GEOMETRY AND MANIFOLDS FOR INDEPENDENT COMPONENT ANALYSIS

Mark D. Plumbley

Centre for Digital Music, Department of Electronic Engineering
Queen Mary University of London, Mile End Road, London E1 4NS, UK
mark.plumbley@elec.qmul.ac.uk

ABSTRACT

In the last few years, there has been a great interest in the use of geometrical methods for independent component analysis (ICA), both to gain insight into the optimization process and to develop more efficient optimization algorithms. Much of this work involves concepts from differential geometry, such as Lie groups, Stiefel manifolds, or tangent planes that may be unfamiliar to signal processing researchers. The purpose of this tutorial paper is to introduce some of these geometry concepts to signal processing and ICA researchers, without assuming any existing background in differential geometry. The emphasis of the paper is on making the important concepts in this field accessible, rather than mathematical rigour.

Index Terms— Geometry, Optimization methods, Signal analysis, Learning systems, Neural networks

1. INTRODUCTION

In independent component analysis (ICA), we are given a set of observations assumed to be generated according to

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) \tag{1}$$

with real sources $\mathbf{s} = (s_1, \ldots, s_n)$ and an $n \times n$ mixing matrix $\mathbf{A}$. The task is to obtain an estimate of the original sources, assuming *independence* of the sources s_j.

Many ICA algorithms operate in two stages. Firstly, the data $\mathbf{x}$ is *pre-whitened* to give $\mathbf{z} = \mathbf{Q}\mathbf{x}$ where $\mathbf{z}$ has identity covariance $\mathbf{\Sigma}_\mathbf{z} = E((\mathbf{z}-\bar{\mathbf{z}})(\mathbf{z}-\bar{\mathbf{z}})^T) = \mathbf{I}$. We then search for an orthonormal matrix $\mathbf{W}$, i.e. with $\mathbf{W}$ satisfying $\mathbf{W}^T\mathbf{W} = \mathbf{W}\mathbf{W}^T = \mathbf{I}_n$, such that the outputs $\mathbf{y} = \mathbf{W}\mathbf{z} = \mathbf{W}\mathbf{Q}\mathbf{A}\mathbf{s}$ are as independent as possible.

This independence is typically measured by some cost or likelihood function. For example, we could use the following cost function based on negative log likelihood [1]

$$J(\mathbf{W}) = -\left(E\left[\textstyle\sum_i \log p_i(\mathbf{w}_i^T\mathbf{z})\right] + \log|\det \mathbf{W}|\right) \tag{2}$$

where we must find the minimum of J under the constraint that $\mathbf{W}$ is orthonormal.

Partly supported by EPSRC Grants GR/S75802/01, GR/S85900/01 and EP/C005554/1.

2. GRADIENT METHODS

Many algorithms are based on the method of *steepest descent* (or steepest ascent if J is to be maximized). The idea is to find the direction of fastest change in J for a given amount of change in $\mathbf{W}$, and update $\mathbf{W}$ by a small amount in that direction. Normally we calculate the gradient using $\nabla_\mathbf{W} J \equiv \partial J/\partial \mathbf{W}$ where $[\partial J/\partial \mathbf{W}]_{ij} = \partial J/\partial w_{ij}$ is the derivative of the cost function. For example, for J in (2) we get the gradient

$$\nabla_\mathbf{W} J = [\mathbf{W}^T]^{-1} - E[g(\mathbf{y})\mathbf{z}^T] \tag{3}$$

where $g(\mathbf{y}) = [g(y_1), \ldots, g(y_n)]^T$, $g(y) = -(d/dy)\log p(y)$. We then update $\mathbf{W}$ by

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \eta \nabla_\mathbf{W} J \tag{4}$$

for small $\eta > 0$, which we can also write as $\Delta\mathbf{W} = -\eta\nabla_\mathbf{W} J$.

