94-65

94-65

Yi C., Dehaene J., De Moor B., "Computation of the structured singular value with gradient based optimization algorithms", in *Proc. of the American Control Conference (ACC'95)*, Seattle, Washington, Jun. 1995, pp. 3034-3040., Lirias number: 180040.

Calculation of the structured singular value with gradient based optimization algorithms

Cheng Yi, Jeroen Dehaene and Bart De Moor*
ESAT - Katholieke Universiteit Leuven,
Kardinaal Mercierlaan 94,
B-3001 Leuven (Heverlee), Belgium,
tel 32/16/22 09 31, fax 32/16/22 18 55,
jeroen.dehaene@esat.kuleuven.ac.be
cheng.yi@esat.kuleuven.ac.be
bart.demoor@esat.kuleuven.ac.be

1995 ACC

Abstract

The structured singular value problem, which is a basic problem in robustness analysis and synthesis for control systems with structured uncertainties, can be formulated as an optimization problem over the manifold of unitary matrices with a given structure. We show how geometric optimization methods, such as the steepest ascent method and the conjugate gradient method for optimization on a Riemannian manifold, lead to algorithms with guaranteed convergence to a lower bound for the structured singular value.

Keywords: structured singular value (μ), gradient based optimization.

1 Introduction

The structured singular value μ is an important tool for the robustness analysis of systems with structured uncertainty[1]. The underlying structure of the problem (see [2] for notational conventions) is described by a set $\mathbf{\Delta}$ of block diagonal matrices $\Delta \in \mathbb{C}^{n\times n}$, with s repeated scalar blocks and f full blocks. Repeated scalar blocks have the form $\delta_k I_{r_k}$ where $\delta_k \in \mathbb{C}$ and I_{r_k} denotes the identity matrix in $\mathbb{C}^{r_k\times r_k}$, $k = 1,\ldots,s$. Full blocks are $\Delta_l \in \mathbb{C}^{m_l\times m_l}$, $l = 1,\ldots,f$. Hence, the set $\mathbf{\Delta}$ is defined as:

$$\mathbf{\Delta} = \{\mathrm{diag}[\delta_1 I_{r_1},\ldots,\delta_s I_{r_s},\Delta_1,\ldots,\Delta_f] : \delta_k \in \mathbb{C}, \Delta_l \in \mathbb{C}^{m_l\times m_l}\}.$$

*Jeroen Dehaene is a research assistant with the N.F.W.O. (Belgian National Fund for Scientific Research). Bart De Moor is a senior research associate of the N.F.W.O. The following text presents research results obtained within the framework of the Belgian programme on interuniversity attraction poles (IUAP-17 and IUAP-50) initiated by the Belgian State - Prime Minister's Office - Science Policy Programming. The scientific responsibility is assumed by its authors. The research work was supported by the F.K.F.O. project G0292.95.

Given a matrix $M \in \mathbb{C}^{n\times n}$, and a structure $\mathbf{\Delta}$, the structured singular value is defined by

$$\mu_{\mathbf{\Delta}}(M) = \frac{1}{\min_{\Delta\in\mathbf{\Delta}}\{\bar{\sigma}(\Delta) : \det(I + M\Delta) = 0\}}, \tag{1}$$

where $\bar{\sigma}$ denotes the maximal singular value. Equivalent definitions are [2]

$$\mu_{\mathbf{\Delta}}(M) = \max_{\Delta\in B\mathbf{\Delta}} \rho(\Delta M), \tag{2}$$

where $B\mathbf{\Delta} = \{\Delta \in \mathbf{\Delta} : \bar{\sigma}(\Delta) \leq 1\}$ and ρ denotes the spectral radius,
and

$$\mu_{\mathbf{\Delta}}(M) = \max_{Q\in\mathcal{Q}} \rho(QM), \tag{3}$$

where $\mathcal{Q} = \{Q \in \mathbf{\Delta} : Q^H Q = I_n\}$ is the set of structured unitary matrices.

Remark 1 The presented results also hold for non square matrices M, by extending them with zero rows or columns, making all blocks square.

