A NEW APPROACH TO JOINT DIAGONALIZATION

Gen Hori

Laboratory for Open Information Systems,
Brain Science Institute, RIKEN, Saitama 351-0198, Japan
`mailto:hori@open.brain.riken.go.jp`

ABSTRACT

Continuous gradient algorithms on matrix Lie groups for joint diagonalization are introduced. Based on the continuous gradient algorithms, discrete algorithms are derived. One of the derived discrete algorithms is faster than the extended Jacobi method in the early phase of calculation on the Matlab `tic-toc` measurement.

1. INTRODUCTION

Joint diagonalization of several matrices is a problem of finding a similar transformation which transform the matrices as diagonal as possible simultaneously. From the purely mathematical viewpoint, if all the given matrices commute each other they can be diagonalized simultaneously with a similar transformation. And from the viewpoint of numerical methods for read data, even if they don't commute each other we can find a similar transformation which transform them as diagonal as possible by optimizing some joint diagonality criterion. We call the former "exact joint diagonalization" and the latter "approximate joint diagonalization".

Joint diagonalization algorithm is a numerical engine for some ICA methods. JADE algorithm developed by Cardoso and Souloumiac[2] is based on the joint diagonalization of cumulant matrices. Delayed correlation method introduced by Molgedey and Schuster[4] turns the signal separation to the joint diagonalization of time delayed correlation matrices.

Bunse-Gerstner, Byers and Mehrmann[1] studied numerical method for joint diagonalization emphasizing stability and convergence problems. The numerical method studied by them is some extension of the traditional Jacobi method for matrix eigenvalue problem and iterates Jacobi rotations (or plane rotations) to optimize some joint diagonality criterion defined. Cardoso and Souloumiac[3] gave a closed expression of the optimal Jacobi angle for the extended Jacobi method.

This article proposes a new approach to joint diagonalization algorithm using gradient flows on matrix Lie groups. Section 2 gives the joint diagonality criterion which is the potential function of the continuous gradient algorithms. Section 3 introduces the generic framework for making continuous gradient algorithms for joint diagonalization and derives some practical examples of gradient algorithms. Section 4 derives discrete algorithms from continuous algorithms introduced in Section 3. Section 5 tests the performance of our new method through some numerical experiments. The final section contains some remarks on our new numerical methods for joint diagonalization.

2. JOINT DIAGONALITY CRITERION

Consider a set of K matrices

$$\mathfrak{A} = \{A_1, A_2, \ldots, A_K\}, \quad A_k \in M(n, C)$$

and a matrix Lie group $\mathfrak{G}$ of $n \times n$ matrices. Joint diagonalization of the set of matrices $\mathfrak{A}$ is a problem of finding $G \in \mathfrak{G}$ which makes

$$G^{-1}A_1G, G^{-1}A_2G, \ldots, G^{-1}A_KG$$

as diagonal as possible simultaneously, where $\mathfrak{G}$ is typically $SO(n)$ or $U(n)$.

To measure a diagonality of a matrix, Bunse-Gerstner et al.[1] defined

$$\mathbf{off}(A) = \sum_{1 \le i \ne j \le n} |a_{ij}|^2$$

where a_{ij} denotes the (i, j)-th entry of matrix A, and introduced the following joint diagonality criterion

$$\sum_{k=1}^{K} \mathbf{off}(U^* A_k U) \tag{1}$$

for the set of matrices $\mathfrak{A}$ where U is a unitary matrix. In this context, joint diagonalization is a problem of finding a unitary matrix U which minimize (1), and its numerical solution may be obtained by minimizing

(1) with some iterative modification to U. The extended Jacobi methods for joint diagonalization studied by [1][3] construct U as a product of matrices which express Jacobi rotations (or plane rotations).

In this paper, we consider more general formulation of joint diagonalization problem with a general joint diagonality criterion of the form

$$\psi(G) = \sum_{k=1}^{K} \psi_k(G^{-1} A_k G) \qquad (2)$$

where $\psi_k : M(n,C) \to R$ $(k = 1, \ldots, K)$ are arbitrary diagonality measures of a matrix and $\psi : \mathfrak{G} \to R$ is a joint diagonality criterion. Each $\psi_k(A)$ is supposed to take its minimum when A is diagonal, or as diagonal as possible. We derive the gradient flow of the potential function (2) on $\mathfrak{G}$ and suggest the use of the flow as an algorithm for joint diagonalization.

