Lie Group Methods for Optimization with Orthogonality Constraints

Mark D. Plumbley

Department of Electronic Engineering, Queen Mary University of London,
Mile End Road, London E1 4NS, United Kingdom.
Email: mark.plumbley@elec.qmul.ac.uk

Abstract. Optimization of a cost function $J(\mathbf{W})$ under an orthogonality constraint $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ is a common requirement for ICA methods. In this paper, we will review the use of *Lie group* methods to perform this constrained optimization. Instead of searching in the space of $n \times n$ matrices $\mathbf{W}$, we will introduce the concept of the Lie group $\mathrm{SO}(n)$ of orthogonal matrices, and the corresponding *Lie algebra* $\mathfrak{so}(n)$. Using $\mathfrak{so}(n)$ for our coordinates, we can multiplicatively update $\mathbf{W}$ by a rotation matrix $\mathbf{R}$ so that $\mathbf{W}' = \mathbf{R}\mathbf{W}$ always remains orthogonal. Steepest descent and conjugate gradient algorithms can be used in this framework.

1 Introduction

The independent component analysis problem has a natural 2-step solution: *whitening* followed by *orthogonal rotation* [1]. Given pre-whitened observation vectors $\mathbf{z}$, and a linear transformation $\mathbf{y} = \mathbf{W}\mathbf{z}$, the latter 'rotation' step requires optimization of some function $J = J(\mathbf{W})$ subject to an orthogonality constraint $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ on the solution. For standard ICA, J is typically a kurtosis or negentropy measure. For non-negative ICA, we can use a mean squared negativity measure $J = \frac{1}{2}E(|\mathbf{y}_-|^2)$ where $[\mathbf{y}_-]_i = \min(y_i, 0)$ is a negative-rectified version of the output $\mathbf{y}$ [2]. A simple approach to function minimization would be to perform steepest-descent search, $\mathbf{W}_{k+1} = \mathbf{W}_k - \eta\nabla_{\mathbf{W}}J$ where η is a small constant and $[\nabla_{\mathbf{W}}J]_{ij} = \partial J/\partial w_{ij}$ is the gradient of J in $\mathbf{W}$-space. For example, for non-negative ICA, we have $\nabla_{\mathbf{W}}J = E(\mathbf{y}_-\mathbf{z}^T)$. However, this ignores the constraint $\mathbf{W}\mathbf{W}^T = \mathbf{I}$: how do we find a minimum of J, subject to the constraint $\mathbf{W}\mathbf{W}^T = \mathbf{I}$?

One approach is to modify J with the addition of a penalty term which has a minimum when the constraint is satisfied. Another is to re-impose the constraint (e.g. through Gram-Schmidt orthogonalization) after each update of $\mathbf{W}$. We can also restrict changes to $\mathbf{W}$ so that components in any direction that would change the quantity $\mathbf{W}\mathbf{W}^T - \mathbf{I}$ are eliminated, and self-stabilized algorithms can be constructed that tend to reduce deviations away from $\mathbf{W}\mathbf{W}^T \approx \mathbf{I}$ (see e.g. [3]). However, these methods do not constrain $\mathbf{W}$ to be orthogonal at all times: we are continually having to fight against the tendency of $\mathbf{W}$ to "drift away" from the constraint surface.

In this paper we will briefly review an approach to this problem that has gained some interest recently: the *Lie group* method [4, 5]. In this approach we represent the possible movements of $\mathbf{W}$ using a set of coordinates which only allows $\mathbf{W}$ to take values on a *manifold* that satisfies the constraints. The coordinates $\mathbf{B}$, forming a *Lie algebra*, identify a matrix $\mathbf{W}$ in our Lie group and manifold using the exponential map $\mathbf{W} = \exp \mathbf{B}$. We will see that this elegant property allows us to construct methods for ICA type problems which always maintain the orthogonality constraints we require.

