Steepest Descent Algorithms for Optimization Under Unitary Matrix Constraint

Traian E. Abrudan, *Student Member, IEEE*, Jan Eriksson, *Member, IEEE*, and Visa Koivunen, *Senior Member, IEEE*

***Abstract*—In many engineering applications we deal with constrained optimization problems with respect to complex-valued matrices. This paper proposes a Riemannian geometry approach for optimization of a real-valued cost function $\mathcal{J}$ of complex-valued matrix argument $\mathbf{W}$, under the constraint that $\mathbf{W}$ is an $n \times n$ unitary matrix. We derive steepest descent (SD) algorithms on the Lie group of unitary matrices $U(n)$. The proposed algorithms move towards the optimum along the geodesics, but other alternatives are also considered. We also address the computational complexity and the numerical stability issues considering both the geodesic and the nongeodesic SD algorithms. Armijo step size [1] adaptation rule is used similarly to [2], but with reduced complexity. The theoretical results are validated by computer simulations. The proposed algorithms are applied to blind source separation in MIMO systems by using the joint diagonalization approach [3]. We show that the proposed algorithms outperform other widely used algorithms.**

***Index Terms*—Array processing, optimization, source separation, subspace estimation, unitary matrix constraint.**

I. Introduction

CONSTRAINED optimization problems arise in many signal processing applications. One common task is to minimize a cost function with respect to a matrix, under the constraint that the matrix has orthonormal columns. Some typical applications in communications and array signal processing are subspace tracking [4]–[6], blind and constrained beamforming [7]–[9], high-resolution direction finding (e.g., MUSIC and ESPRIT), and generally all subspace-based methods. Another straightforward application is the independent component analysis (ICA), [3], [10]–[19]. This type of optimization problem has also been considered in the context of multiple-input multiple-output (MIMO) communication systems [6], [20]–[23]. Most of the existing optimization algorithms are derived for the real-valued case and orthogonal matrices [10], [11], [13]–[15], [17], [24]–[27]. Very often in communications and signal processing applications we are dealing with complex matrices and signals. Consequently, the optimization need to be performed under unitary matrix constraint.

Commonly optimization algorithms employing orthogonal/unitary matrix constraint minimize a cost function on the space of $n \times n$ matrices using a classical steepest descent (SD) algorithm. Separate orthogonalization procedure must be applied after each iteration [12], [20]–[22]. Approaches stemming from the Lagrange multipliers method have also been used to solve such problems [16]. In such approaches, the error criterion contains an extra term that penalizes for the deviations from orthogonality property. Self-stabilized algorithms have been developed to provide more accurate, but still approximate solutions [17]. Major improvements over the classical methods are obtained by taking into account the geometrical aspects of the optimization problem. Pioneering work by Luenberger [28] and Gabay [29] convert the constrained optimization problem into an unconstrained problem, on an appropriate differentiable manifold. An extensive treatment of optimization algorithms with orthogonality constraints is given later by Edelman *et al.* [24] in a Riemannian geometry context. A non-Riemannian approach has been proposed in [2], which is a general framework for optimization under unitary matrix constraint. A more-detailed literature review is presented in Section II.

In this paper we derive two generic algorithms stemming from differential geometry. They optimize a real-valued cost function $\mathcal{J}(\mathbf{W})$ with respect to *complex-valued* matrix $\mathbf{W}$ satisfying $\mathbf{W}\mathbf{W}^H = \mathbf{W}^H\mathbf{W} = \mathbf{I}$, i.e., perform optimization subject to the unitary matrix constraint. SD algorithms operating on the Lie group of unitary matrices $U(n)$ are proposed. They move towards the optimum along the locally shortest paths, i.e., geodesics. Geodesics on Riemannian manifolds correspond to the straight lines in Euclidean space. Our motivation to opt for the geodesic algorithms is that on the Lie group of unitary matrices $U(n)$, the geodesics have simple expressions described by the exponential map. We can fully exploit recent developments in computing the matrix exponential needed in the multiplicative update on $U(n)$. The generalized polar decomposition [30] proves to be one of the most computationally efficient method if implemented in a parallel architecture, or the Cayley transform (CT) [31] otherwise. We also consider other parametrizations proposed in the literature and show that all these parametrizations are numerically equivalent up to a certain approximation order. However, the algorithms differ in terms of computational complexity, which is also addressed in this paper. The proposed generic geodesic algorithms, unlike other parametrizations, can be relatively easily adapted to different problems with varying complexity and strictness of the unitarity property requirements. This is due the fact that the computation of the matrix exponential function employed in the proposed algorithms is a well-researched problem [32] with some recent progress relevant to the unitary optimization [30]. Moreover, we show that the expo-

Manuscript received January 30, 2007; revised August 13, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sergiy Vorobyov. This work was supported in part by the Academy of Finland and by the GETA Graduate School.