To solve an optimization problem like (3), one can either apply *general optimization algorithms* using function evaluations and possibly gradient information, to find a locally optimal solution, or one can concentrate directly on the *optimality conditions* for the given problem. If the *optimality conditions* cannot be solved analytically, an iterative scheme is needed to solve them. The power iteration algorithm (PIA) of [2] is actually an iterative scheme to solve the optimality condition of (3) from which a lower bound for $\mu_{\mathbf{\Delta}}$ is then computed. PIA is computationally efficient if it converges. However, up to this date, there is no formal theory about convergence. As a matter of fact, current implementations do not necessarily converge as is demonstrated with an example in section 5. (The continuous time counter part of PIA [3] seems to have better convergence properties but also lacks a convergence proof). In this paper we choose the approach of *general optimization algorithms.* Actually, such an approach may be regarded as just another iterative algorithm to satisfy the equilibrium conditions, but now one with guaranteed convergence. The approach also takes into account the highly regular nature of the constraint in the optimization problem (3), by applying optimization methods on Riemannian manifolds, making use of intrinsic properties of the manifold.
This paper is organized as follows. Section 2 defines the setting for the derivations in the next sections, by introducing some theoretical concepts about the set $\mathcal{Q}$ of structured unitary matrices. In sections 3 and 4 the steepest ascent algorithm (SAA) and the conjugate gradient algorithm (CGA) for the structured singular value problem are derived. Section 5 summarizes some numerical experiments and section 6 gives the conclusion.

2 The manifold $\mathcal{Q}$ of structured unitary matrices

The optimization methods discussed by Smith [5] generalize optimization methods over Euclidean spaces to optimization methods over Riemannian manifolds by making use of intrinsic properties of the manifold. Straight lines are replaced by geodesics. Tangent vectors at one point are adapted to other points by parallel translations along geodesics. In this section we establish some properties of the manifold $\mathcal{Q}$ of structured unitary matrices, which will be needed further on in this paper. If one is willing to accept the formulas for geodesics and parallel translation as God given, most elements of the next theorem are not needed to understand the algorithms derived below. For the areligious or agnostic reader we offer:

Theorem 1 *i) The manifold $\mathcal{Q} = \{Q \in \Delta : Q^H Q = I_n\} = U(n) \cap \Delta$ is a Lie group, where $U(n)$ is the unitary group.* [1]
ii) the tangent space at $Q \in \mathcal{Q}$ equals

$$T_Q\mathcal{Q} = \{T \in \mathbb{C}^{n\times n} | T = SQ, S \in \Delta, S^H + S = 0_n\}. \tag{4}$$

The Lie algebra $\mathcal{L}$ of $\mathcal{Q}$, identified with the tangent space at the identity, is given by

$$\mathcal{L} = \{S \in \Delta | S^H + S = 0\}. \tag{5}$$

iii) the inner product of tangent vectors $A = RQ \in T_Q\mathcal{Q}$ and $B = SQ \in T_Q\mathcal{Q}$ with $R, S \in \mathcal{L}$, given by

$$\begin{aligned} \langle A, B\rangle &= \langle R, S\rangle = \mathrm{Tr}(R^H S) = \mathrm{Tr}(A^H B) & (6)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} Re(R_{i,j})Re(S_{i,j}) + Im(R_{i,j})Im(S_{i,j}) & (7)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} Re(A_{i,j})Re(B_{i,j}) + Im(A_{i,j})Im(B_{i,j}), & (8)\end{aligned}$$

defines a bi-invariant Riemannian metric on $\mathcal{Q}$ and corresponds to the inner product induced by the standard inner product in the embedding space $\mathbb{C}^{n\times n}$, identified with $\mathbb{R}^{2n^2}$ ($\mathrm{Tr}(\cdot)$ denotes the trace of a matrix) .
iv) Given the inner product (6) and the corresponding Riemannian connection, the geodesic, parameterized by the real parameter t, emanating from $Q = Q(0)$ in a direction SQ is given by

$$Q(t) = \exp(St)Q. \tag{9}$$

v) Parallel translation of a tangent vector $RQ = R(0)Q(0) \in T_Q\mathcal{Q}$ along the geodesic (9) is given by

$$R(t)Q(t) = \exp(\frac{1}{2}St)R(t)\exp(-\frac{1}{2}St)Q(t). \tag{10}$$

Proof: We refer the reader to [4] for the proof.

