SUBSPACE SEPARATION BY DISCRETIZATIONS OF DOUBLE BRACKET FLOWS

K. HÜPER, J. GÖTZE, and S. PAUL
Technical University of Munich
Institute of Network Theory and Circuit Design
D-80290 Munich
Germany
knhu@nws.e-technik.tu-muenchen.de

ABSTRACT. A method for the separation of the signal and noise subspaces of a given data matrix is presented. The algorithm is derived by a problem adapted discretization process of an equivalent dynamical system. The dynamical system belongs to the class of isospectral matrix flow equations. A matrix valued differential equation, whose time evolution converges for $t \to \infty$ to block diagonal form is considered, i.e., only the cross-terms, correlating signal and noise subspaces, are removed. The iterative scheme is performed by computing some highly regular orthogonal matrix-vector multiplications. The algorithm essentially works like a Jacobi-type method. An updating scheme is also discussed.

KEYWORDS. Double Bracket Flow, Gradient Flow, Block Diagonalization, Principal Eigenspace, Jacobi-type Methods.

Introduction

Given a data matrix $H \in \mathbb{R}^{n \times m}$ $(n > m)$, a frequently encountered problem in signal processing (DOA–estimation, harmonic retrieval, system identification) is the separation of the column space of H into signal and noise subspaces. The SVD of H is the most robust and reliable tool for this task. The signal subspace is defined by the right singular vectors corresponding to the large singular values. Computing the SVD of H, however, is computationally expensive and yields much more information than necessary to separate the signal and the noise subspaces. In order to avoid the computationally expensive SVD, various methods for determining the subspaces in a computationally less expensive way have been published. Among these methods are the Rank–Revealing QRD [2], the URV decomposition [11] and the SVD–updating algorithms [8, 9] which can be considered as

approximate SVD's. Other methods not based on the SVD have also been proposed, e.g. the Schur–type method presented in [4]. Another quite obvious approach is the block diagonalization of H, i.e.

$$H = U \begin{bmatrix} H_{11_\infty} & 0 \\ 0 & H_{22_\infty} \\ 0 & 0 \end{bmatrix} V^T ,$$

where $U \in O(n)$, $V \in O(m)$, and $H_{11_\infty} \in \mathbb{R}^{d\times d}$, $H_{22_\infty} \in \mathbb{R}^{m-d\times m-d}$ have dimensions due to the dimensions of signal and noise subspaces, respectively. It is not straight forward, however, to extend known linear algebra algorithms such, that *any* block diagonalization of an arbitrary matrix is obtained. In this paper an algorithm for block diagonalizing a given matrix by orthogonal transformations is derived. This is achieved by a problem adapted discretization process of an equivalent dynamical system. The matrix flow is Brockett's double bracket flow [1] $\dot{H} = [H,[H,N]]$, choosing the matrix N appropriately with respect to the subspace separation problem. Essentially, after discretization, this results in a Jacobi–type method which only works on the cross terms $H_{12} \in \mathbb{R}^{d\times m-d}$ (assuming a preparatory QRD). Therefore, only $d(m-d)$ rotations are required per sweep, while the standard Jacobi-type methods apply $m(m-1)/2$ rotations. The algorithm can be considered as a method for maximizing an objective function of a continuous-time gradient flow. It is a specific example of a gradient ascent method using geodesic interpolation (see [5], [7], and [10] for related work). An updating scheme for this algorithm is very similar to the SVD–updating scheme of [8]. Again, the updating scheme only operates on the cross terms.

1 Dynamical System for Block Diagonalization

In this section a dynamical system for block diagonalizing symmetric matrices with distinct eigenvalues is briefly presented. We shall state the necessary propositions without proofs. For convenience, we have decided to present the results for symmetric matrices (covariance matrices). The extension to arbitrary $n \times m$ data matrices (SVD) is straightforward.