The authors are with the SMARAD CoE, Signal Processing Laboratory, Department of Electrical Engineering, Helsinki University of Technology, FIN-02015 HUT, Finland (e-mail: tabrudan@wooster.hut.fi; jamaer@wooster.hut.fi; visa@wooster.hut.fi).

Digital Object Identifier 10.1109/TSP.2007.908999

nential map is well suited for adapting the step size for the SD method on the unitary group.

This paper is organized as follows. In Section II, an overview of the problem of optimization under unitary matrix constraint is provided. A brief review of different approaches presented in the literature is given as well. A simple geometric example is used to illustrate the differences among various approaches. In Section III, we derive the Riemannian gradient on the Lie group of unitary matrices and the corresponding SD algorithms. Equivalence relationships between the proposed algorithms and other algorithms are established in Section IV. The computational complexity and the numerical stability issues are studied in Sections V and VI, respectively. Simulation results are presented in Section VII. The proposed algorithms are used to solve the unitary matrix optimization problem encountered in the joint approximate diagonalization of eigenmatrices (JADE) algorithm [3] which is applied for blind source separation in a MIMO system. Finally, Section VIII concludes the paper.

II. Optimization Under Unitary Matrix Constraint

In this section, a brief overview of optimization methods under orthonormal or unitary matrix constraint is provided. Different approaches are reviewed and the key properties of each approach are briefly studied. A simple example is presented to illustrate how each algorithm searches for the optimum.

A. Overview

Most of classical optimization methods with unitary matrix constraint operate on the Euclidean space by using a SD algorithm. The unitary property of the matrix is lost in every iteration, and it needs to be restored in each step. Moreover, the convergence speed is reduced. Other algorithms use a Lagrangian type of optimization, by adding an extra-penalty function which penalizes for the deviation from unitarity [16]. These methods suffer from slow convergence and find only an approximate solution in terms of orthonormality. Self-stabilized algorithms provide more accurate solutions [17], [33].

A major drawback of the classical Euclidean SD and Lagrange type of algorithms [12], [16], [20]–[22] is that they do not take into account the special structure of the parameter space where the cost function needs to be optimized. The constrained optimization problem may be formulated as an unconstrained one in a different parameter space called *manifold*. Therefore, the space of unitary matrices is considered to be a "constrained surface." Optimizing a cost function on a manifold is often considered [10], [14], [24], [25], [29], [34] as a problem of Riemannian geometry [35]. Algorithms more general than the traditional Riemannian approach are considered in [2].

The second important aspect neglected in classical algorithms is that the $n \times n$ unitary matrices are algebraically closed under the multiplication operation, not under addition. Therefore, they form a group under the multiplication operation, which is the *Lie group* of $n \times n$ unitary matrices, $U(n)$ [36]. Consequently, by using an iterative algorithm based on an additive update the unitarity property is lost after each iteration. Even though we are moving along a straight line pointing in the right direction, we depart from the constrained surface in each step. This happens because a Riemannian manifold is a "curved space." The locally length-minimizing curve between two points on the Riemannian manifold is called a *geodesic* and it is not a straight line like on the Euclidean space. Several authors [10], [11], [13], [14], [24], [25], [28], [29], [34], [37], [38] have proposed that the search for the optimum should proceed along the geodesics of the constrained surface. Relevant work in Riemannian optimization algorithms may be found in [24], [29], [34], and [38]–[41]. Algorithms considering the *real-valued* Stiefel and/or Grassmann manifolds have been proposed in [10], [11], [15], [17], [24]–[26], and [42]. Edelman *et al.* [24] consider the problem of optimization under orthonormal constraints. They propose SD, conjugate gradient, and Newton algorithms along geodesics on Stiefel and Grassman manifolds.

A general framework for optimization under unitary matrix constraints is presented in [2]. It is not following the traditional Riemannian optimization approach. A modified SD algorithm, coupled with Armijo's step size adaptation rule [1] and a modified Newton algorithm are proposed for optimization on both the complex Stiefel and the complex Grassmann manifold. These algorithms do not employ a geodesic motion, but geodesic motion could be used in the general framework. A local parametrization based on an Euclidean projection of the tangent space onto the manifold is used in [2]. Hence, the computational cost may be reduced. Moreover, it is suggested that the geodesic motion is not the only solution, since there is no direct connection between the Riemannian geometry of the Stiefel (or Grassmann) manifold (i.e., the "constrained surface") and an arbitrary cost function.