2.1. Natural Gradient

In the classical Euclidean space that we are most familiar with, a small (squared) distance from $\mathbf{W}$ to $\mathbf{W} + \Delta\mathbf{W}$ is given by the squared length $l^2 = \|\mathbf{W} - (\mathbf{W} + \Delta\mathbf{W}))\|^2 = \|\Delta\mathbf{W}\|^2 = \sum_{ij} \Delta w_{ij}^2$. The *natural gradient* method, introduced by Amari [2] uses a different measure of length more suited to the multiplicative action of matrices: this is known as a *Riemannian metric* $\mathbf{G} = [g_{ij}]$. Using this new measure of length, the steepest descend direction for our ICA problem is given by the natural gradient

$$\tilde{\nabla}_\mathbf{W} J = (\nabla_\mathbf{W} J)\,\mathbf{W}^T\mathbf{W} \tag{5}$$

instead of the usual Euclidean gradient $\nabla_\mathbf{W} J$. In a Euclidean vector space however, the Riemannian metric is equal to the identity ($\mathbf{G} = \mathbf{I}$), so the steepest descent direction is (minus) the usual Euclidean gradient $\nabla_\mathbf{W} J$, and consequently we do not consider this to be an issue. (We also notice in passing that for orthonormal matrices, where $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, from (5) we have $\tilde{\nabla}_\mathbf{W} J = \nabla_\mathbf{W} J$.)

While the natural gradient has proved useful in many ICA algorithms, it does not directly help us to impose our orthogonality constraint. Therefore let us review some approaches that have been used to perform the constrained optimization we are looking for.

3. CONSTRAINED OPTIMIZATION APPROACHES

3.1. Penalty function method

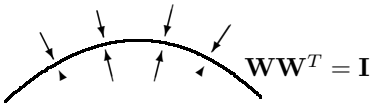

Fig. 1. Penalty function 'force' towards constraint surface.

A simple method often used to attempt to enforce a constraint is to construct a *penalty function* $P(\mathbf{W})$ which is zero when the constraint holds, and positive elsewhere. The idea is that this will tend to 'force' the optimization to find the constraint surface (Fig. 1). The hope is that the minimum of the combined target function $J_\mu = J + \mu P(\mathbf{W})$ will correspond to the desired constrained minimum J at $P(\mathbf{W}) = 0$. The weight μ is often chosen heuristically.

For example, we might choose the penalty function

$$\begin{aligned} P(\mathbf{W}) &= \|\mathbf{W}\mathbf{W}^T - \mathbf{I}_n\|_F^2 \\ &= \text{trace}((\mathbf{W}^T\mathbf{W} - \mathbf{I}_n)(\mathbf{W}^T\mathbf{W} - \mathbf{I}_n)) \end{aligned} \quad (6)$$

giving a new target function $J_\mu = J + \mu P(\mathbf{W})$ with gradient

$$\nabla J_\mu = \nabla J + \mu \nabla P(\mathbf{W})$$

and hence a new update algorithm

$$\Delta\mathbf{W} = -\eta(\nabla_\mathbf{W} J + \mu \nabla_\mathbf{W} P(\mathbf{W})) \quad (7)$$

$$= -\eta(\mathbf{I} - E[g(\mathbf{y})\mathbf{y}^T] + 4\mu(\mathbf{W}\mathbf{W}^T - \mathbf{I}))\mathbf{W} \quad (8)$$

However, the penalty function method is often not sufficient to force the constraint to be satisfied. For example, if the factor μ is too weak, the update may simply diverge away from the constraint surface. Furthermore, if the original cost function J does not itself have zero gradient at the desired constrained minimum, the penalty function method is not sufficient to give a constrained optimization. We would therefore like a better method, to constrain $\mathbf{W}$ to *remain* orthogonal.

3.2. Repeated constraint application

An alternative approach is to re-impose the constraint after each update step, making each update a 2-step process: (i) perform an unconstrained steepest descent update; (ii) reimpose the constraint. For example, we can repeatedly apply Gram-Schmidt Orthogonalization (GSO) after each update of $\mathbf{W}$, to ensure that $\mathbf{W}$ is orthonormal after each update (Fig. 2).