The isospectral matrix flow is governed by

$$\dot{H} = [H,[H,N]], \tag{1}$$

and the associated flow on orthogonal matrices by

$$\dot{\Theta} = \Theta[H,N], \tag{2}$$

with $H = H^T \in \mathbb{R}^{m\times m}$, $\Theta(t) \in O(m)$, $[X,Y] \stackrel{def}{=} XY - YX$, and $N = diag(I_d, 0_{m-d})$.

Proposition 1.1 *Eq.(1) is the gradient flow $\dot{H} = grad\, f(H)$ of the function $f(H) = -tr\, H_{11}$. Eq.(2) is the gradient flow $\dot{\Theta} = grad\, \phi(\Theta)$ of the function $\phi(\Theta) = -tr\, N\Theta^T H_0 \Theta + tr\, H_0 = f(H)$.*

Remark The gradient flows in Proposition 1.1 are defined with respect to different metrics. See [5] for details.

Proposition 1.2 *For $t \to \infty$ the time dependent $H(t)$ of (1) with initial $H(0)$ converges*

to block diagonal form:

$$H(0) = \begin{bmatrix} H_{11}(0) & H_{12}(0) \\ H_{12}^T(0) & H_{22}(0) \end{bmatrix} \stackrel{t\to\infty}{\longrightarrow} \begin{bmatrix} H_{11_\infty} & 0 \\ 0 & H_{22_\infty} \end{bmatrix}$$

with $H_{11} \in \mathbb{R}^{d\times d}$, $H_{12} \in \mathbb{R}^{d\times m-d}$, *and* $H_{22} \in \mathbb{R}^{m-d\times m-d}$. *Assuming* $\sigma(H_{11_\infty}) \cap \sigma(H_{22_\infty}) = \{\}$, *for all* $\lambda_i \in \sigma(H_{11_\infty})$ *and for all* $\lambda_j \in \sigma(H_{22_\infty})$ *holds* $\lambda_i > \lambda_j$. *For* $t \to \infty$ *the time dependent* $\Theta(t)$ *of (2) with initial* $\Theta(0) \in O(m)$ *converges to* $(\Theta_{1_\infty}, \Theta_{2_\infty})$ *with* $\Theta_{1_\infty}^T\Theta_{1_\infty} = I_d$ *and* $\Theta_{2_\infty}^T\Theta_{2_\infty} = I_{m-d}$. *It holds*

$$span(\Theta_{1_\infty} | H\Theta_{1_\infty} = \Theta_{1_\infty} H_{11_\infty}) = span\{q_1, \ldots, q_d | Hq_i = q_i\lambda_i, \lambda_i \in \sigma(H_{11_\infty})\},$$

$$span(\Theta_{2_\infty} | H\Theta_{2_\infty} = \Theta_{2_\infty} H_{22_\infty}) = span\{q_{d+1}, \ldots, q_m | Hq_i = q_i\lambda_i, \lambda_i \in \sigma(H_{22_\infty})\}.$$

Proof: see e.g. [7].

2 Discretization Scheme for Block Diagonalization

A discretization scheme for the above gradient flow is now presented. Taking into account that at each time step only an orthogonal transformation in a specific 2−dimensional plane of the matrix H is performed, a modification of the cyclic-by-row Jacobi method results. This method works with the full group $SO(2)$ or equivalently, the rotation angles $\phi \in [-\pi/2, \pi/2]$. Excluding certain pathological initial conditions (saddle points) which in general do not occur for subspace separation problems, this algorithm is obviously globally convergent. Furthermore local quadratic convergence for the case of full diagonalization (including multiple eigenvalues) was recently proved using methods from global analysis [6] (see also the contribution by U. Helmke to this workshop). Here we borrowed from [3] the very readable MATLAB-like algorithmic language and notation.