2 Illustration: The Lie group of unit-length complex numbers

Consider the complex numbers $z = x + iy$. If we want to 'move about' in the space of complex numbers, addition is a simple way to do it. However, suppose we were only interested in the unit-length complex numbers, those for which $|z| = 1$. These are no longer "closed" under addition: e.g. while 1 and i are each unit-length, $1 + i$ is not.

However, the unit-length complex numbers are closed under *multiplication*. To see this, it is convenient to use the 'length and angle' notation $z = re^{i\theta}$ (Fig. 1(a)). We would know that $r = 1$ for any unit-length complex number,

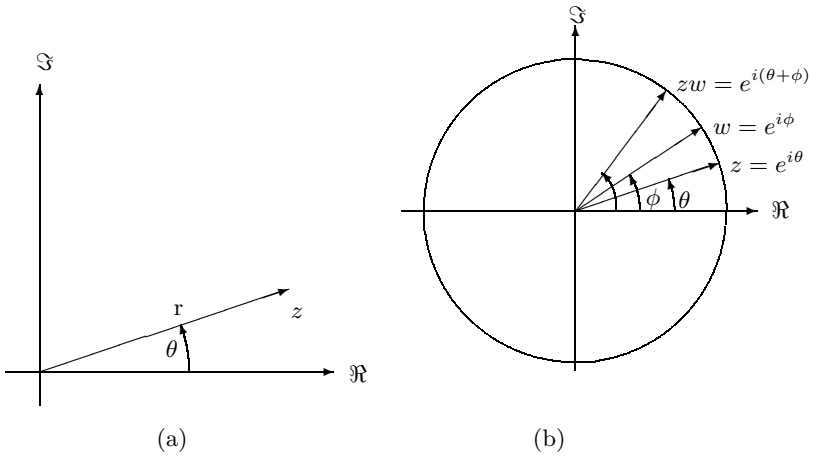

Fig. 1. Complex value (a) z in r-θ notation, and (b) product of unit-length complex numbers$z$ and w

so we could simply write $z = e^{i\theta}$. We can now see that multiplication of any two unit-length complex numbers z and $w = e^{i\phi}$ will give us a third unit-length complex number $zw = e^{i(\theta+\phi)}$. In fact, the unit-length complex numbers form a *group* G. It is easy to check that they satisfy the group axioms:

1. Closed under the operation: if $z, w \in G$, then $zw = y \in G$;

2. Associativity: $z(wy) = (zw)y$ for $z, w, y \in G$;
3. Identity element: $I \in G$, such that $Iz = zI = z$;
4. Each element has an inverse: z^{-1} such that $z^{-1}z = zz^{-1} = I$;

Our unit-length complex numbers are also commutative, i.e. $zw = wz$, so this is a commutative, or *Abelian* group.

While addition of unit-length complex numbers is of no use to us here, we notice that multiplication of complex numbers has a direct correspondence to addition of angles θ. So another way to 'move about' in the space of unit-length complex numbers is to add *angles* to get a desired angle θ', then convert this to a unit-length complex number using the exponential function $z' = e^{i\theta'}$.

This also makes clear another property of our group: it is "smooth". A local region looks like the real line $\mathbb{R}$, and we can use real-valued coordinates (e.g. the angles θ) to describe a path over this local region. Every local region can be given a coordinate system $\mathbb{R}$, and any overlaps between regions can be 'stitched together' smoothly, so our group with this set of coordinate systems forms a *manifold* [6]. The fact that each local coordinate system is one-dimensional ($\mathbb{R}$) means that we have a one-dimensional manifold. The smoothness in the group means that we have what is called a *Lie group*. The theory of Lie groups is particularly important in physics, such as for general relativity, and has more recently been used in robotics and computer vision. The key to how this helps with our optimization is how we form *derivatives* over this group.