3. CONTINUOUS GRADIENT ALGORITHMS

Let $\mathfrak{G}$ be a Lie subgroup of the matrix Lie group $GL(n,C)$ and $\mathfrak{g}$ its Lie algebra.

We define the metric on the Lie algebra $\mathfrak{gl}(n,C)$ as

$$\mu(X,Y) = \sum_{i,j} (\hat{x}_{ij}\, \hat{y}_{ij} + \check{x}_{ij}\, \check{y}_{ij})$$

where $\hat{x}_{ij}$ and $\check{x}_{ij}$ are the real and imaginary parts of the (i,j)-th entry x_{ij} of X respectively.

For the derivation of the gradient equation on $\mathfrak{G}$, we use the metric μ by restricting the domain of μ to $\mathfrak{g}$. This metric on $\mathfrak{g}$, the tangent space of $\mathfrak{G}$ at the identity, is extended to the tangent space at each point of $\mathfrak{G}$ by the left action of $\mathfrak{G}$ and determines a Riemannian structure on $\mathfrak{G}$. Let $T_G\mathfrak{G}$ denote the tangent space of $\mathfrak{G}$ at $G \in \mathfrak{G}$. Note that an arbitrary element of $T_G\mathfrak{G}$ can be expressed as GX with $X \in \mathfrak{g}$. The metric on $T_G\mathfrak{G}$ is defined as

$$\mu_{T_G\mathfrak{G}}(GX, GY) = \mu(X,Y).$$

For a complex matrix $A \in M(n,C)$ and a real-valued function $f(A)$ on $M(n,C)$ we define a new matrix

$$\frac{df}{dA}$$

as a $n \times n$ complex matrix whose (i,j)-th entry is

$$\frac{\partial f(A)}{\partial \hat{a}_{ij}} + \frac{\partial f(A)}{\partial \check{a}_{ij}} i$$

where $\hat{a}_{ij}$ and $\check{a}_{ij}$ are the real and imaginary parts of the (i,j)-th entry a_{ij} of A respectively.

We use the following lemma for the derivation of the gradient equation.

Lemma. *Consider two functions $f : \mathfrak{G} \to R$ where $\mathfrak{G}$ is a Lie subgroup of $GL(n,C)$, $g : M(n,C) \to R$ and a matrix $A \in M(n,C)$ defined as*

$$A = G^{-1} A_0 G$$

where $A_0 \in M(n,C)$ is a fixed matrix and $G \in \mathfrak{G}$. Then the gradient direction of

$$f(G) \stackrel{\text{def}}{=} g(A) = g(G^{-1} A_0 G)$$

on $\mathfrak{G}$ with respect to the metric μ is given as

$$\begin{aligned} \text{grad} f &= G\, \pi_{\mathfrak{g}}[A^*, \frac{dg}{dA}] \\ &= G\, \pi_{\mathfrak{g}}[(G^{-1} A_0 G)^*, \frac{dg}{dA}(G^{-1} A_0 G)] \end{aligned}$$

where $\pi_{\mathfrak{g}}$ is the μ-orthogonal projection from $\mathfrak{gl}(n,C)$ to $\mathfrak{g}$ and $[A,B] = AB - BA$ is a commutator product.

Proof. Suppose that G is moving on $\mathfrak{G}$ and the time-derivative of G is expressed as $\frac{dG}{dt} = GX$ with $X \in \mathfrak{g}$, then the time-derivative of $A = G^{-1} A_0 G$ is calculated as

$$\frac{dA}{dt} = [A, X]$$

using that A_0 is fixed and $\frac{d}{dt} G^{-1} = -G^{-1} \frac{dG}{dt} G^{-1}$. Let $\{X_1, \ldots, X_D\}$ be a μ-orthonormal basis of $\mathfrak{g}$ on R. (Note that we treat $\mathfrak{gl}(n,C)$ not as a n^2-dimensional complex linear space but as a $2n^2$-dimensional real linear space and hence $\mathfrak{g}$ as a real linear subspace of the real linear space.) Then $\{GX_1, \ldots, GX_D\}$ is a $\mu_{T_G\mathfrak{G}}$-orthonormal basis of $T_G\mathfrak{G}$ on R. The directed derivative of $g(A) = g(G^{-1} A_0 G)$ at G along the direction GX_d is