Algorithm 2.1 *Given a symmetric* $H \in \mathbb{R}^{m\times m}$ *and* $p, q \in \mathbb{N}$ *that satisfy* $1 \le p < d$ *and* $d + 1 \le q \le m$, *this algorithm computes a cosine-sine pair* (c, s) *such that if* $H' = J(p,q)^T H J(p,q)$ *then* $h'_{pq} = 0$ *and* $h'_{pp} \ge h'_{qq}$.

function: $(j_{pp}, j_{pq}, j_{qp}, j_{qq}) =$ ***sym.schur2.sorted*** (H, p, q)

if $h_{pq} \neq 0$

$$\tau = \frac{h_{qq} - h_{pp}}{2h_{pq}};\ t = \frac{sign\,\tau}{|\tau| + \sqrt{1+\tau^2}};\ c = \frac{1}{\sqrt{1+t^2}};\ s = t\,c$$

else

$c = 1;\ s = 0$

end

if $h_{pp} \le h_{qq}$

$j_{pp} = -s;\ j_{pq} = c;\ j_{qp} = c;\ j_{qq} = s$

else

$j_{pp} = c;\ j_{pq} = s;\ j_{qp} = -s;\ j_{qq} = c$

end

end sym.schur2.sorted

Algorithm 2.2 ***(Block Diagonalizing Cyclic Jacobi)*** *Given a symmetric* $H \in \mathbb{R}^{m\times m}$ *and a tolerance* $\epsilon \ge 0$, *this algorithm overwrites* H *with block diagonal* $H' = \Theta^T H \Theta$ *where*

$\Theta \in O(m)$ *and* $\Delta f = tr(H'_{11} - H_{11}) \leq \epsilon$. *It returns* $\Theta = (\Theta_1, \Theta_2)$ *with* $\Theta_1(\Theta_2)$ *being an orthogonal basis for the principal* $d-$*dimensional (minor* $(m-d)-$ *dimensional) eigenspace of* H, *respectively.*

function: $(\Theta, H) =$ ***block.diag.jacobi*** (Θ, H, ϵ)
  $\Theta = I_m$
  $trOld = 0$
  $trNew = trace(H(1:d, 1:d))$
  while $(trNew - trOld) > \epsilon$
    for $p = 1:d$
      for $q = d+1:m$
        $(j_{pp}, j_{pq}, j_{qp}, j_{qq}) = sym.schur2.sorted(H, p, q)$
        $H = J(p,q)^T H J(p,q)$; $\Theta = \Theta J(p,q)$
      end
    end
    $trOld = trNew$; $trNew = trace(H(1:d, 1:d))$
  end
end ***block.diag.jacobi***

Obviously, Algorithm 2.2 is essentially a Jacobi-type method. The case of fully diagonalizing the matrix H corresponds to choosing the matrix N with distinct eigenvalues. Modifying Algorithm 2.2 such, that each sweep works on the whole off-diagonal part, the standard cyclic-by-row Jacobi method results, except that rotations sorting the diagonal entries are applied via Algorithm 2.1. Note, that instead of the off–diagonal norm the function trace(NH) is used as the optimization criterion.

3 Updating Procedure

In a time varying environment, the most common strategy for updating the covariance matrix H is the rank one update, which involves applying a forgetting factor μ to H due to nonstationary processes. Suppose that at a certain time step, the matrix H is reduced to block diagonal form via Algorithm 2.2, with associated invariant subspaces $span(\Theta_1)$ and $span(\Theta_2)$. Our update procedure is as follows (number of sweeps is predetermined):

Algorithm 3.1 ***(Update)*** *Given a symmetric* $H = diag(H_{11}, H_{22})$ *and associated* $\Theta = (\Theta_1, \Theta_2)$, *a new data vector* x *and a possibly time-varying forgetting factor* μ, *with* $H_{11} \in \mathbb{R}^{d \times d}$, $H_{22} \in \mathbb{R}^{m-d \times m-d}$, $\Theta_1^T \Theta_1 = I_d$, $\Theta_2^T \Theta_2 = I_{m-d}$, *and* $x \in \mathbb{R}^{m \times 1}$.

