^\top A_k)_{sl},$$

therefore, the first summand $(a_s, a_l)_F$ can be computed once and for all y and λ in $\mathcal{O}(n^3 r)$ operations. The cost of computing vectors $a_s y$ and $a_s^\top \lambda$ for all $s = 1, ..., n$ is $\mathcal{O}(n^2 r + r^2 n)$. The cost of computing scalar products $(a_s y, a_l y)$ and $(a_s^\top \lambda, a_l^\top \lambda)$ is of the same order. The total complexity of computing Γ is

$$\mathcal{O}(n^3 r)$$

operations on the zero (initialization) step for the given matrices $A_1, ..., A_k$ plus

$$\mathcal{O}(n^2 r + nr^2)$$

operations for each y and λ arising during iterations.

Now we are ready to describe the algorithm for solving the simultaneous eigenvalue problem.

ALGORITHM 2.2. *Given a sequence of $n \times n$ matrices $A_1, ..., A_r$ and an initial approximation to the solution of the SEP (2.2) x^0, y^0, λ^0, proceed as follows:*

1. *Set $k = 0$ and calculate the initial Gram matrix*

$$\Gamma_0 = \sum_{i=1}^{r} A_i^\top A_i.$$

.

2. *If converged then stop, else continue.*
3. *Calculate the vectors $a_s y^k$ and $a_s \lambda^k$ for all $s = 1, ..., n$.*
4. *Calculate the matrix Γ using the formula (2.9).*
5. *Set x^{k+1} to the eigenvector corresponding to the minimal eigenvalue of Γ.*
6. *Calculate y^{k+1} and λ^{k+1} from (2.8).*
7. *Increase k by 1 and proceed to the step 2.*

It is important to note that the matrix Γ can be updated fast during the work of Algorithm (2.1). Indeed, the most "hard" work is the calculation of

$$\Gamma_0 = \sum_{k=1}^{r} B_k^\top B_k.$$

After the Q-Z transformation of each B_k, Γ_0 becomes

$$\sum_{k=1}^{r} \widehat{B}_k^\top \widehat{B}_k = \sum_{k=1^r} (Q_m B_k Z_m)^\top Q_m B_k Z_m = Z_m^\top \Gamma_0 Z_m.$$

We need to calculate

$$\widehat{\Gamma}_0 = \sum_{k=1}^{r} C_k^\top C_k,$$

where C_k is an $(n-1) \times (n-1)$ leading submatrix of $\widehat{B}_k$ starting from position (2,2). It is easy to see that

$$(C_k^\top C_k)_{ij} = (\widehat{B}_k^\top \widehat{B}_k)_{(i+1)(j+1)} - (\widehat{B}_k)_{i1}(\widehat{B}_k)_{j1}, \quad i = 1, ..., n-1, \ j = 1, ..., n-1.$$

Consequently,

$$(\widehat{\Gamma}_0)_{ij} = (Z_m^\top \Gamma_0 Z_m)_{(i+1)(j+1)} - \sum_{k=1}^{r} (\widehat{B}_k)_{i1}(\widehat{B}_k)_{j1}, \quad i = 1, ..., n-1, \ j = 1, ..., n-1.$$

The complexity of this update is $\mathcal{O}(n^2 r)$.

It remains to analyze the convergence properties of the algorithm.

3. Convergence. Assume that x^*, y^* and λ^* solve the nonlinear minimization problem

$$\sum_{k=1}^{r} ||A_k x - \lambda_k y||^2 \to \min, \tag{3.1}$$

$$||x||_2 = 1,$$

and

$$A_k x^* = \lambda_k^* y^* + \varepsilon_k. \tag{3.2}$$

Suppose we have constructed approximations

$$y = y^* + \delta y,$$

$$\lambda = \lambda^* + \delta\lambda,$$

and then compute the new approximation x to x^* using the Algorithm (2.1). What can we say about $||x - x^*||$?