The SD algorithms proposed in this paper operate on the Lie group of unitary matrices $U(n)$. We have derived the Riemannian gradient needed in the optimization on $U(n)$. We choose to follow a geodesic motion. This is justified by the desirable property of $U(n)$ that the right multiplication is an isometry with respect to the canonical bi-invariant metric [35]. This allows us to translate the descent direction at any point in the group to the identity element and exploit the fact that the tangent space at identity is the Lie algebra of skew-Hermitian matrices. This leads to lower computational complexity because the argument of the matrix exponential operation is skew-Hermitian. Novel methods for computing the matrix exponential operation for skew-symmetric matrices recently proposed in [30] and [32] may be exploited. Moreover, we show that using an adaptive step size according to Armijo's rule [1] fits very well to the proposed algorithms.

B. Illustrative Example

We present a rather simple simulation example, in order to illustrate how different algorithms operate under the unitary constraint. We consider the Lie group of unit-norm complex numbers $U(1)$, which are the 1×1 unitary matrices. The unitary constraint is in this case the unit circle. We minimize the cost function $\mathcal{J}(w) = |w+0.2|^2$, subject to $ww^* = 1$. Five different algorithms are considered. The first one is the unconstrained SD algorithm on the Euclidean space, with the corresponding update $w_{k+1} = w_k - \eta(w_k + 0.2)$, where η is the step size. The

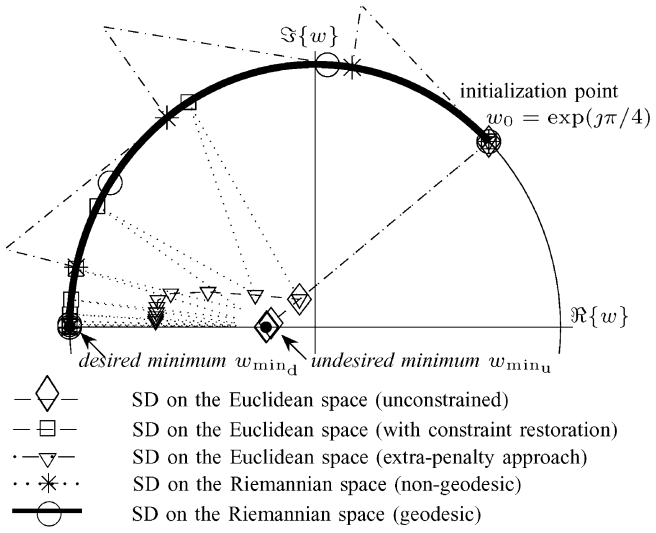

Fig. 1. Minimization of a cost function on the unit circle $U(1)$. Euclidean versus Riemannian SD methods.

second one is the same SD with enforcing the unit norm constraint, $w_{k+1} = w_{k+1}/|w_{k+1}|$, after each iteration. The third method is similar to the bigradient method [16] derived from the Lagrange multiplier method. An extra-penalty $(|w|^2 - 1)^2$ weighted by a parameter λ is added to the original cost function $\mathcal{J}(w)$ in order to penalize the deviation from the unit norm. In this case, $w_{k+1} = w_k - \eta[(w_k + 0.2) + \lambda w_k(|w_k|^2 - 1)]$. The fourth SD algorithm operates on the right parameter space determined by the constraint. At each point the algorithm takes a direction tangent to the unit circle and the resulting point is projected back to the unit circle. The corresponding update is $w_{k+1} = \boldsymbol{\pi}(w_k + \gamma[w_k(|w_k|^2 - 1) + 0.2(w_k^2 - 1)])$, where γ is the step size and $\boldsymbol{\pi}(\cdot)$ is the operator which projects an arbitrary point to the closest point on the unit circle in terms of Euclidean norm. The fifth algorithm is a *multiplicative update* SD algorithm derived in this paper. The corresponding update is a rotation, i.e., $w_{k+1} = w_k \exp[0.04\mu j \Im\{w_k\}]$, where $j = \sqrt{-1}$ and $\Im\{w_k\}$ is the imaginary part of w_k. The parameter μ represents the step size. The starting point is $w_0 = \exp(j\pi/4)$ for all the algorithms (see Fig. 1). The point $w_{\min_u} = -0.2$, sets the cost function to its minimum $\mathcal{J}(w_{\min_u}) = 0$, but this is an undesired minimum because it does not satisfy the constraint. The desired optimum is $w_{\min_d} = -1$, where the constraint is satisfied and $\mathcal{J}(w_{\min_d}) = 0.64$. We may notice in Fig. 1 that the unconstrained SD (marked by $\Diamond$) takes the SD direction in $\mathbb{R}^2$, and goes straight to the undesired minimum. By enforcing the unit norm constraint, we project radially the current point on the unit circle ($\Box$). The enforcing is necessary at every iteration in order to avoid the undesired minimum. The extra-penalty SD algorithm ($\bigtriangledown$) follows the unconstrained SD in the first iteration, since initially the extra-penalty term is equal to zero. It converges somewhere between the desired and the undesired minimum.