$H = \mu H + \Theta^T x x^T \Theta$
$(\Theta, H) = block.diag.jacobi(\Theta, H, \epsilon)$

4 Simulations and Discussion

The performance of the different Jacobi methods is compared for the subspace separation problem using a frequency estimation scenario. $n = 100$ sample points of a signal composed

of 2 sinusoids is used, i.e.,

$$s(t) = 2\cos(2\pi an) + 2\cos(2\pi bn) + w(n)\,,$$

where $a = 0.15$, $b = 0.25$ and $w(n)$ is Gaussian white noise. Data vectors of dimension $m = 8$ are formed generating the $n \times m$ Toeplitz data matrix X.

Computing the signal (noise) subspace of X requires the separation of the row space of the data matrix X (the row space of the covariance matrix $A = X^T X$, respectively). Once these subspaces are computed the unknown parameters can be estimated using methods like ESPRIT or MUSIC. Usually, the subspaces can be obtained by computing the SVD of the data matrix X or the EVD of the covariance matrix A. Assuming a certain signal to noise ratio SNR the row space of $X(A)$ can be separated into signal and noise subspace according to the magnitude of the singular(eigen)values, respectively. The right singular(eigen)vectors corresponding to the large singular(eigen)values form the signal subspace. In our example the dimension of the signal subspace is $d = 4$ (two times the number of signals in case of working with real data).

In the following simulations we distinguish between three different Jacobi–type methods:

- SCJ (standard cyclic Jacobi): a diagonalization of A is executed applying rotations ($\theta \leq \pi/4$) which do not take into consideration the magnitude of the resulting diagonal matrix elements.
- SORTCJ (sorting cyclic Jacobi): a diagonalization of A is executed applying rotations ($\theta \leq \pi/2$), which sort the resulting diagonal elements according to their magnitude.
- SORTCJ–CT (SORTCJ working only on the cross terms): it works like SORTCJ but only the cross terms are annihilated in each sweep, i.e., one sweep consists of $d(m-d)$ rotations. Note that for working only with the cross terms the sorted version of the Jacobi method is mandatory in order to guarantee convergence.

Also, two different off–diagonal quantities are considered in the following:

- $sd = \|A - D\|_F$, where $D = \mathrm{diag}(a_{ii})$, i.e., the square root of the sum of the squares of *all* off–diagonal elements (the usual off–diagonal norm).
- $sc = \|A(1:d, d+1:m)\|_F$, i.e., the square root of the sum of the squares of the cross terms, only.

First, SCJ and SORTCJ are compared – both methods actually diagonalizing the matrix. In figure 1 the reduction of the off–diagonal quantities sd and sc is shown for SCJ and SORTCJ. Obviously, the diagonalization of the matrix (i.e. considering sd) works as well for SCJ as for SORTCJ. Reduction of the cross terms (i.e. sc), however, is significantly faster for SORTCJ than for SCJ. Therefore, SORTCJ performs favorable with respect to the subspace separation task.

In figure 2 the reduction of the cross terms sc is shown for SORTCJ and SORTCJ–CT. Obviously, in this case convergence of SORTCJ–CT is only linear, such that a greater number of sweeps is required. Note also, that sc even increases in the beginning of SORTCJ–CT.