However, as illustrated in the figure, we can spend a significant amount of effort moving away from the constraint surface and back again, particularly as we get closer to the desired solution. If possible, we would really like to remove the tendency to move away from the constraint surface, so that we stay on or close to it.

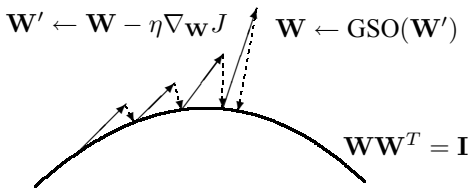

Fig. 2. Reimposition of constraint using GSO after each update.

3.3. Tangent direction updates

To reduce the tendency to move away from the constraint surface, we can make sure that the update $\Delta\mathbf{W}$ does not have a component pointing away from the surface. If we are on the constraint surface, such that $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, then we want any infinitesimal change $d\mathbf{W}$ to satisfy

$$\mathbf{0} = d(\mathbf{W}^T\mathbf{W} - \mathbf{I}) = (d\mathbf{W})^T\mathbf{W} + \mathbf{W}^T(d\mathbf{W}). \quad (9)$$

The set of $\Delta\mathbf{W}$ that satisfies $(\Delta\mathbf{W})^T\mathbf{W}_0 + \mathbf{W}_0^T(\Delta\mathbf{W}) = \mathbf{0}$ at a point $\mathbf{W}_0$ is known as the *tangent space* $T_{\mathbf{W}_0}$ at $\mathbf{W}_0$.

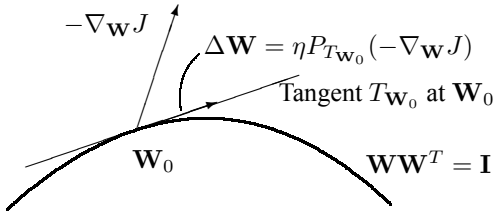

Fig. 3. Tanget update at $\mathbf{W}_0$.

So rather than reimposing the constraint *after* the update, we form the update by projecting the gradient $\nabla_\mathbf{W} J$ in to the tangent space (Fig. 3). For our ICA problem, this leads to the update

$$\Delta\mathbf{W} = -\frac{1}{2}\eta((\nabla_\mathbf{W} J)\mathbf{W}^T\mathbf{W} - \mathbf{W}(\nabla_\mathbf{W} J)^T\mathbf{W}) \quad (10)$$

$$= -\frac{1}{2}\eta E[g(\mathbf{y})\mathbf{y}^T - (g(\mathbf{y})\mathbf{y}^T)^T]\mathbf{W} \quad (11)$$

using the fact that $\mathbf{W}^T\mathbf{W} = \mathbf{W}\mathbf{W}^T = \mathbf{I}$.

While the tangent update method goes some way towards a solution, it is only a first order approximation, and we can still find a tendency to 'drift' away from the constrain surface.

It is possible to deal with this 'drift' problem, for example through tangent updates followed by occasional reimposition of the constraints, or by introducing higher-order terms to produce self-stabilizing algorithms [3]. However, there is an alternative approach designed specifically to maintain the constraint at all times during the updates: these are the *Lie group methods*.

4. LIE GROUP METHODS

4.1. Lie groups

The idea behind Lie group methods is to find a way to 'move about' on the constraint surface without ever leaving it. To get an idea of how we might do this, consider the set of unit-length complex numbers (Fig. 4). If we multiply the unit

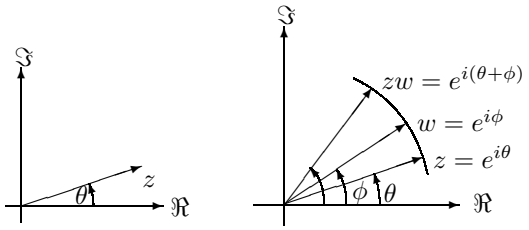

Fig. 4. Unit-length complex numbers.

length complex number $w = e^{i\phi}$ by another $z = e^{i\theta}$ we get $zw = e^{i(\theta+\phi)}$, which is itself also a unit-length complex number. Hence we have 'moved' from one unit length complex number (w) to another (zw), through multiplying by z. (Notice too that we can also view this as adding angles.)