We are now in a position to derive a first algorithm to calculate a lower bound for μ_Δ.

3 The Steepest Ascent Algorithm

In this section we derive the steepest ascent algorithm (SAA) on the manifold $\mathcal{Q}$, solving the optimization problem as defined by (3). The steepest ascent algorithm (SAA) is an iterative algorithm, calculating in each step $k = 1, 2, \ldots$ a new point Q_k on the manifold $\mathcal{Q}$, given a point Q_{k-1}. To this end, a (one dimensional) optimization problem, often referred to as *line search*, is solved, maximizing $\rho(QM)$ along a geodesic on $\mathcal{Q}$, emanating from Q_{k-1} in the direction of the gradient. To simplify some of the calculations, $\Phi(Q) = (\rho(QM))^2$ will be used as an objective function, instead of $\rho(QM)$.
However, $\Phi(Q)$ is not differentiable at all points $Q \in \mathcal{Q}$. Therefore, theoretically, three cases are to be distinguished.

[1]For background material about Lie groups we refer to [6, 7].

case a: the spectral radius $\rho(QM)$ corresponds to one eigenvalue of QM with algebraic multiplicity 1.
case b: the spectral radius $\rho(QM)$ corresponds to $k > 1$ eigenvalues of QM with algebraic multiplicity 1.
case c: the spectral radius $\rho(QM)$ corresponds to an eigenvalue with algebraic multiplicity greater than 1.
Case a is generic. Case b is likely to occur at isolated points of some geodesics (when the absolute value of one eigenvalue crosses the absolute value of another eigenvalue). However, since $\rho(QM)$ is defined as $\max_i |\lambda_i|$ where λ_i are the eigenvalues of QM, and since these absolute values are smooth functions of Q in a neighborhood of points of case b, the maximum of $\rho(QM)$ cannot occur at a point of case b unless $\rho(QM)$ and $\Phi(QM)$ are differentiable at that point. Case c can be considered pathological and can in general be avoided by some heuristics (remark 4). The following theorem gives a formula for the gradient in differentiable points, and shows how the same formula gives an ascent direction (or 0 in a local maximum) for points of case b. The use of non steepest ascent directions, could theoretically prevent the ascent algorithm from converging to a local maximum, but only if the ascent direction would converge to a direction of no change. However, this scenario is excluded by the exceptional nature of both cases b and c.

Theorem 2 *i) Let the spectral radius of $X = QM$ correspond to one eigenvalue λ of X with algebraic multiplicity 1 (case a), and let p be a corresponding right eigenvector and v a corresponding left eigenvector v, satisfying $v^Hp = 1$. The gradient of $\Phi(Q) = \rho^2(QM)$, at the point $Q \in \mathcal{Q}$, with respect to the inner product (6) is given by*

$$\nabla_Q\Phi = \Phi(Q)GQ \text{ where } G = \pi(vp^H - pv^H), \tag{11}$$

where π denotes the orthogonal projection on $\boldsymbol{\Delta}$ and is given by

$$\pi(A) = diag[\alpha_1 I_{r_1}, \ldots, \alpha_s I_{r_s}, A_{f_1,f_1}, \ldots, A_{f_f,f_f}] \text{ for any } A \in \mathbb{C}^{n\times n},$$

where $\alpha_k = \frac{1}{r_k}\mathrm{Tr}(A_{r_k,r_k}), k = 1, \ldots, s$, and $A_{r_k,r_k}, k = 1, \ldots, s$ and $A_{f_l,f_l}, l = 1, \cdots, f$ are the submatrices of A with the same size and position as $\delta_k I_{r_k}$ and Δ_l of $\Delta \in \boldsymbol{\Delta}$ respectively. That is, π zeros elements that have to be zero in $\boldsymbol{\Delta}$, and averages elements that have to be equal in $\boldsymbol{\Delta}$.
ii) If the spectral radius $\rho(QM)$ corresponds to $m > 1$ eigenvalues of QM with algebraic multiplicity 1, then $\rho(QM)$ is the maximum of m differentiable functions, corresponding to the different eigenvalues. For any choice λ among these eigenvalues, and the corresponding right and left eigenvectors p and v, satisfying $v^Hp = 1$, $\Phi(Q)GQ$ is the gradient of one of these m functions, and gives an ascent direction of $\rho(QM)$ or gives 0 in a critical point. One of these choices gives the steepest ascent direction.