$$\begin{aligned} & \sum_{i,j} \left\{ \frac{\partial g(A)}{\partial \hat{a}_{ij}} \frac{d\hat{a}_{ij}}{dt} + \frac{\partial g(A)}{\partial \check{a}_{ij}} \frac{d\check{a}_{ij}}{dt} \right\} \\ =\; & \mu(\left(\frac{\partial g(A)}{\partial \hat{a}_{ij}} + \frac{\partial g(A)}{\partial \check{a}_{ij}} i \right), \left(\frac{d\hat{a}_{ij}}{dt} + \frac{d\check{a}_{ij}}{dt} i \right)) \\ =\; & \mu(\frac{dg}{dA}, \frac{dA}{dt}) \\ =\; & \mu(\frac{dg}{dA}, [A, X_d]). \end{aligned}$$

Then the gradient direction of $g(A) = g(G^{-1} A_0 G)$ is calculated as

$$\sum_{d=1}^{D} \mu(\frac{dg}{dA}, [A, X_d]) G X_d$$

$$= \quad -G\sum_{d=1}^{D}\mu([\frac{dg}{dA},A^*],X_d)X_d$$
$$= \quad -G\,\pi_{\mathfrak{g}}[\frac{dg}{dA},A^*]$$
$$= \quad G\,\pi_{\mathfrak{g}}[A^*,\frac{dg}{dA}]$$

using that $\mu(X,[Y,Z]) = -\mu([X,Y^*],Z)$ holds for arbitrary $X,Y,Z \in M(n,C)$. ■

Using the lemma, the steepest descent equation on $\mathfrak{G}$ of the potential function

$$\psi(G) = \sum_{k=1}^{K}\psi_k(G^{-1}A_kG)$$

with respect to the metric μ is calculated as

$$\frac{dG}{dt} = -G\,\pi_{\mathfrak{g}}\sum_{k=1}^{K}[(G^{-1}A_kG)^*,\frac{d\psi_k}{dA}(G^{-1}A_kG)]. \quad (3)$$

Example 1. Gradient equation of sum of squares of non-diagonal entries

Let $\psi_k(A)$'s of (2) be defined as

$$\psi_k(A) = \mathbf{off}(A) = \sum_{i\neq j}(\hat{a}_{ij}^2+\check{a}_{ij}^2) \quad (k=1,\ldots,K).$$

Then, by differentiating $\psi_k(A)$ with respect to each entries of A, we get

$$\frac{d\psi_k}{dA} = 2(A-\mathrm{diag}A)$$

where $\mathrm{diag}A$ is the diagonal part of a matrix A. By setting ψ_k's as above and $\mathfrak{G} = GL(n,C)$ in (3), we obtain the following gradient equation of $G \in GL(n,C)$,

$$\frac{dG}{dt} = -2G\sum_{k=1}^{K}[(G^{-1}A_kG)^*,G^{-1}A_kG-\mathrm{diag}(G^{-1}A_kG)]$$

where $\pi_{\mathfrak{g}}(n,C)$ is omitted since it is an identity operator. By setting ψ_k's as the same and $\mathfrak{G} = U(n)$, we obtain the following equation of $U \in U(n)$,

$$\frac{dU}{dt} = -2U\,\pi_{\mathfrak{u}(n)}\sum_{k=1}^{K}[(U^*A_kU)^*,U^*A_kU-\mathrm{diag}(U^*A_kU)] \quad (4)$$

where $\pi_{\mathfrak{u}(n)}$, the μ-orthogonal projection from $\mathfrak{gl}(n,C)$ to $\mathfrak{u}(n)$, is given as

$$\pi_{\mathfrak{u}(n)}A = \frac{1}{2}(A-A^*). \quad (5)$$

Note that the orthogonality $\mu(\pi_{\mathfrak{u}(n)}A, A-\pi_{\mathfrak{u}(n)}A) = 0$ holds. Using (5), (4) is reduced to

$$\frac{dU}{dt} = 2U\,\pi_{\mathfrak{u}(n)}\sum_{k=1}^{K}[(U^*A_kU)^*,\mathrm{diag}(U^*A_kU)]. \quad (6)$$