Recall that (3.2) can be written in terms of matrices a_j; so (2.6) takes on the form

$$\sum_{j=1}^{n} x_j^* a_j = \lambda^* (y^*)^\top + \varepsilon, \tag{3.3}$$

where the residue ε is inserted. Also we will need the normalized vectors

$$\widetilde{y} = \frac{y}{||y||}, \widetilde{\lambda} = \frac{\lambda}{||\lambda||}, \widetilde{y^*} = \frac{y^*}{||y^*||}, \widetilde{\lambda}^* = \frac{\lambda^*}{||\lambda^*||}.$$

The vector x is an eigenvector of the matrix Γ defined by (2.9). Denote by Γ^* the matrix for y^* and λ^*. Then the following lemma holds true.

LEMMA 3.1.

The following inequality holds:

$$|(\Gamma x^* - \Gamma^* x^*)_s| \leq \; ||a_s||(||y^*||\;||\lambda^*||(4\;||\delta\widetilde{y}||^2 + ||\delta\widetilde{y}||\;||\delta\widetilde{\lambda}|| + 4\;||\widetilde{\lambda}||^2) + ||\varepsilon||(||\delta\widetilde{y}|| + ||\delta\widetilde{\lambda}||)) + \mathcal{O}(\delta^3 + ||\varepsilon||\delta^2),$$

where $\delta = \max(||\delta y||, ||\delta\lambda||)$.

Proof. Using the definition of the matrix Γ and the equality (3.3) we have

$$((\Gamma - \Gamma_0)x^*)_s = -(a_s\widetilde{y}, \sum_{l=1}^{n} x_l^* a_l \widetilde{y}) - (a_s^\top \widetilde{\lambda}, \sum_{l=1}^{n} x_l^* a_l^\top \widetilde{\lambda}) + (a_s\widetilde{y}, \widetilde{\lambda})(a_s\widetilde{y}, \widetilde{\lambda})(\sum_{l=1}^{n} x_l^* a_l \widetilde{y}, \widetilde{\lambda}) =$$

$$= -(a_s\widetilde{y}, \lambda^*)(y^*, \widetilde{y}) - (a_s\widetilde{y}, \varepsilon\widetilde{y}) - (a_s^\top\widetilde{\lambda}, y^*)(\lambda^*, \widetilde{\lambda}) - (a_s^\top\widetilde{\lambda}, \varepsilon^\top\widetilde{\lambda}) +$$

$$+(a_s\widetilde{y}, \widetilde{\lambda})((y^*, \widetilde{y})(\lambda^*, \widetilde{\lambda}) + (\varepsilon\widetilde{y}, \widetilde{\lambda})).$$

Now set

$$\widetilde{y} = \widetilde{y^*} + \delta\widetilde{y}, \widetilde{\lambda} = \widetilde{\lambda}^* + \delta\widetilde{\lambda}.$$

Since $||\widetilde{y}|| = ||\widetilde{y^*}|| = 1$

$$(\delta\widetilde{y}, y^*) = -2\;||y||^*\;||\delta\widetilde{y}||^2,$$

we obtain the following first order terms (denote them by Φ_1):

$$\Phi_1 = -(a_s\delta\widetilde{y}, \varepsilon\widetilde{y^*}) - (a_s\widetilde{y^*}, \varepsilon\delta\widetilde{y}) - (a_s^\top\widetilde{\lambda}^*, \varepsilon^\top\delta\widetilde{\lambda}) - (a_s^\top\delta\widetilde{\lambda}, \varepsilon^\top\widetilde{\lambda}^*) + (a_s\widetilde{y^*}, \delta\widetilde{\lambda})(\varepsilon\widetilde{y^*}, \widetilde{\lambda}^*) +$$

$$+(a_s\delta\widetilde{y},\widetilde{\lambda}^*)(\varepsilon\widetilde{y}^*,\widetilde{\lambda}^*)+(a_s\widetilde{y}^*,\widetilde{\lambda}^*)((\varepsilon\delta\widetilde{y},\widetilde{\lambda}^*)+(\varepsilon\widetilde{y}^*,\delta\widetilde{\lambda})).$$

Φ_1 is estimated from above by

$$4\ ||a_s||\ ||\varepsilon||\ (||\delta\widetilde{y}||+||\delta\widetilde{\lambda}||).$$