The SD algorithm [2] on the space determined by the constraint ($*$) takes in this case *the SD direction on* $U(1)$, tangent to the unit circle. The resulting point is projected to the closest point on the unit circle in terms of Euclidean norm. The proposed SD algorithm ($\bigcirc$) uses a multiplicative update which is a phase rotation. The phase is proportional to the imaginary part of the complex number associated with the point. For this reason, the constraint is satisfied at every step in a natural way. Although this low-dimensional example is rather trivial, it has been included for illustrative purposes. In the case of multidimensional unitary matrices, a similar behavior is encountered.

III. Algorithm Derivation

In this section, we derive two generic SD algorithms on the Lie group of unitary matrices. Consider a real-valued cost function $\mathcal{J}$ of a complex $n \times n$ matrix $\mathbf{W}$, i.e., $\mathcal{J} : \mathbb{C}^{n\times n} \to \mathbb{R}$. Our goal is to minimize (or maximize) the function $\mathcal{J} = \mathcal{J}(\mathbf{W})$ under the constraint that $\mathbf{W}\mathbf{W}^H = \mathbf{W}^H\mathbf{W} = \mathbf{I}$, i.e., $\mathbf{W}$ is unitary. We proceed as follows. First, in Section III-A we describe the Lie group $U(n)$ of $n \times n$ unitary matrices, which is a *real* differentiable manifold. Moreover, we describe the real differentiation of functions defined in complex spaces in a way which is suitable for the optimization. In Section III-B, we introduce the Riemannian metric on the Lie group $U(n)$. The definition of the gradient on the Riemannian space is intimately related to this metric. The Riemannian gradient is derived in Section III-C, and a basic generic optimization algorithm is given in Section III-D. Finally, a Riemannian SD algorithm with an adaptive step size is given in Section III-E.

A. Differentiation of Functions Defined in Complex Spaces

A Lie group is defined to be a differentiable manifold with a smooth, i.e., differentiable group structure [36]. The Lie group of unitary matrices $U(n)$ is a *real* differentiable manifold because it is endowed with a real differentiable structure. Therefore, we deal with a real-valued cost function essentially defined in a real parameter space. However, since the algebraic properties of the group are defined in terms of the complex field, it is convenient to operate directly with complex representation of the matrices instead of using separately their real and the imaginary parts, i.e., without using reals for representing the complex space [43], [44]. Now the real differentiation can be described by a pair of complex-valued operators defined in terms of real differentials with respect to real and imaginary parts [43], [45]

$$\frac{\partial \mathcal{J}}{\partial \mathbf{A}} \triangleq \frac{1}{2}\left(\frac{\partial \mathcal{J}}{\partial \mathbf{A}_{\mathcal{R}}} - \jmath \frac{\partial \mathcal{J}}{\partial \mathbf{A}_{\mathcal{I}}}\right)$$

and

$$\frac{\partial \mathcal{J}}{\partial \mathbf{A}^*} \triangleq \frac{1}{2}\left(\frac{\partial \mathcal{J}}{\partial \mathbf{A}_{\mathcal{R}}} + \jmath \frac{\partial \mathcal{J}}{\partial \mathbf{A}_{\mathcal{I}}}\right) \tag{1}$$

with $\mathbf{A}_{\mathcal{R}} \triangleq \Re\{\mathbf{A}\}$ and $\mathbf{A}_{\mathcal{I}} \triangleq \Im\{\mathbf{A}\}$. If a function $\mathcal{J}$ is holomorphic (analytical), the first differential operator in (1) coincides with the complex differential and the second one is identically zero (Cauchy-Riemann equations). It should be noted that a real-valued function is holomorphic only if it is a constant. Therefore, the complex analyticity is irrelevant to optimization problems. The above representation is more compact, allows differentiation of complex argument functions without

using reals for the representation, and it is appropriate for many applications [45].