Proof: i) Clearly, $\Phi(Q)GQ$ belongs to the tangent space $T_Q\mathcal{Q}$, since $\Phi(Q)\pi(vp^H - pv^H)$ is skew Hermitian and belongs to $\boldsymbol{\Delta}$.
The only thing left to prove is that the directional derivative of Φ at Q in a direction SQ is given by $\langle\nabla_Q\Phi, SQ\rangle$. Or equivalently, if $Q(t)$ describes some path in $\mathcal{Q}$ through $Q = Q(0)$, with velocity $\dot{Q}(0) = SQ \in T_Q\mathcal{Q}$ (thinking of t as time), and $\Phi(t) = \Phi(Q(t))$ that

$$\dot{\Phi}(0) = \langle\nabla_Q\Phi, SQ\rangle \tag{12}$$

From $X = QM$ it is clear that $\dot{X} = \dot{Q}M = SQM = SX$. And since, by assumption the spectral radius of X corresponds to only one eigenvalue λ, $\lambda(t)$ is differentiable at $t = 0$ and $\dot{\lambda}(0) = v^H \dot{X} p = v^H SXp = \lambda v^H Sp$ [8, 2]. Finally, $\dot{\Phi}(0) = \frac{d}{dt}(\lambda^* \lambda) = \dot{\lambda}^* \lambda + \lambda^* \dot{\lambda} = \lambda^* \lambda v^H Sp + \lambda^* \lambda p^H S^H v = \Phi(Q)(v^H Sp - p^H Sv) = \langle S, \Phi(Q)(vp^H - pv^H) \rangle = \langle \pi(S), \Phi(Q)(vp^H - pv^H) \rangle = \langle S, \Phi(Q)\pi(vp^H - pv^H) \rangle = \langle SQ, \Phi(Q)\pi(vp^H - pv^H)Q \rangle$, which proves (12).
ii) In a neighborhood of Q, $\Phi(Q)$ can now be regarded as the maximum of m functions, that are equal at Q. The gradient of each of these functions separately is given by $\Phi(Q)GQ$, with one possible choice for λ, and the corresponding p and v. These gradients give steepest ascent directions (or 0 in critical points) for the individual functions, and therefore certainly give ascent directions for the maximum $\Phi(Q)$. One of the gradients corresponds to the steepest ascent direction. □

Remark 2 Note that this theorem could be easily extended to include repeated full blocks in the structure, analogous to repeated scalar blocks.

Application of the steepest ascent method to the present problem now yields the following algorithm.

Algorithm 1 (Steepest Ascent)

```
1.  initialize $Q_0$
2.  $k = 0$
3.  repeat
4.      compute $\rho = \rho(Q_k M)$
                and corresponding right and left eigenvectors $p$ and $v$,
                    such that $v^H p = 1$
5.      $G = \pi(vp^H - pv^H)$
6.      $t = \arg\max_{t>0} \rho(\exp(Gt)Q_k M)$
7.      $Q_{k+1} = \exp(Gt)Q_k$
8.      $k = k + 1$
9.  until convergence
```

Remark 3 Note that G in the algorithm and G in theorem 2 differ by a factor $\Phi(Q)$. Since only the direction of G is important for the algorithm this makes no difference. The exponentiation of $\exp(Gt) = \exp(\Phi(Q)\pi(vp^H - pv^H)t)$ can be calculated from the block components separately,

$$\exp(Gt) = \mathrm{diag}[\exp(G_{r_1,r_1}t), \ldots, \exp(G_{r_s,r_s}t), \exp(G_{f_1,f_1}t), \ldots, \exp(G_{f_f,f_f}t)]. \tag{13}$$