The second order terms are estimated in the same way (we omit terms of order $\mathcal{O}(\delta^2||\varepsilon||)$). We give here only the final result:

$$|\Phi_2|\le||y^*||\ ||\lambda^*||\ ||a_s||\ ||\delta\widetilde{y}||\ ||\delta\widetilde{\lambda}||+4\ ||a_s||\ ||y^*||\ ||\lambda^*||\ (||\delta\widetilde{y}||^2+||\delta\widetilde{\lambda}||^2).$$

To finish the proof, it is left to note that $|(\Gamma-\Gamma^*)x^*)_s|\le(|\Phi_1|+|\Phi_2|)+\mathcal{O}(\delta^3+\delta^2\varepsilon)$. □

Now we can estimate the residue $||x-x^*||$.

THEOREM 3.2.

If x is computed from y and λ using the Algorithm 2.1, x^*, y^* and λ^* are the solution of the minimization problem (3.1) then

$$||x-x^*||\le\frac{1}{\gamma_{n-1}-\gamma_n}\sqrt{\sum_{s=1}^{n}||a_s||^2}\ (6\delta^2||y^*||\ ||\lambda^*||+2\ ||\varepsilon||\ \delta)+\mathcal{O}(\delta^3+||\varepsilon||\delta^2),\quad(3.4)$$

where

$$\delta=\max(||\delta y||,||\delta\lambda||)$$

and γ_n,γ_{n-1} are two smallest eigenvalues of matrix Γ^*.

If we consider matrix Γ as a perturbation of Γ^*, then x is a perturbation of the eigenvector x^*. Using the theorem about the perturbation of the eigenvector of a symmetric matrix, we obtain

$$||x-x^*||\le\frac{1}{\gamma_{n-1}-\gamma_n}||(\Gamma-\Gamma^*)x^*||.$$

Now, applying the inequality (3.4) and taking into account that

$$||\delta\widetilde{y}||\le||\delta y||,\quad||\delta\widetilde{\lambda}||\le||\delta\lambda||,$$

we arrive at (3.4).

The estimate (3.4) fully describes the local convergence of our method. If $||\varepsilon||=0$ (that is, matrices A_k can be exactly reduced to the triangular form), then the convergence is quadratic. In the case of non-zero but sufficiently small $||\varepsilon||$, the convergence is linear, but the convergence speed is proportional to $||\varepsilon||$.

Remark. The numerical experiments show that when $||\varepsilon||$ is small enough, the algorithm converges globally. But at present we have no rigorous formulations of the conditions required and/or sufficient for the algorithm to be globally convergent.

4. Numerical experiments. In this section we present some numerical experiments confirming the efficiency of our method. It was implemented in FORTRAN. The first series of example is created in the following way. We generate random two matrices X and Y of order n and r diagonal matrices Λ_k, $k=1,...,r$, of the same order, and set

$$A_k=X\Lambda_kY,\quad k=1,...,r.$$

The elements of X Y and Λ_k are drawn from the uniform distribution on $[-1,1]$. As it was shown in [1], these sequence of matrices have an exact $SGSD$, because we can find orthogonal Q and Z such that

$$X = QR_1, Y = R_2Z,$$

with R_1 and R_2 being upper triangular. We also corrupt these matrices with multiplicative noise, setting

$$(\widehat{A}_k)_{ij} = (A_k)_{ij}(1 + \sigma\phi),$$

where ϕ are taken from the uniform distribution on $[-1,1]$ and σ is a "noise level". We are interested in the following quantities:

- The convergence speed, its dependence from n,r, and σ.
- The stability: the dependence of the *residue* of the SGSD from σ.

We have observed that the speed of the algorithm does not depend any pronouncedly on σ. We perform two experiments. First we fix r and σ setting them to 10 and 10^{-6} respectively and change n. The timings (in seconds) are given in Table 1.

n	Time
16	0.01
32	0.11
64	1.6
128	14.77
256	210.61

Table 4.1 Timings(in seconds) for the computation of SGCD of 10 n-by-n matrices.

n	Time
16	0.02
32	0.14
64	3.41
128	54.96
256	810.77

Table 4.2 Timings(in seconds) for the computation of SGCD of n n-by-n matrices.