In fact, the unit length complex numbers form a *group*. As a reminder, a group G is a set of elements together with a group operation which have the following properties: (i) Closure (if $z, w \in G$, then $zw = y \in G$); (ii) Associativity ($z(wy) = (zw)y$ for $z, w, y \in G$); (iii) Identity element ($I \in G$, such that $Iz = zI = z$); and (iv) Inverse (for each z there is some z^{-1} such that $z^{-1}z = zz^{-1} = I$). Certain groups are Abelian (commutative), such that $zw = wz$: this is the case for the unit length complex numbers, but is not true in general for matrices.

The unit length complex numbers also have the property that the group is *smooth*, in the sense that we can make infinitesimally small movements on it. In fact, locally it looks like a segment of real line $\mathbb{R}$ (if we flatten it out a bit). A group that is smooth, or *differentiable*, is called a *Lie group*.

4.2. Manifolds

At this point it is helpful to introduce the idea of a *manifold*. A *manifold* is a set where we can put a local m-dimensional coordinate system $\mathbb{R}^m$ on any small neighbourhood. We can join these local coordinate systems together, with overlaps as necessary, but without anything 'nasty' happening at the joins. Since we can move about in $\mathbb{R}^m$, using the usual vector addition operation, if we could work in these local coordinates we could (in theory) move about locally on our manifold without ever leaving it.

For a 'real-world' example, consider the surface of the earth. Locally it looks like $\mathbb{R}^2$. We can make a series of 2-dimensional *charts* covering the earth, with each chart having a corresponding *map* from a point on the chart to a point on the manifold. The set of all these charts forms an *atlas*. The charts should all overlap nicely, with no 'holes' left anywhere. We can therefore trace any journey from any point on the earth to any other by drawing a line across a sequence of charts. Therefore the surface of the earth forms a 2-dimensional manifold. We can also say that the manifold representing the earth is *embedded* in $\mathbb{R}^3$, since it naturally forms a subset of the 3-dimensional Euclidean space $\mathbb{R}^3$.

For the set of unit-length complex numbers (or circle), they locally look like $\mathbb{R}^1$. We can cover this circle with e.g. two segments of $\mathbb{R}^1$ with smooth overlaps. Alternatively, we can simply use a single 1-dimensional line segment to cover it, the angles $0 \leq \theta \leq 2\pi + \epsilon$, with a smooth overlap around the join where $\theta = 0$ and $\theta = 2\pi$ meet.

4.3. Manifold of Orthonormal Matrices

Let us now return to our original problem of optimization over the set of real orthonormal matrices. It turns out that the equation $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ imposes $n(n+1)/2$ constraints, and the set of orthonormal matrices forms a manifold with $n(n-1)/2 < n^2$ dimensions.

Furthermore, the set of orthonormal matrices also forms a group, called the *orthogonal group* $\mathrm{O}(n)$. To check this, we see that if $\mathbf{W}$ and $\mathbf{Z}$ are orthogonal ($\mathbf{W}, \mathbf{Z} \in \mathrm{O}(n)$), then for $\mathbf{V} = \mathbf{W}\mathbf{Z}$ we get $\mathbf{V}^T\mathbf{V} = \mathbf{Z}^T\mathbf{W}^T\mathbf{W}\mathbf{Z} = \mathbf{Z}^T\mathbf{Z} = \mathbf{I}$ so $\mathbf{V} \in \mathrm{O}(n)$; the identity matrix $\mathbf{I} \in \mathrm{O}(n)$ is the group identity, and so on. So we should be able to use the group operation (matrix multiplication) to move about on the manifold, instead of our classical addition operations.

One well-known way to perform some of these group operations in ICA is to use *Givens rotations* [4]:

$$\begin{pmatrix} u_{i_1}(k+1) \\ u_{i_2}(k+1) \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} u_{i_1}(k) \\ u_{i_2}(k) \end{pmatrix} \qquad (12)$$

where (i_1, i_2) is an axis pair to rotate over. This is one type of movement over $\mathrm{SO}(n)$, which roughly corresponds to moving parallel to an axis when moving in $\mathbb{R}^n$.