(We can use $\pi_{\mathfrak{su}(n)}$ instead of $\pi_{\mathfrak{u}(n)}$ so that U stays in $SU(n)$, but this is not so important in a sense of numerical algorithm.) In the case that all the A_k's are Hermitian, the equation is reduced to

$$\frac{dU}{dt} = 2U\sum_{k=1}^{K}[U^*A_kU,\mathrm{diag}(U^*A_kU)]$$

where $\pi_{\mathfrak{u}(n)}$ can be omitted since the operand is an element of $\mathfrak{u}(n)$.

Example 2. Gradient equation of sum of absolute values of non-diagonal entries

Let $\psi_k(A)$'s of (2) be defined as

$$\psi_k(A) = \sum_{i\neq j}(|\hat{a}_{ij}|+|\check{a}_{ij}|) \quad (k=1,\ldots,K).$$

While $|x|$ is not differentiable at $x=0$, we treat the derivative of $|x|$ at $x=0$ as 0 from a viewpoint of practical algorithm. Then $\frac{d\psi_k}{dA}$ is a matrix whose (i,j)-th entry is

$$\left(\frac{d\psi_k}{dA}\right)_{ij} = \begin{cases} 0 & (i=j) \\ \mathrm{sign}_0(\hat{a}_{ij})+\mathrm{sign}_0(\check{a}_{ij})i & (i\neq j)\end{cases}$$

where $\mathrm{sign}_0(x)$ is defined as

$$\mathrm{sign}_0(x) = \begin{cases} 1 & (0<x) \\ 0 & (x=0) \\ -1 & (x<0)\end{cases}. \quad (7)$$

By setting $\mathfrak{G} = GL(n,C)$ or $\mathfrak{G} = U(n)$ for this potential function in (3), we can obtain gradient equations parallel to the equations of the previous example. Using $\mathrm{sign}_\epsilon(x)$ defined as follows instead of $\mathrm{sign}_0(x)$ is suitable for practical implementation,

$$\mathrm{sign}_\epsilon(x) = \begin{cases} 1 & (\epsilon<x) \\ 0 & (-\epsilon\leq x\leq\epsilon) \\ -1 & (x<-\epsilon)\end{cases}$$

where ϵ is a small positive number.

4. DISCRETE ALGORITHMS

Continuous matrix dynamics introduced in the previous section are computationally expensive for numerical integration and not suitable for practical algorithm. We derive discrete algorithms from continuous matrix dynamics utilizing matrix exponential function.

One-parameter set

$$G(t) = G(0)\exp(tX), \quad G(0) \in \mathfrak{G}, X \in \mathfrak{g}, t \in R$$

is something like a straight line on a Lie group $\mathfrak{G}$ which is through $G(0) \in \mathfrak{G}$ and with the direction $GX \in T_G\mathfrak{G}$ and it is reasonable to utilize large steps along the "straight line" to make a computationally cheap discrete approximation of a continuous matrix dynamics,

$$\frac{dG(t)}{dt} = G(t)X(t) \quad \Rightarrow \quad G_{t+1} = G_t \exp(\alpha_t X_t)$$

where α_t is an adaptive stepsize.

5. NUMERICAL EXPERIMENTS

In this section, we benchmark the performance of our new method featuring the discrete version of (6) with exponential stepsize scheduling,

$$U_{t+1} = U_t \exp(ct^{-\lambda}\frac{X_t}{||X_t||}),\ U_0 = I,$$

$$X_t = \pi_{\mathfrak{u}(n)} \sum_{k=1}^{K} [({U_t}^* A_k U_t)^*, \mathrm{diag}({U_t}^* A_k U_t)],$$

where c and λ are positive constants and $||\cdot||$ denotes the Frobenius norm, $||X|| = \sqrt{\sum |x_{ij}|^2}$.