For the calculation of $\exp(G_\square t)$ for different t, where the index $\square$ denotes one of the block components in (13), the eigenvalue decomposition of $G_\square$ needs to be calculated only once. If the eigenvalue decomposition of $G_\square$ is given by $G_\square = EiDE^H$, where $E^H E = I$ and D is real diagonal, $\exp(G_\square t)$ is found as:

$$\exp(G_\square t) = E \exp(iDt) E^H.$$

Note that for repeated scalar blocks one finds $E = I$. For full blocks, more efficient formulas can be obtained also, by exploiting the fact that the block components of G have rank 2 (if not 0), but the computational benefit is rather small compared to the cost of the repeated computation of ρ, p and v.

Remark 4 For points of case c, calculation of an ascent direction is much more difficult than for other points. In addition the effort is not worthwhile, as this case does not normally occur and if it occurs, it can generally be avoided. In most cases, when a point of case c is met during the line search, one can simply take another point. In many cases points of case c can be avoided by multiplying Q with $U \in \mathcal{Q}$, for which $Up = p$ and $U^H v = v$. In that case $\Phi(UQ) = \Phi(Q)$ but in general UQ will no longer be of case c. If none of this works the algorithm is stopped (and possibly restarted from another initial value). After all, the point of case c, can be a local optimum in contrived examples.

The computationally most expensive steps of the algorithm are the evaluations of the spectral radius of X and the corresponding right and left eigenvector. These do not only occur in line 4 of the algorithm but mainly in line 6, during the line search. (Actually, except for the first step in the iteration, nothing needs to be calculated in line 4, since ρ, p and v were calculated already during the last execution of line 6). In order to minimize this cost, two things can be done.

1. The number of eigenvalue/eigenvector calculations is kept low, using an efficient line search algorithm by Fletcher [9, pp.33]. This algorithm finds a point, satisfying the (modified) Wolfe-Powell conditions (normally given for minima instead of maxima)

$$\begin{array}{rcl} \Phi(t) & \geq & \Phi(0) + t\,\rho_F \frac{d\Phi}{dt}(0) \\ \|\frac{d\Phi}{dt}(t)\| & \leq & \sigma_F \frac{d\Phi}{dt}(0) \end{array} \qquad (14)$$

where ρ_F and $\sigma_F \geq \rho_F$ are fixed parameters. Typically $\sigma_F = 0.5$ and $\rho_F = 0.01$, for a moderately accurate line search [9, p.30]. Fletcher's algorithms are designed for smooth objective functions, but can be easily adapted for the present case, because no optima occur at non differentiable points.

2. For the calculation of ρ, p and v, some profit can be taken from the fact that only the principal eigenvalue and the corresponding eigenvectors are needed, and the fact that previous calculations for nearby X provide good initial estimations. Both the well known power algorithm (not to be confused with the PIA for the structured singular value, called power algorithm for its resemblance with the classical power algorithm) and the inverse iteration algorithm can save a lot of computations. However, the power algorithm is not efficient if the maximal eigenvalue of X is not well separated (in absolute value) from the other eigenvalues, and the inverse iteration algorithm can converge to the wrong eigenvector with a bad initial guess for the eigenvalue. Therefore, the prize of the computational improvement, is an amount of heuristic rules to handle the distinction between different cases, or the loss of theoretically guaranteed convergence to a local optimum. However, our experience shows that problems are rare and efficiency can be increased, when the line search is made more robust against inaccurate gradient information. In most numerical experiments however we have used the MATLAB function "eig" [10], to calculate a full eigenvalue decomposition. Also in that case the efficiency of the algorithm is fully acceptable.