Now it turns out that the orthogonal group $\mathrm{O}(n)$ is actually composed of two disconnected subsets. For example, the matrices in $\mathrm{O}(2)$ are one of two types:

$$\text{(a)} \quad \mathbf{W} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \quad \text{(b)} \quad \mathbf{W} = \begin{pmatrix} \cos\theta & -\sin\theta \\ -\sin\theta & -\cos\theta \end{pmatrix}.$$

Type (a) have determinant 1, while type (b) have determinant -1. To get from an (a) to a (b) matrix we have to permute the matrix: imaging picking up a pair of vectors on a plane and flipping them over. The set of matrices of type (a) are called the 'special' orthogonal matrices, $\mathrm{SO}(n)$, and form a subgroup of $\mathrm{O}(n)$. The permutation ambiguity in ICA means that any optimum of J must exist in both halves of $\mathrm{O}(n)$, so we can restrict our search to the connected subgroup $\mathrm{SO}(n)$ only, without missing any important solutions.

To move about in $\mathrm{SO}(2)$ we can simply add angles:

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

For $n = 2$ this is an Abelian (commutative) group, and behaves like (is *isomorphic to*) unit-length complex numbers.

4.4. Differentiating on a Lie Group

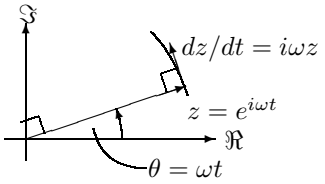

Fig. 5. Derivative of unit length complex number.

A Lie groups has a local smooth coordinate system, so we can differentiate functions on it. For unit-length complex numbers, with $z = \exp(i\omega t)$, the derivative is given by $dz/dt = i\omega \exp(i\omega t) = i\omega z$, and the derivative is *tangent to* the manifold surface at z (Fig. 5). In fact this pattern "$d\mathbf{W}/dt \propto \mathbf{W}$" also holds for $\mathrm{SO}(n)$. For example, for $\mathbf{W} = \left(\begin{smallmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{smallmatrix}\right) \in \mathrm{SO}(2)$ where $\theta = t\phi$, differentiating $\mathbf{W}$ wrt t, we get

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

which we could verify is tangent to the constraint surface at $\mathbf{W}$. Defining the matrix exponential operator

$$\exp(t\mathbf{B}) = \mathbf{I} + t\mathbf{B} + \frac{t^2\mathbf{B}^2}{2!} + \cdots + \frac{t^k\mathbf{B}^k}{k!} + \cdots, \quad (13)$$

which satisfies $(d/dt)\exp(t\mathbf{B}) = \mathbf{B}\exp(t\mathbf{B})$, we see that

$$\mathbf{W} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} = \exp\Theta, \quad \Theta = \begin{pmatrix} 0 & \theta \\ -\theta & 0 \end{pmatrix}.$$

Generalizing to $SO(n)$, the tangent space $T_{\mathbf{W}}$ at $\mathbf{W}$ is set of matrices $\Psi\mathbf{W}$ where $\Psi^T = -\Psi$ is skew-symmetric.

4.5. Lie algebras

The skew-symmetric matrices $\mathbf{B}^T = -\mathbf{B}$ form what is called a *Lie algebra* $\mathfrak{so}(n)$, related to those in the Lie group $\mathrm{SO}(n)$ through matrix exponentiation, $\mathbf{B} \in \mathfrak{so}(n) \mapsto \exp(\mathbf{B}) \in \mathrm{SO}(n)$. The advantage of working in the Lie algebra $\mathfrak{so}(n)$ is