To compare the early phase performance of our new method and the extended Jacobi method for joint diagonalization by Cardoso and Souloumiac[3], we diagonalize bechmark matrices simultaneously using the first 10 steps of our new method and the extended Jacobi method with the threshold of the rotation angle 0.1.

Experiments are done on the windows version of Matlab on the machine with PentiumIII 500MHz and 287Mbyte RAM. The elapsed time and the number of floating point operations used through each calculations are counted using the Matlab `tic-toc` and `flops` utilities.

The eigenvalues of the benchmark matrices are random complex numbers whose real and imaginary parts are normally distributed random numbers generated by the Matlab `randn()` function and a common similar transformation by a random unitary matrix is applied to these eigenvalues. The number of matrices $K = 4$ and the size of matrices $n = 50, 100$ and 200. Each elements of matrices are contaminated by `0.001*randn()`.

Both methods are written in usual Matlab scripts. The matrix exponential function in our new method is replaced with it's first order Páde approximation. The constants in our new method are decided through trial and error as $c = 5.0$ and $\lambda = 0.5$ for $K = 50$, $c = 7.0$ and $\lambda = 0.4$ for $K = 100$ and $c = 10.0$ and $\lambda = 0.4$ for $K = 200$.

Experiments are done 10 times for each size of matrices with randomly generated benchmark matrices and average results are given in the following tables where (gain) $= -\log \sum \mathbf{off}(\text{end}) / \sum \mathbf{off}(\text{begin})$.

New method (first 10 steps)

n	50	100	200
gain	1.30	0.956	0.763
`tic-toc`/gain	0.723 sec	6.39 sec	89.6 sec
`flops`/gain	8.90×10^7	9.51×10^8	9.42×10^9

Extended Jacobi method (threshold=0.1)

n	50	100	200
gain	1.62	1.17	0.756
`tic-toc`/gain	2.33 sec	18.1 sec	175 sec
`flops`/gain	1.49×10^7	1.35×10^8	1.34×10^9

The reversal in the results of `tic-toc` and `flops` is due to the Matlab's nature (which is more or less common to recent parallel computing environments) that matrix calculations such as matrix addition and matrix multiplication are faster than calculations with access to each element of matrix. Our new method uses former and the extended Jacobi method does latter.

6. REMARKS

The above results of numerical experiments show that there is a possibility that our new method does better than the extended Jacobi method in early phase of calculation of joint diagonalization on some computational environment. On the other hand, the extended Jacobe method is proved to have local quadratic convergence[1]. This suggests us using our new method in early phase of computation and switching to the extended Jacobi method in final phase.

The performance of our new method depends seriously on the choice of the positive constants c and λ. We can do some rigorous analysis of optimal stepsize, but rigorous solution to the optimal stepsize is probably computationally heavy and will not improve the performance. We need to construct some computationally cheap way to estimate the optimal stepsize.

It is observed by numerical integration using the Euler method that continuous matrix dynamics introduced in Section 3 also converge to desired equilibrium points from some initial points. This gives a rise to interest in hardware implementation of these dynamics on analogue circuits which will be suitable for on-line applications such as ICA for tracking moving targets.

7. REFERENCES

[1] A.Bunse-Gerstner, R.Byers and V.Mehrmann, "Numerical methods for simultaneous diagonalization", *SIAM J.Mat.Anal.Appl.*, 14, pp.927–949, 1993.

[2] J.-F.Cardoso and A.Souloumiac, "Blind beamforming for non Gaussian signals", *IEEE Proc.-F*, 140, pp.362–370, 1993.

[3] J.-F.Cardoso and A.Souloumiac, "Jacobi angles for simultaneous diagonalization", *SIAM J.Mat.Anal.Appl.*, 17, pp.161–164, 1996.

[4] L.Molgedey and H.G.Schuster, "Separation of a mixture of independent signals using time delayed correlations", *Phys.Rev.Lett.*, 72, pp.3634–3637, 1994.

[5] R.W.Brockett, "Differential geometry and the design of gradient algorithms", in *Differential Geometry*, Vol.1, R.E.Green and S.-T.Yau, eds., Amer.Math.Soc., Providence, pp.69–91, 1993.

[6] U.Helmke and J.B.Moore, "Optimization and dynamical systems", Springer-Verlag, London, 1994.