4 The Conjugate Gradient Algorithm

In this section we derive the conjugate gradient algorithm (SAA) on the manifold $\mathcal{Q}$, solving the optimization problem as defined by (3). For the conjugate gradient algorithm (CGA) the search direction at a point Q_k is determined in most steps by the gradient at Q_k and the previous search direction at Q_{k-1}. In most versions, the gradient direction itself is taken every n-th step (where n is the problem dimension). Smith proposes the following formula for the search direction H_kQ, at a point Q_k using the gradients G_kQ and $G_{k-1}Q$ and the previous search direction $H_{k-1}Q$. It is very similar to the classical formula for the CGA in Euclidean spaces, but tangent vectors at the point Q_{k-1} from the previous iteration step, are adapted to Q_k by parallel translation along the geodesic through Q_{k-1} and Q_k.

$$H_k = G_k + \gamma_k \tau H_{k-1}$$

where

$$\gamma_k = \frac{\langle G_k - \tau G_{k-1}, G_k\rangle}{\langle G_{k-1}, H_{k-1}\rangle}$$

where τ denotes parallel translation from Q_{k-1} to Q_k, applied to matrices $S \in \mathcal{L}$ representing a tangent vector SQ at Q. It follows from theorem 1 that $\tau H_{k-1} = H_{k-1}$. To calculate τG_{k-1} formula (10) is needed. This leads to the following algorithm.

Algorithm 2 (Conjugate Gradient)

```
1.   initialize $Q_0$
2.   $k = 0$
3.   repeat
4.       compute $\rho = \rho(Q_k M)$
                 and corresponding right and left eigenvectors $p$ and $v$,
                    such that $v^H p = 1$
5.       $G_k = \Phi(Q)\pi(vp^H - pv^H)$
6.       if $(k \bmod n) = 0$ then $H_k = G_k$
7.       else $\tau G_{k-1} = \exp(\frac{1}{2}H_{k-1}t)G_{k-1}\exp(-\frac{1}{2}H_{k-1}t)$
8.           $\gamma = \frac{\langle G_k - \tau G_{k-1}, G_k\rangle}{\langle G_{k-1}, H_{k-1}\rangle}$
9.           $H_k = G_k + \gamma H_{k-1}$
10.      $t = \arg\max_{t>0} \rho(\exp(H_k t)Q_k M)$
11.      $Q_{k+1} = \exp(H_k t)Q_k$
12.      $k = k + 1$
13.  until convergence
```

Remark 5 For the calculation of $\exp(Ht)$ for different values of t, the same remark holds as for the SAA (remark 3).

Remark 6 For the line search of line 10, the same strategy as for the SAA is used. However, since the performance of the CGA is more sensitive to the quality of this line search, the conditions (14) are made stronger by taking $\sigma_F = 0.1$ ($\rho_F = 0.01$) [9, p.85].

The CGA is known to have better convergence properties than the SAA. The additional cost of computing the conjugate gradient search direction is small compared to the cost of the function evaluations and gradient calculations, especially when the structured matrices consist of small blocks, since these computations can be performed on the block components separately. However, due to the stronger requirements for the line search (remark 6), individual iterations may require more computations. In section 5 we show that for the present problem, both algorithms are more or less equivalent. CGA can be preferred if accuracy is important.

5 Comparison with the power iteration algorithm and numerical experiments

In this section we support our theoretical results with numerical experiments, and give an idea of the efficiency of our algorithms and compare them with the PIA as implemented in the MATLAB μ toolbox [11]. The MATLAB code, used for these experiments, is available via anonymous ftp from gate@esat.kuleuven.ac.be in /pub/SISTA/dehaene/prog/ccmu.
The derivation of the power iteration algorithm [2] starts from conditions for a locally optimal value of μ_Δ, to obtain a computationally cheap iteration scheme, which, if it converges, finds a local optimum. Convergence however is not guaranteed. Our algorithms have been designed in the first place as convergent algorithms, but require more computations. We propose to use both algorithms together, for example first try the power iteration algorithm, and if it doesn't converge within a preset time limit, use the SAA or CGA.
In experiments below we have always used the same convergence test. The algorithm stops if the norm of the gradient drops below 1E-3, or if the relative increase of Φ drops below 1E-3 for 2 consecutive steps.
One example of a simple problem for which the PIA does not converge is

$$M = \begin{bmatrix} 1.0308 & 0.7611 & -0.3225 \\ -0.7599 & -0.1659 & -0.3684 \\ 0.8741 & 0.3009 & 1.1479 \end{bmatrix}$$