Let $z(t) = e^{i\theta(t)}$ be a unit-length complex number changing with time. Then the time derivative of z is

$$dz/dt = \frac{d}{dt}e^{i\theta(t)} = i(d\theta/dt)e^{i\theta(t)} = i\omega z \tag{1}$$

where $\omega = d\theta/dt$. In other words, the derivative of z is proportional to iz, which is at right angles to z (Fig. 2). This means, in particular, that the derivative at z

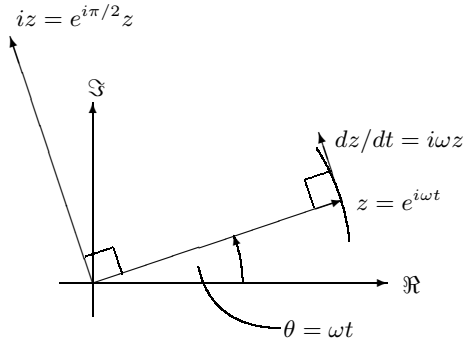

Fig. 2. Derivative of $z = \exp(i\omega t)$

is *tangent* to the group, pointing 'along' the direction that z changes. The space of all possible derivatives at z is called the *tangent space* T_z at z. We have to be

careful to say 'at z' since the tangent spaces T_w and T_z at different unit-length complex numbers w and z will be different.

How does this help with our optimization over orthogonal matrices $\mathbf{W}\mathbf{W}^T = \mathbf{I}$? It turns out that the set of orthogonal 2×2 matrices also forms a manifold and Lie group which is very similar to the unit-length complex numbers, so these ideas carry over directly. More generally, orthogonal $n \times n$ matrices can be decomposed into a *block diagonal* form of 2×2 matrices, the *Jordan canonical form* [6, 7], so knowing how these unit-length complex numbers behave (and hence the 2×2 matrices) is key to visualizing the general case.

3 The Lie group of (special) orthogonal matrices

For any integer $n \geq 1$, the set of $n \times n$ orthogonal matrices $\mathbf{W}$ with real entries, i.e. those real matrices that satisfy $\mathbf{W}\mathbf{W}^T = \mathbf{I}$, form a group under matrix multiplication, given the symbol $\mathrm{O}(n)$. We can check the group axioms if we wish: for example, if $\mathbf{W}$ and $\mathbf{Z}$ are orthogonal, then $\mathbf{V} = \mathbf{W}\mathbf{Z}$ is also orthogonal since $\mathbf{V}\mathbf{V}^T = \mathbf{W}\mathbf{Z}(\mathbf{W}\mathbf{Z})^T = \mathbf{W}\mathbf{Z}\mathbf{Z}^T\mathbf{W}^T = \mathbf{W}\mathbf{W}^T = \mathbf{I}$.

However, further investigation reveals that the Lie group $\mathrm{O}(n)$ actually consists of two disconnected parts: those matrices with determinant $+1$, and those with determinant -1. We cannot get smoothly from one half of $\mathrm{O}(n)$ to the other: every time we multiply by a matrix with determinant -1 we 'flip' from one half to the other. We therefore restrict ourself to the so-called 'special' orthogonal matrices $\mathrm{SO}(n)$, meaning those matrices in $\mathrm{O}(n)$ with determinant 1.

Consider for the moment the special case $n = 2$. The matrices in $\mathrm{SO}(2)$ can all be written in the following form:

$$\mathbf{W} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{2}$$

for some $0 \leq \theta < 2\pi$, and multiplication of these matrices satisfies

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} = \begin{pmatrix} \cos(\theta+\phi) & \sin(\theta+\phi) \\ -\sin(\theta+\phi) & \cos(\theta+\phi) \end{pmatrix} \tag{3}$$