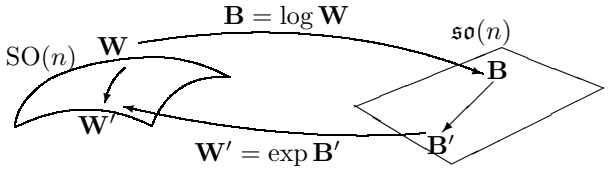

Fig. 6. Lie group update method

that it is a *vector space*, so we can add elements or multiply by scalars, yet still stay in the Lie algebra [5]. So to move about in $\mathrm{SO}(n)$, we use the matrix log to get to $\mathbf{B} = \log \mathbf{W} \in \mathfrak{so}(n)$, use addition to get to $\mathbf{B}' \in \mathfrak{so}(n)$, then use the matrix exponential to get to $\mathbf{W}' = \exp \mathbf{B}' \in \mathrm{SO}(n)$ (Fig. 6).

4.6. A geometric algorithm: geodesic flow

By defining an inner product and norm (length) in $\mathfrak{so}(n)$ such as $\langle \mathbf{B}, \mathbf{H} \rangle = \sum_{ij} b_{ij} h_{ij}/2$ with $l^2_{\mathbf{B}} = \langle \mathbf{B}, \mathbf{B} \rangle$ we can work out a steepest gradient descent direction. (A vector space with this type of norm is called a *Hilbert space*.) So for the gradient in $\mathbf{B}$-space we eventually get

$$\nabla_{\mathbf{B}} J = (\nabla_{\mathbf{W}} J)\mathbf{W}^T - \mathbf{W}(\nabla_{\mathbf{W}} J)^T. \quad (14)$$

By taking a small step in $\mathfrak{so}(n)$ in this direction, we get the *geodesic flow* method introduced to ICA by Nishimori [6]:

$$\mathbf{W}_{k+1} = \exp(-\eta \mathbf{G})\mathbf{W}_k \quad (15)$$

where $\mathbf{G} = \nabla_{\mathbf{B}} J|_{\mathbf{B}=\mathbf{0}} = (\nabla_{\mathbf{W}} J)\mathbf{W}^T - \mathbf{W}(\nabla_{\mathbf{W}} J)^T$. Since $\mathbf{G}$ is skew-symmetric, if $\mathbf{W}_k$ is orthonormal, then $\mathbf{W}_{k+1}$ will also be orthonormal. This is a constrained steepest descent search towards our required constrained optimization.

5. CONCLUSIONS

In this brief tutorial, we have seen that methods based on ideas from differential geometry can give us a way to tackle the type of constrained optimization found in ICA. In addition to the special orthogonal matrices $\mathrm{SO}(n)$ considered here, other manifolds of interest include the Stiefel manifold (the set of $n \times p$ full rank rectangular matrices), and the Grassmann manifold (the set of p-dimensional subspaces in $\mathbb{R}^n$), and Newton and conjugate gradients methods can also be used. For some further reading on constrained optimization using these Riemannian manifolds, see e.g. [7, 8].

While there are some remaining issues with these methods, including the cost of calculating (or approximating) the matrix exponential, they represent an interesting and promising approach to optimization for ICA and related tasks.

6. REFERENCES

[1] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*, John Wiley & Sons, 2001.

[2] S. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, pp. 251–276, 1998.

[3] S. C. Douglas, "Self-stabilized gradient algorithms for blind source separation with orthogonality constraints," *IEEE Trans. Neural Networks*, vol. 11, pp. 1490–1497, 2000.

[4] P. Comon, "Independent component analysis - a new concept?," *Signal Proc.*, vol. 36, pp. 287–314, 1994.

[5] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna, "Lie-group methods," *Acta Num.*, vol. 9, pp. 215–365, 2000.

[6] Y. Nishimori, "Learning algorithm for ICA by geodesic flows on orthogonal group," in *Proc. IJCNN'99*, pp. 933–938, 1999.

[7] S. T. Smith, "Optimization techniques on Riemannian manifolds," *Fields Inst. Com.*, vol. 3, pp. 113–136, 1994.

[8] D. Gabay, "Minimizing a differentiable function over a differential manifold," *J. Optim. Theory Appl.*, vol. 37, pp. 177–219, 1982.