with eigenvalues $0.8678 \pm 0.7474i$ (corresponding to spectral radius $\rho(M) = 1.1453$) and 0.2773, and the block structure of $s = 1$, $f = 1$, $r_1 = 2$ and $m_1 = 1$ (one repeated scalar block of dimension 2×2 and one full block with dimension 1×1. The PIA gives a lower bound of 1.2745, while inspection inside the MATLAB implementation of the algorithm reveals non-convergent behavior. Fig. 1 shows the evolution of the variable β which should converge to a locally optimal μ_Δ. The CGA and the SDA, with MATLAB complete eigenvalue/eigenvector calculation or with power iterations or inverse iterations (see section 3), all give answers in the range [1.3840 1.3846]. Real matrices with complex eigenvalues, often give rise to a similar situation. It should be emphasized that these examples can be relevant for practical applications.
Fig. 2 compares the computational cost of SAA, the CGA and PIA, for complex matrices with different sizes and a similar block structure, consisting of two repeated scalar blocks of size z and two full blocks of size z, where $z = 1, \ldots, 10$. The matrices were generated at random and for each size the computational cost was averaged over 10 matrices. Experiments with different block structures give no significant difference. Both the SAA and the CGA are less efficient than the PIA. However the computational efficiency of the SAA or the CGA is still acceptable. A considerable improvement is already realized by calculating the eigenvectors and eigenvalues with power iterations or inverse iterations (see section 3). For both power iterations and inverse iterations the same convergence test was used, stopping the iterations if the change in one iteration of the eigenvector being calculated drops below 1E-3. For power iterations a (seldomly reached) upper limit of 1000 iterations was set. For inverse iterations this limit was set to 100 iterations. If the limit is reached the eigenvectors and eigenvalues are calculated with MATLAB function "eig". Fig. 3 shows the result for matrices of different sizes, generated in the same way as for the comparison of the SAA and the CGA (figure 2). Note that power iterations and inverse iterations require approximately the same number of operations.

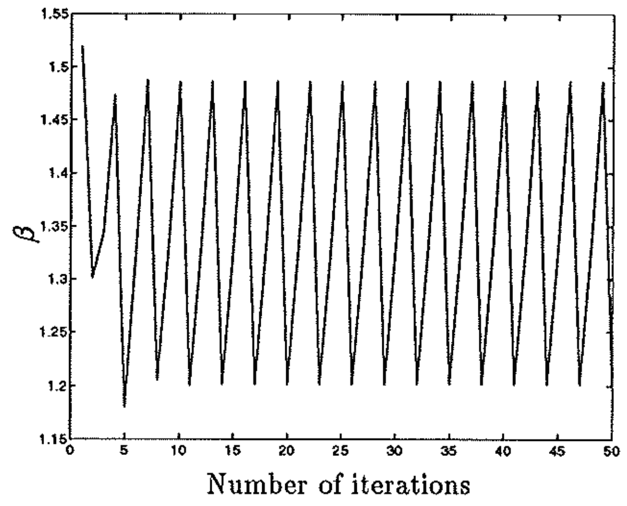

Figure 1: The evolution of the estimate β of the structured singular value in PIA algorithm shows non-convergent behavior. β eventually evolves in a limit cycle

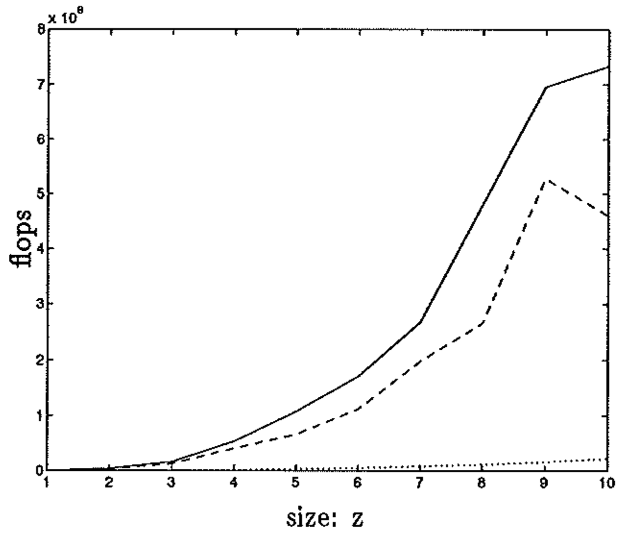

Figure 2: Number of floating point operations for CGA (full line —), SAA (dashed line – – –) and PIA (dotted line $\cdots$), for matrices with different sizes and a block structure, consisting of two repeated scalar blocks of size z and two full blocks of size z, where $z = 1, \ldots, 10$.