so that multiplication of matrices $\mathbf{W} \in \mathrm{SO}(2)$ corresponds to addition of angles θ. We can therefore see that the group $\mathrm{SO}(2)$ with matrix multiplication behaves in exactly the same way as the unit-length complex numbers with complex number multiplication. Both can be specified completely by an angle $0 \leq \theta < 2\pi$, with the operation of addition modulo 2π. (Groups that "act the same" like this are said to be *isomorphic*.) So given some $\mathbf{W} = \left(\begin{smallmatrix} c & s \\ -s & c \end{smallmatrix}\right) \in \mathrm{SO}(2)$, to move to a new matrix we just need to find the angle $\theta = \arctan(s, c)$, move to a new angle θ', then transform to the new matrix $\mathbf{W}' = \left(\begin{smallmatrix} \cos\theta' & \sin\theta' \\ -\sin\theta' & \cos\theta' \end{smallmatrix}\right)$. The constraint $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ is maintained automatically.

Now, to apply a gradient-based search method, we will need to calculate derivatives. For the special case of $n = 2$, if we let $\theta = t\phi$ and differentiate

$\mathbf{W} = \left(\begin{smallmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{smallmatrix}\right)$ with respect to t, we get

$$\frac{d}{dt}\mathbf{W} = \begin{pmatrix} -\sin\theta & \cos\theta \\ -\cos\theta & -\sin\theta \end{pmatrix} \cdot \phi = \begin{pmatrix} 0 & \phi \\ -\phi & 0 \end{pmatrix} \mathbf{W}. \tag{4}$$

Now we know that for scalars, if $dz/dt = bz$ then $z = e^{tb}$. We can check that this also works for matrices. The matrix exponent of $t\mathbf{B}$ is, by definition

$$\exp(t\mathbf{B}) = \mathbf{I} + t\mathbf{B} + \frac{t^2\mathbf{B}^2}{2!} + \cdots + \frac{t^k\mathbf{B}^k}{k!} + \cdots \tag{5}$$

from which is is straightforward to verify that

$$\frac{d}{dt}\exp(t\mathbf{B}) = \mathbf{0} + \mathbf{B} + \mathbf{B}\frac{t\mathbf{B}}{1!} + \cdots + \mathbf{B}\frac{t^{k-1}\mathbf{B}^{k-1}}{(k-1)!} + \cdots = \mathbf{B}\exp(t\mathbf{B}) \tag{6}$$

and therefore (since $\theta = t\phi$) we can write

$$\mathbf{W} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} = \exp\Theta \qquad \text{where} \qquad \Theta = \begin{pmatrix} 0 & \theta \\ -\theta & 0 \end{pmatrix}. \tag{7}$$

(Note that we must be rather careful with this matrix exponential: since matrix multiplication is not commutative, $\exp(\mathbf{A}+\mathbf{B}) \neq \exp(\mathbf{A})\exp(\mathbf{B})$ in general.) In fact, given any skew-symmetric $\Phi \neq \mathbf{0}$, all of the elements in SO(2) can be specified by a single real parameter t, as $\mathbf{W}(t) = \exp(t\Phi)$.

For optimization over our 2×2 orthogonal matrices SO(2), we now have a clear way to proceed: instead of searching over the space of matrices $\mathbf{W}$, we can search over the space of angles θ, or alternatively, the space of skew-symmetric matrices Θ. To get back to an orthogonal matrix, we apply the exponential map $\mathbf{W} = \exp\Theta$.

4 Searching over SO(n) using the Lie algebra $\mathfrak{so}(n)$

The ideas that we have seen for SO(2) generalize to SO(n). However, we have to be a little careful for $n > 2$, due to the non-commutation of matrices that we mentioned earlier, meaning that $\exp(\mathbf{A})\exp(\mathbf{B}) \neq \exp(\mathbf{B})\exp(\mathbf{A})$. This is true even for very small $\mathbf{A}$ and $\mathbf{B}$: for small scalar ϵ we find

$$\exp(\epsilon\mathbf{A})\exp(\epsilon\mathbf{B}) - \exp(\epsilon\mathbf{B})\exp(\epsilon\mathbf{A}) = \epsilon^2[\mathbf{A},\mathbf{B}] + O(\epsilon^3) \tag{8}$$

where the matrix commutator, or *bracket*, $[\cdot,\cdot]$ is defined by $[\mathbf{A},\mathbf{B}] = \mathbf{AB} - \mathbf{BA}$.