B. Riemannian Structure on $U(n)$

A differentiable function $\boldsymbol{\mathcal{C}} : (-\epsilon, \epsilon) \rightarrow U(n)$ represents a *curve* on the smooth manifold $U(n)$ (see Fig. 2). Let $\boldsymbol{\mathcal{C}}(\mathbf{0}) = \mathbf{W}$ and let $\mathcal{D}$ be the set of functions on $U(n)$ that are differentiable at $\mathbf{W}$. The *tangent vector to the curve* $\boldsymbol{\mathcal{C}}$ at $t = 0$ is a function $\mathbf{X} = \boldsymbol{\mathcal{C}}'(0) : \mathcal{D} \rightarrow \mathbb{R}$ given by

$$\mathbf{X}(\mathcal{J}) = \left.\frac{d\mathcal{J}(\boldsymbol{\mathcal{C}}(t))}{dt}\right|_{t=0}, \qquad \mathcal{J} \in \mathcal{D}. \tag{2}$$

A *tangent vector* $\mathbf{X}$ at $\mathbf{W}$ is the tangent vector at $t = 0$ of some curve $\boldsymbol{\mathcal{C}}$ with $\boldsymbol{\mathcal{C}}(\mathbf{0}) = \mathbf{W}$. All the tangent vectors at a point $\mathbf{W} \in U(n)$ form the *tangent space* $T_{\mathbf{W}}U(n)$. The tangent space is a *real* vector space attached to every point in the differential manifold. It should be noted that the value $\mathbf{X}(\mathcal{J}) \in \mathbb{R}$ is independent of the choice of local coordinates (*chart*) and independent of the curve as long as $\boldsymbol{\mathcal{C}}(\mathbf{0}) = \mathbf{W}$ and $\boldsymbol{\mathcal{C}}'(0) = \mathbf{X}$ [35]. Since the curve $\boldsymbol{\mathcal{C}} \subset U(n)$, we have $\boldsymbol{\mathcal{C}}(t)^H\boldsymbol{\mathcal{C}}(t) = \mathbf{I}$. Differentiating both sides with respect to t, the tangent space at $\mathbf{W} \in U(n)$ may be identified with the n^2-dimensional real vector space (i.e., it is a vector space isomorphic to)

$$T_{\mathbf{W}}U(n) = \{\mathbf{X} \in \mathbb{C}^{n\times n} | \mathbf{X}^H\mathbf{W} + \mathbf{W}^H\mathbf{X} = \mathbf{0}\}. \tag{3}$$

From (3), it follows that the tangent space of $U(n)$ at the group identity is the *real Lie algebra* of skew-Hermitian matrices $\mathfrak{u}(n) \triangleq T_{\mathbf{I}}U(n) = \{\mathbf{S} \in \mathbb{C}^{n\times n} | \mathbf{S}^H + \mathbf{S} = \mathbf{0}\}$. We emphasize that $\mathfrak{u}(n)$ is not a complex vector space (Lie algebra), because the skew-Hermitian matrices are not closed under multiplication with complex scalars. For example, if $\mathbf{S}$ is a skew-Hermitian matrix, then $\jmath\mathbf{S}$ is Hermitian. Let $\mathbf{X}$ and $\mathbf{Y}$ be two tangent vectors, i.e., $\mathbf{X}, \mathbf{Y} \in T_{\mathbf{W}}U(n)$. The inner product in $T_{\mathbf{W}}U(n)$ is given by

$$\langle \mathbf{X}, \mathbf{Y} \rangle_{\mathbf{W}} = \frac{1}{2}\Re\{\mathrm{trace}\{\mathbf{X}\mathbf{Y}^H\}\}. \tag{4}$$

This inner product induces a *bi-invariant Riemannian metric* on the Lie group [35]. We may define the normal space at $\mathbf{W}$ considering that $U(n)$ is embedded in the ambient space $\mathcal{A} = \mathbb{R}^{2n\times 2n}$, equipped with the Euclidean metric. The normal space $N_{\mathbf{W}}U(n)$ is the orthogonal complement of the tangent space $T_{\mathbf{W}}U(n)$ with respect to the metric of the ambient space [24], i.e., for any $\mathbf{X} \in T_{\mathbf{W}}U(n)$ and $\mathbf{N} \in N_{\mathbf{W}}U(n)$, we have $\langle \mathbf{X}, \mathbf{N} \rangle_{\mathcal{A}} \triangleq \Re\{\mathrm{trace}\{\mathbf{X}\mathbf{N}^H\}\} = 0$. It follows that the normal space at $\mathbf{W} \in U(n)$ is given as

$$N_{\mathbf{W}}U(n) = \{\mathbf{W}\mathbf{H} \,|\, \mathbf{H} = \mathbf{H}^H, \mathbf{H} \in \mathbb{C}^{n\times n}\}. \tag{5}$$