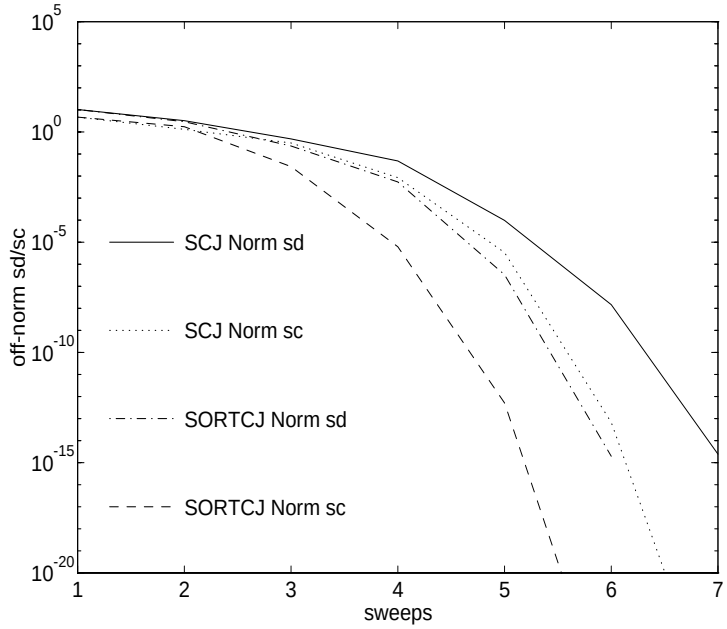

Figure 1: Reduction of the off–diagonal norms *sd* and *sc* vs. sweeps for SCJ and SORTCJ.

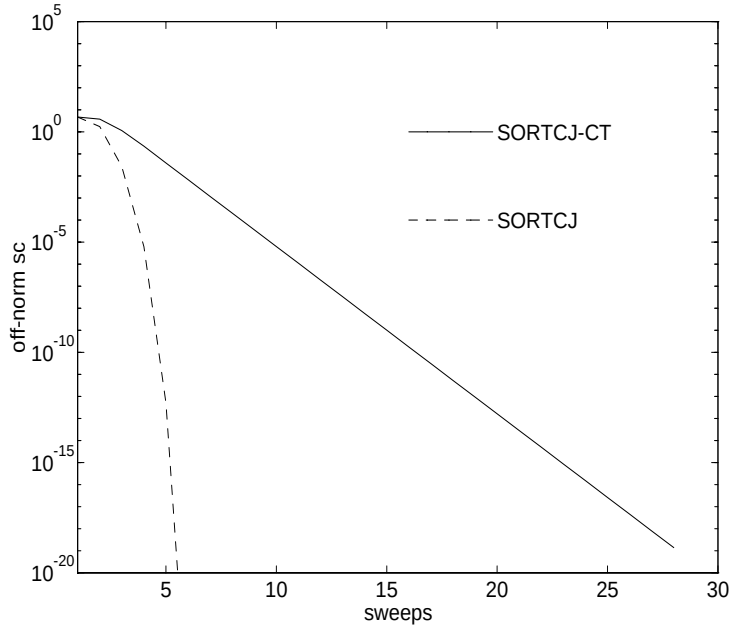

Figure 2: Reduction of the cross terms, i.e. *sc*, vs. sweeps for SORTCJ and SORTCJ–CT.

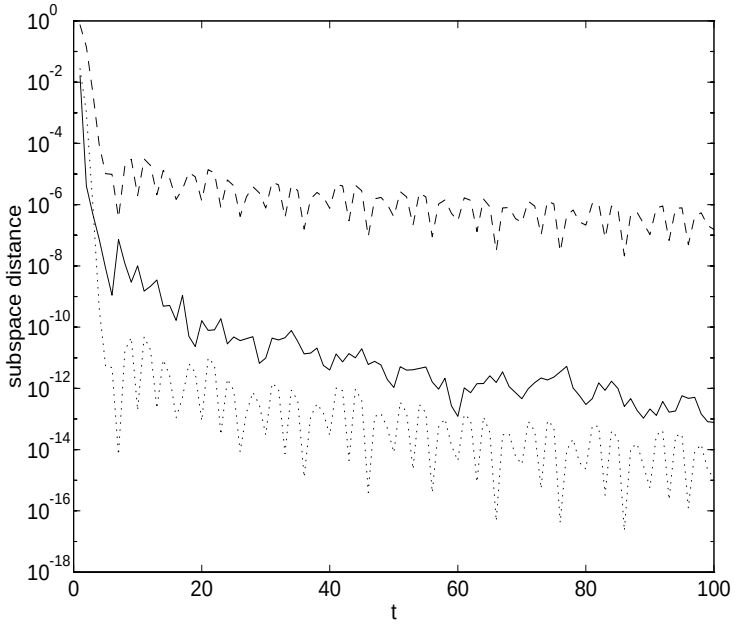

Figure 3: Distances between updated subspaces and exact instantaneous subspaces vs. time (solid line: SORTCJ $s = 1$; dashed line: SORTCJ–CT $s = 1$; dotted line: SORTCJ–CT $s = 3$).