Interestingly, the commutator of two skew-symmetric matrices $\mathbf{A} = -\mathbf{A}^T$ and $\mathbf{B} = -\mathbf{B}^T$ is itself skew-symmetric: $[\mathbf{A},\mathbf{B}]^T = -[\mathbf{A},\mathbf{B}]$. This set of $n \times n$ skew-symmetric matrices, closed under addition, multiplication by scalars, and the bracket operation, is an example of a *Lie algebra*, and is denoted $\mathfrak{so}(n)$. We can therefore map from an element $\mathbf{W}$ in the Lie group SO(n) to an element $\mathbf{B}$ in the Lie algebra $\mathfrak{so}(n)$ using the matrix logarithm operator (the inverse of

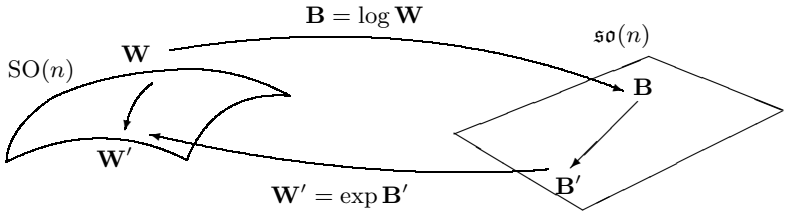

Fig. 3. Motion from $\mathbf{W}$ on the Lie group $\mathrm{SO}(n)$ by mapping $\mathbf{B}$ to the Lie algebra $\mathfrak{so}(n)$, moving to $\mathbf{B}' \in \mathfrak{so}(n)$, and mapping back to $\mathbf{W}' \in \mathrm{SO}(n)$.

$\exp(\cdot)$), move to a new element $\mathbf{B}' \in \mathfrak{so}(n)$, and then map from $\mathbf{B}'$ to $\mathbf{W}' = \exp(\mathbf{B}') \in \mathrm{SO}(n)$ using the matrix exponential (Fig. 3). Since $\mathfrak{so}(n)$ is a vector space, it is easy to stay in $\mathfrak{so}(n)$, because it is closed under addition of elements and multiplication by scalars. It also has dimension $n(n-1)/2$, the number of independent entries in an $n \times n$ skew-symmetric matrix, and hence is a "smaller" space to search over than the original n^2-dimensional space of matrices $\mathbf{W}$. In this way we can make sure that our matrices $\mathbf{W}$ always stay on $\mathrm{SO}(n)$.

In fact we can use a slightly modified form of this method, to avoid calculating the initial logarithm. Due to the group properties of $\mathrm{SO}(n)$, we know that $\mathbf{W}' = \mathbf{RW}$ for some "rotation" matrix $\mathbf{R} \in \mathrm{SO}(n)$. Moving from $\mathbf{W} = \mathbf{IW}$ to $\mathbf{W}' = \mathbf{RW}$ is equivalent to moving from $\mathbf{I}$ to $\mathbf{R}$, and we already know that $\log \mathbf{I} = \mathbf{0}$. Our modified method is therefore

1. Start at $\mathbf{0} \in \mathfrak{so}(n)$, equivalent to $\mathbf{I} \in \mathrm{SO}(n) = \exp(\mathbf{0})$
2. Move about in $\mathfrak{so}(n)$ from $\mathbf{0}$ to $\mathbf{B} \in \mathfrak{so}(n)$
3. Use $\exp(\cdot)$ to map back into $\mathrm{SO}(n)$, giving $\mathbf{R} = \exp(\mathbf{B})$
4. Calculate $\mathbf{W}' = \mathbf{RW} = \exp(\mathbf{B})\mathbf{W} \in \mathrm{SO}(n)$.