C. The SD Direction on the Riemannian Space

We consider a differentiable cost function $\mathcal{J} : U(n) \rightarrow \mathbb{R}$. Intuitively, the SD direction is defined as "the direction where the cost function decreases the fastest per unit length." Having the Riemannian metric, we are now able to derive the Riemannian gradient we are interested in. A tangent vector

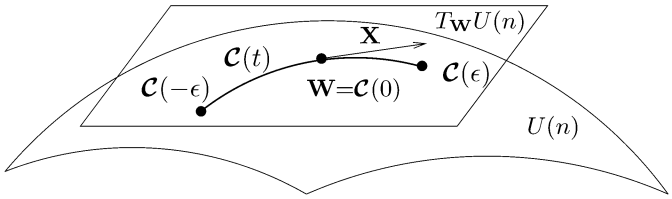

Fig. 2. Illustrative example representing the tangent space $T_{\mathbf{W}}U(n)$ at point $\mathbf{W}$, and a tangent vector $\mathbf{X} \in T_{\mathbf{W}}U(n)$.

$\tilde{\nabla}\mathcal{J} = \tilde{\nabla}\mathcal{J}(\mathbf{W}) \in T_{\mathbf{W}}U(n)$ satisfying for all $\mathbf{X} \in T_{\mathbf{W}}U(n)$ the condition

$$\langle \boldsymbol{\Gamma}_{\mathbf{W}}, \mathbf{X} \rangle_{\mathcal{A}} = \langle \tilde{\nabla}\mathcal{J}, \mathbf{X} \rangle_{\mathbf{W}} \tag{6}$$

is the gradient on $U(n)$ evaluated at $\mathbf{W}$. The direction $\boldsymbol{\Gamma}_{\mathbf{W}} = (\partial\mathcal{J}/\partial\mathbf{W}^*)(\mathbf{W})$ defined in (1) represents the steepest ascent direction of the cost function $\mathcal{J}$ of complex argument $\mathbf{W}$ on the Euclidean space at a given $\mathbf{W}$ [45]. The left-hand side (LHS) in (6) represents an inner product in the ambient space, whereas the right-hand side (RHS) represents a Riemannian inner product at $\mathbf{W}$. Equation (6) may be written as

$$\langle \boldsymbol{\Gamma}_{\mathbf{W}} - \frac{1}{2}\tilde{\nabla}\mathcal{J}, \mathbf{X} \rangle_{\mathcal{A}} = 0, \qquad \mathbf{X} \in T_{\mathbf{W}}U(n). \tag{7}$$

Equation (7) shows that the difference $(\boldsymbol{\Gamma}_{\mathbf{W}} - (1/2)\tilde{\nabla}\mathcal{J})$, is orthogonal to all $\mathbf{X} \in T_{\mathbf{W}}U(n)$. Therefore, it lies in the normal space $N_{\mathbf{W}}U(n)$ (5), i.e.,

$$\boldsymbol{\Gamma}_{\mathbf{W}} - \frac{1}{2}\tilde{\nabla}\mathcal{J} = \mathbf{W}\mathbf{H}, \qquad \mathbf{H} = \mathbf{H}^H \tag{8}$$

where the matrix $\mathbf{H}$ is a Hermitian matrix determined by imposing the condition that $\tilde{\nabla}\mathcal{J} \in T_{\mathbf{W}}U(n)$. From (3) it follows that:

$$(\tilde{\nabla}\mathcal{J})^H\mathbf{W} + \mathbf{W}^H(\tilde{\nabla}\mathcal{J}) = \mathbf{0}. \tag{9}$$

From (8) and (9), we get the expression for the Hermitian matrix $\mathbf{H} = (1/2)[\mathbf{W}^H\boldsymbol{\Gamma}_{\mathbf{W}} + \boldsymbol{\Gamma}_{\mathbf{W}}^H\mathbf{W}]$. The gradient of the cost function on the Lie group of unitary matrices at $\mathbf{W}$ may be written by using (8) as follows:

$$\tilde{\nabla}\mathcal{J}(\mathbf{W}) = \boldsymbol{\Gamma}_{\mathbf{W}} - \mathbf{W}\boldsymbol{\Gamma}_{\mathbf{W}}^H\mathbf{W} \quad \in T_{\mathbf{W}}U(n). \tag{10}$$