However, this does not contradict to the convergence of the algorithm, since the optimization criteria used for the derivation of SORTCJ–CT (SORTCJ) is *not* the off–diagonal norm but the trace of the matrix H weighted by the matrix N! Now, let f_s denote the factor of the increase in the number of sweeps, i.e., f_s = sweeps(SORTCJ–CT)/sweeps(SORTCJ) (in our example $f_s \approx 5$). SORTCJ–CT only requires $N_C = d(m-d)$ rotations per sweep while SORTCJ requires $N_S = m(m-1)/2$. Therefore, SORTCJ–CT is advantageous for $N_C f_s < N_S$, i.e., for $d \ll m$.

Finally the tracking capabilities of the different algorithms are considered. Figure 3 shows the ensemble average (20 runs) of the distance between the signal subspaces $\hat{V}(t)$ computed by the updating procedure and the instantaneous exact subspaces $V(t)$. The distance between these subspaces is measured by $dist(\hat{V}(t), V(t)) = (1 - \sigma^2_{min}(\hat{V}^T(t)V(t))^{1/2}$ [3]. The updating by SORTCJ uses $s = 1$ sweep per updating (solid line). The updating using SORTCJ-CT is shown for $s = 1$ (dashed line) and $s = 3$ (dotted line) per update.

5 Conclusion and Outlook

In this paper a Jacobi-like method is presented for block diagonalizing symmetric matrices with application to subspace separation problems. A generalization to SVD is straight forward. The algorithm is also well suited to neural network applications, where block diagonalization into more than two blocks is required.

Acknowledgments

The authors are grateful to W. Mathis for drawing their attention to the topic of isospectral matrix flows. They would also like to thank U. Helmke who spent considerable efforts for introducing them to the optimizational point of view and R.E. Mahony for making his thesis available. The authors are indebted to J.A. Nossek for his continuous support and many helpful discussions.

References

[1] R. W. Brockett. Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Lin. Algebra & Applic.*, 146:79–91, 1991.

[2] T.F. Chan. Rank revealing QR-factorization. *Lin. Algebra & Applic.*, 89:67–82, 1987.

[3] G. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 2nd edition, 1989.

[4] J. Götze and A.-J. van der Veen. On–line subspace estimation using a Schur–type method. *IEEE Transactions on Signal Processing*, 1993. submitted.

[5] U. Helmke and J.B. Moore. *Optimization and Dynamical Systems*. CCES. Springer, London, 1994.

[6] K. Hüper and U. Helmke. On the convergence and structure of Jacobi-type methods, 1994. preprint.

[7] R.E. Mahony. *Optimization algorithms on homogeneous spaces*. PhD thesis, Australian National University, 1994.

[8] M. Moonen, P. van Dooren, and J. Vandewalle. A singular value decomposition updating algorithm for subspace tracking. *SIAM J. Matrix Anal. Appl.*, 13:1015–1038, 1992.

[9] M. Moonen, P. van Dooren, and F. Vanpoucke. On the QR algorithm and updating the SVD and the URV decomposition in parallel. *Lin. Algebra & Applic.*, 188:549–568, 1993.

[10] S. T. Smith. *Geometric optimization methods for adaptive filtering*. PhD thesis, Harvard University, Cambridge, May 1993.

[11] G.W. Stewart. An updating algorithm for subspace tracking. *IEEE Transactions on Signal Processing*, 40:1535–1541, 1992.