Of course we must start from a $\mathbf{W}$ that is orthogonal: we typically choose the identity matrix $\mathbf{W}_0 = \mathbf{I}$.

5 Gradient in $\mathfrak{so}(n)$ and geodesic flow

To choose the search direction in $\mathfrak{so}(n)$ ($\mathbf{B}$-space), we need to calculate the gradient of J in $\mathbf{B}$-space, $\nabla_{\mathbf{B}} J$. Calculating this, which must be skew-symmetric since the matrices $\mathbf{B}$ must remain skew-symmetric, we get [8]

$$\nabla_{\mathbf{B}} J = (\nabla_{\mathbf{W}} J)\mathbf{W}^T - \mathbf{W}(\nabla_{\mathbf{W}} J)^T. \tag{9}$$

This gives us the search direction for a steepest-descent search in $\mathbf{B}$-space, instead of using our original steepest-descent search in $\mathbf{W}$-space. For example, for non-negative ICA, we get $\nabla_{\mathbf{B}} J = E(\mathbf{y}_- \mathbf{y}^T - \mathbf{y}\mathbf{y}_-^T)$ which is clearly equivariant, since it only depends on $\mathbf{y}$ [9].

Using this approach, for the steepest-descent algorithm with small constant update factor η, we start at $\mathbf{B} = \mathbf{0} \in \mathfrak{so}(n)$, move to $\mathbf{B}' = -\eta \nabla_{\mathbf{B}} J$, map to $\mathbf{R} = \exp(\mathbf{B}') \in \mathrm{SO}(n)$, and finally perform a multiplicative update $\mathbf{W}_{k+1} = \mathbf{R}\mathbf{W}_k$. Putting this all into one equation, we get

$$\mathbf{W}_{k+1} = \exp(-\eta \nabla_{\mathbf{B}} J|_{\mathbf{B}=\mathbf{0}}) \mathbf{W}_k \tag{10}$$

which is the *geodesic flow* method introduced to ICA by Nishimori [7]. In the non-negative ICA case this gives $\mathbf{W}_{k+1} = \exp(-\eta E(\mathbf{y}_{-}\mathbf{y}^T - \mathbf{y}\mathbf{y}_{-}^T))\mathbf{W}_k$. Since $\mathbf{W}$ always remains in $\mathrm{SO}(n)$, the constraint $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ is maintained without the use of penalty functions or any constraint re-imposition [5].

6 Related methods

Now we have the basic approach of working in $\mathfrak{so}(n)$ instead of $\mathrm{SO}(n)$, we can also implement faster search methods equivalent to line search or conjugate gradients. For example, for a repeated line search (a *geodesic search* [2]), we can proceed as follows. We choose a search direction in the direction of steepest descent, i.e. $\mathbf{H} = -\nabla_{\mathbf{B}} J / |\nabla_{\mathbf{B}} J|$, make large steps along $\mathbf{B}(t) = t\mathbf{H}$ to get close to a minimum of J at t^*, update $\mathbf{W}_{k+1} = \exp(t^*\mathbf{H})\mathbf{W}_k$, and repeat with a new line search direction until the gradient or error is as small as we wish.

If we can calculate second derivative information, we can also use Newton updates in our line search to find the minimum. Also, due to the rotational structure of the group $\mathrm{SO}(n)$, we can also use a Fourier expansion in some situations [10].

Edelman, Arias and Smith [11] constructed conjugate gradient algorithms on Stiefel manifolds (a generalization of $\mathrm{SO}(n)$), and this approach was used by Martin-Clemente et al [12] for ICA. For the conjugate gradients method on manifolds such a $\mathrm{SO}(n)$ the basic idea is the same as in the more usual Euclidean space, but it needs a little care to ensure that the various gradients that are used are all 'transported' to the same point in an appropriate way before they are used [11].