σ	Mean residue	Min residue	Max residue
10^{-16}	$2 \cdot 10^{-15}$	$9 \cdot 10^{-16}$	$5 \cdot 10^{-15}$
10^{-15}	$7 \cdot 10^{-15}$	$1 \cdot 10^{-15}$	$2 \cdot 10^{-14}$
10^{-14}	$3 \cdot 10^{-14}$	$4 \cdot 10^{-15}$	$8 \cdot 10^{-14}$
10^{-13}	$5 \cdot 10^{-14}$	$4 \cdot 10^{-14}$	$8 \cdot 10^{-14}$
10^{-12}	$2 \cdot 10^{-12}$	$4 \cdot 10^{-13}$	$4 \cdot 10^{-12}$
10^{-11}	$6 \cdot 10^{-11}$	$4 \cdot 10^{-12}$	$1 \cdot 10^{-11}$
10^{-10}	$4 \cdot 10^{-10}$	$4 \cdot 10^{-11}$	$1 \cdot 10^{-9}$
10^{-9}	$2 \cdot 10^{-8}$	$4 \cdot 10^{-10}$	$5 \cdot 10^{-8}$
10^{-8}	$4 \cdot 10^{-8}$	$4 \cdot 10^{-9}$	$1 \cdot 10^{-7}$
10^{-7}	$2 \cdot 10^{-7}$	$4 \cdot 10^{-8}$	$7 \cdot 10^{-7}$
10^{-6}	$6 \cdot 10^{-7}$	$4 \cdot 10^{-7}$	$1 \cdot 10^{-6}$
10^{-5}	$2 \cdot 10^{-5}$	$4 \cdot 10^{-6}$	$5 \cdot 10^{-5}$
10^{-4}	$4 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$1 \cdot 10^{-3}$
10^{-3}	$2 \cdot 10^{-3}$	$4 \cdot 10^{-4}$	$6 \cdot 10^{-3}$

Table 4.3 Residues for different noise levels.

In the second experiment we set r to be equal to n. Corresponding timings are given in Table 2.

To check stability we take fixed $r = n = 64$ and vary the noise level. For each noise level 10 test sequences of matrices are generated and the mean, maximal and minimum values of the residue are reported.

5. Conclusion. In this paper a problem of the calculation of the simultaneous generalized Schur decomposition is considered. It is shown that this problem can be reduced to a series of smaller optimization problems which are a direct generalization of the generalized eigenvalue problem; that is why we called it *simultaneous eigenvalue problem*. We have proposed the fast Gauss-Newton algorithm for the solution of the simultaneous eigenvalue problem and shown that the computations can be performed in a cheap way using careful update techniques. The local quasi-quadratic convergence result is obtained. If the number of matrices r is of order n (what frequently happens, if we use the SGSD for the computation of the Canonical Decomposition), then the complexity of the algorithm is $\mathcal{O}(n^4)$ arithmetic operations. The efficiency and robustness of the algorithm was demonstrated by some numerical examples.

REFERENCES

[1] L. De Lathauwer, B. De Moor and J. Vanderwalle, *Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition*, SIAM J. Matrix Anal. Appl., 26(2004), pp. 295-227

[2] M. Chu, *A continues Jacobi-like approach to the simultaneous reduction of real matrices* , Linear Alg. Appl., 147(1991), pp. 75-96.

[3] A. Bunse-Gerstner, R. Byers, V. Mehrmann, *Numerical methods for simultaneous diagonalization*, SIAM J. Matrix Anal. Appl., 14(1993), pp. 927-949

[4] A.-J. Van der Veen and A. Paulraj, *An analytical constant modulus algorithm*, IEEE Trans. Signal Process., 44(1996), pp. 1136-1155

[5] A. Ziehe, M. Kawanabe, S. Hamerling, and K.-R. Müller, *A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation* , Journal of Machine Learning Research, 5(2004), pp. 801-818