D. Moving Towards the SD Direction in $U(n)$

Here, we introduce a generic Riemannian SD algorithm along geodesics on the Lie group of unitary matrices $U(n)$. A *geodesic* curve on a Riemannian manifold is defined as a curve $\boldsymbol{\mathcal{G}}(t)$ for which the second derivative $\boldsymbol{\mathcal{G}}''(t)$ is zero or it lies in the normal space for all t (i.e., the acceleration vector stays normal to the direction of motion as long as the curve is traced with constant speed). Locally the geodesics minimize the path length with respect to the Riemannian metric ([35, p. 67]). A geodesic emanating from the identity $\mathbf{I}$ with a velocity $\mathbf{S}$ is characterized by the exponential map:

$$\boldsymbol{\mathcal{G}}_{\mathbf{I}}(t) = \exp(t\mathbf{S}), \qquad \mathbf{S} \in \mathfrak{u}(n). \tag{11}$$

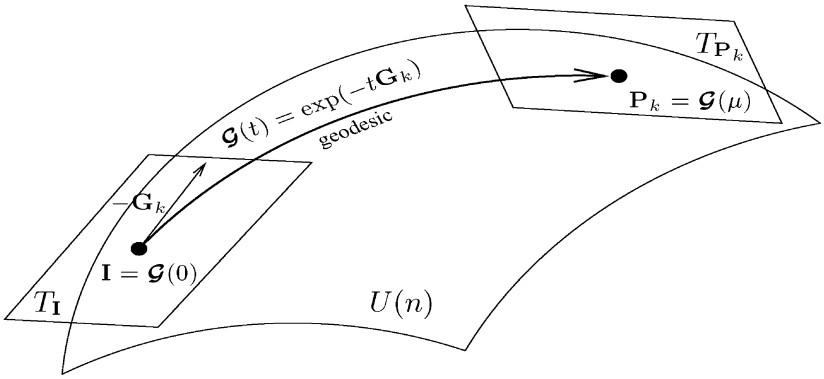

Fig. 3. The geodesic emanating from identity in the direction of $-\mathbf{G}_k$ ending at $\mathbf{P}_k = \exp(-\mu\mathbf{G}_k)$.

The exponential of $n \times n$ complex matrices is given by the convergent power series $\exp(\mathbf{A}) = \sum_{m=0}^{\infty}(\mathbf{A}^m/m!)$. The optimization of $\mathcal{J}(\mathbf{W})$ is carried out along geodesics on the constraint surface. For the optimization we need the equation of the geodesic emanating from $\mathbf{W} \in U(n)$. This may be found by taking into account the fact that the *right translation* in $U(n)$ is an isometry with respect to the metric given by (4) and an isometry maps geodesics to geodesics [10], [35]. Therefore, $\mathcal{G}_{\mathbf{I}}(t) = \mathcal{G}_{\mathbf{W}}(t)\mathbf{W}^H$. It follows that the geodesic emanating from $\mathbf{W}$ is $\mathcal{G}_{\mathbf{W}}(t) = \mathcal{G}_{\mathbf{I}}(t)\mathbf{W}$, i.e.,

$$\mathcal{G}_{\mathbf{W}}(t) = \exp(t\mathbf{S})\mathbf{W}. \tag{12}$$

Consequently, we need to translate the gradient of the cost function at $\mathbf{W}$ (10) to identity, i.e., into the Lie algebra $\mathfrak{u}(n)$. Since the differential of the right translation is a vector space isomorphism, this is performed simply by postmultiplying $\tilde{\nabla}\mathcal{J}(\mathbf{W}) \in T_{\mathbf{W}}U(n)$ by $\mathbf{W}^H$, i.e.,

$$\mathbf{G}(\mathbf{W}) \triangleq \tilde{\nabla}\mathcal{J}(\mathbf{W})\mathbf{W}^H = \boldsymbol{\Gamma}_{\mathbf{W}}\mathbf{W}^H - \mathbf{W}\boldsymbol{\Gamma}_{\mathbf{W}}^H \in \mathfrak{u}(n). \tag{13}$$

We have to keep in mind that this is not the Riemannian gradient of the cost function evaluated at identity. The tangent vector $\mathbf{G}(\mathbf{W})$ is the Riemannian gradient of the cost function evaluated at $\mathbf{W}$ and translated to identity. Note that the argument of the matrix exponential operation is skew-Hermitian. We exploit this very important property later in this paper in order to reduce the computational complexity.