In passing, we mention that the exponential map is not the only map from the skew-symmetric matrices $\mathfrak{so}(n)$ to the orthogonal matrices $\mathrm{SO}(n)$. For example, the Cayley transform $\mathbf{W} = (\mathbf{I} + \mathbf{B})/(\mathbf{I} - \mathbf{B})^{-1}$, can also be used for $\mathrm{SO}(n)$ and other related manifolds [13].

7 Conclusions

We have briefly reviewed the use of Lie group methods to optimize a cost function $J(\mathbf{W})$ under an orthogonality constraint $\mathbf{W}\mathbf{W}^T = \mathbf{I}$, a common requirement for ICA methods (including non-negative ICA). We have seen that we can search in the Lie algebra $\mathfrak{so}(n)$ of skew-symmetric matrices, instead of the original space of matrices, and multiplicatively update $\mathbf{W}$ by a rotation matrix $\mathbf{R}$ so that $\mathbf{W}' = \mathbf{R}\mathbf{W}$ always remains orthogonal. The simplest case of steepest descent over $\mathfrak{so}(n)$ corresponds to Nishimori's *geodesic flow*, and this approach can also be generalised to line search and conjugate gradient methods.

8 Acknowledgements

This work was partially supported by EPSRC grant GR/R54620, and by EU-FP6-IST-507142 project SIMAC (Semantic Interaction with Music Audio Contents: `www.semanticaudio.org`). An extended discussion of this subject will be presented in [8].

References

1. Comon, P.: Independent component analysis - a new concept? Signal Processing **36** (1994) 287–314
2. Plumbley, M.D.: Algorithms for nonnegative independent component analysis. IEEE Transactions on Neural Networks **14** (2003) 534–543
3. Douglas, S.C.: Self-stabilized gradient algorithms for blind source separation with orthogonality constraints. IEEE Transactions on Neural Networks **11** (2000) 1490–1497
4. Iserles, A.: Brief introduction to Lie-group methods. In Estep, D., Tavener, S., eds.: Collected Lectures on the Preservation of Stability Under Discretization (Proceedings in Applied Mathematics Series). SIAM (2002)
5. Fiori, S.: A theory for learning by weight flow on Stiefel-Grassman manifold. Neural Computation **13** (2001) 1625–1647
6. Schutz, B.: Geometrical Methods of Mathematical Physics. Cambridge University Press, Cambridge, UK (1980)
7. Nishimori, Y.: Learning algorithm for ICA by geodesic flows on orthogonal group. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN'99). Volume 2., Washington, DC (1999) 933–938
8. Plumbley, M.D.: Geometrical methods for non-negative ICA: Manifolds, Lie groups and toral subalgebras (2004) Submitted to *Neurocomputing*.
9. Cardoso, J.F., Laheld, B.H.: Equivariant adaptive source separation. IEEE Transactions on Signal Processing **44** (1996) 3017–3030
10. Plumbley, M.D.: Optimzation using Fourier expansion over a geodesic for non-negative ICA (2004) To appear in Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation, ICA2004.
11. Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM J. Matrix Anal. Appl. **20** (1998) 303–353
12. Martin-Clemente, R., Puntonet, C.G., Acha, J.I.: Blind signal separation based on the derivatives of the output cumulants and a conjugate gradient algorithm. In Lee, T.W., Jung, T.P., Makeig, S., Sejnowski, T.J., eds.: Proceedings of the International Conference on Independent Component Analysis and Signal Separation (ICA2001), San Diego, California. (2001) 390–393
13. Yamada, I., Ezaki, T.: An orthogonal matrix optimzation by dual Cayley parametrization technique. In: Proc. 4th Intl. Symp. On Independent Component Analysis and Blind Signal Separation (ICA2003), Nara, Japan. (2003) 35–40