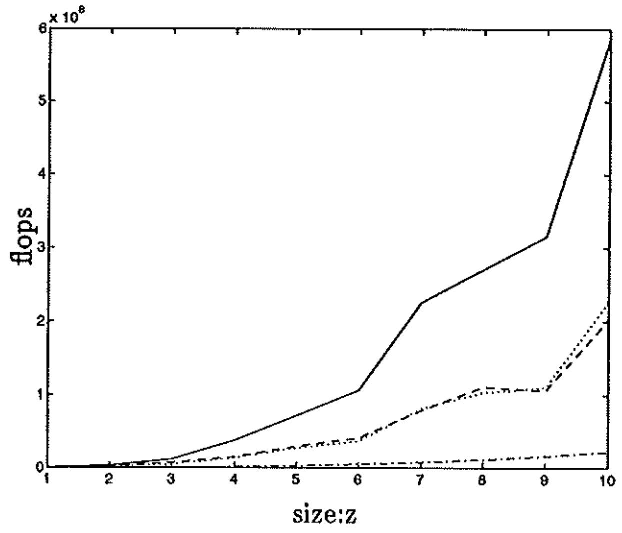

Figure 3: Number of floating point operations for SAA, with full eigenvector/eigenvalue computation with the MATLAB function "eig" (full line —), with power iterations (dashed line – – –), with inverse power iterations (dotted line · · ·) and for PIA (dash-dot line · – · –).

6 Conclusion

Using the theory of gradient based optimization on Riemannian manifolds, we have derived steepest ascent and conjugate gradient algorithms for the structured singular value problem, with acceptable efficiency and with guaranteed convergence to a locally optimal lower bound for the structured singular value.

References

[1] J.C.Doyle, "Analysis of feedback systems with structured uncertainties", IEE Proceedings, Vol. 129, pp. 242-250, 1982.

[2] A. Packard, M.K.H. Fan J. Doyle, "A power method for the structured singular value decomposition", Proceedings of the 27th Conference on Decision an Control, Austin, Texas, December 1988, pp. 2132-2137.

[3] YI Cheng and Bart De Moor, "Continuous time power iteration for computing lower bound of μ", Proceedings of the 1994 American Control Conference, Baltimore, Maryland, June 29 - July 1, 1994, pp. 1416-1417.

[4] Jeroen Dehaene, Cheng Yi and Bart De Moor, "Calculation of the structured singular value with gradient based optimization algorithms on a Lie group of structured unitary matrices", Internal report (ESAT-SISTA/TR 1994-46I), Departement Electrotechniek, Katholieke Universiteit Leuven, 1994, submitted to IEEE Trans. AC.

[5] S.T. Smith, "Geometric Optimization Methods for Adaptive Filtering", PhD. thesis, Harvard University, Cambridge, Massachusetts, May 1993.

[6] D.H. Sattinger and O.L. Weaver, "Lie Groups and Algebras with Applications to Physics, Geometry and Mechanics", Applied Mathematical Sciences, Vol.61, Springer-Verlag, New York, 1993.

[7] S. Helgason, "Differential Geometry, Lie Groups and Symmetric Spaces." Academic Press, New York, 1978.

[8] Kato, T. "A short introduction to Perturbation Theory for Linear Operators" Springer-Verlag, 1982.

[9] Fletcher R., "Practical Methods of Optimization", Wiley & Sons, Chichester, 2nd ed. (repr.), 1993.

[10] The MathWorks Inc., "MATLAB Reference Guide", Natick, Massachusetts, 1992.

[11] G.J. Balas, J.C. Doyle, K. Glover, A. Packard and R. Smith, "μ-Analysis and Synthesis Toolbox", The Math Works Inc., 1993.