The cost function $\mathcal{J}(\mathbf{W})$ may be minimized iteratively by using a geodesic motion. Typically we start at $\mathbf{W}_0 = \mathbf{I} \in U(n)$. We choose the direction in $\mathfrak{u}(n)$ to be the negative direction of the gradient, i.e., $-\mathbf{G}_k = -\mathbf{G}(\mathbf{W}_k)$ (13). Moving from $\mathbf{W}_k = \mathbf{I}\mathbf{W}_k$ to $\mathbf{W}_{k+1} = \mathbf{P}_k\mathbf{W}_k$, is equivalent to moving from $\mathbf{I}$ to $\mathbf{P}_k$, as it is shown in Fig. 3. The geodesic motion in $U(n)$ corresponds to the multiplication by a rotation matrix $\mathbf{P}_k = \exp(-\mu\mathbf{G}_k)$. The parameter $\mu > 0$ controls the magnitude of the tangent vector and consequently the algorithm convergence speed. The update corresponding to the SD algorithm along geodesics on $U(n)$ is given by

$$\mathbf{W}_{k+1} = \exp(-\mu\mathbf{G}_k)\mathbf{W}_k = \mathbf{P}_k\mathbf{W}_k. \tag{14}$$

TABLE I
THE BASIC RIEMANNIAN SD ALGORITHM ON $U(n)$

1	Initialization: $k = 0$ and $\mathbf{W}_k = \mathbf{I}$
2	Compute the gradient of the cost function on the Euclidean space: $\boldsymbol{\Gamma}_k = \frac{\partial \mathcal{J}}{\partial \mathbf{W}^*}(\mathbf{W}_k)$
3	Compute the gradient direction on the Riemannian space: $\mathbf{G}_k = \boldsymbol{\Gamma}_k\mathbf{W}_k^H - \mathbf{W}_k\boldsymbol{\Gamma}_k^H$
4	Determine the rotation matrix: $\mathbf{P}_k = \exp(-\mu\mathbf{G}_k)$
5	Update: $\mathbf{W}_{k+1} = \mathbf{P}_k\mathbf{W}_k$, $k := k+1$. Iterate the steps **2-5** until convergence.

The algorithm is summarized in Table I. Practical algorithms require the computation of the exponential map, which is addressed in Section V.

E. A Self-Tuning Riemannian SD Algorithm on $U(n)$

An optimal value of the step size μ is difficult to determine in practice. Moreover, it is cost function dependent and the appropriate step size may change at each iteration. The SD algorithm with a fixed small step size converges in general *close to a local minimum*. It trades off between high convergence speed, which requires large step size, and low steady-state error, which requires a small step size. An adaptive step size is often a desirable choice.

In [27], a projection algorithm is considered together with three other optimization alternatives along geodesics in the real case, i.e., on the orthogonal group. The first geodesic algorithm in [27] uses a fixed step size, which leads to the "real-valued counterpart" of the algorithm in Table I. The second one is a geodesic search for computing the step size in the update equation. If the geodesic search is performed in a continuous domain [39], [40], it is computationally very expensive since it involves differential equations. A discretized version of the geodesic search may be employed. Two such methods are reviewed in [27]. The third alternative is a stochastic type of algorithm which adds perturbation to the search direction.

We opt for a fourth alternative based on the Armijo step size [1]. It allows reducing the computational complexity and gives the optimal local performance. This type of algorithm takes an initial step along the geodesic. Then, two other possibilities are checked by evaluating the cost function for the case of doubling or halving the step size. The doubling or halving step continues

TABLE II
THE SELF-TUNING RIEMANNIAN SD ALGORITHM ON $U(n)$

1	Initialization: $k=0$,$\mathbf{W}_k=\mathbf{I}$ and $\mu=1$
2	Compute the gradient of the cost function on the Euclidean space: $\mathbf{\Gamma}_k=\frac{\partial \mathcal{J}}{\partial \mathbf{W}^*}(\mathbf{W}_k)$
3	Compute the gradient direction on the Riemannian space: $\mathbf{G}_k=\mathbf{\Gamma}_k\mathbf{W}_k^H-\mathbf{W}_k\mathbf{\Gamma}_k^H$
4	Evaluate $\langle \mathbf{G}_k,\mathbf{G}_k\rangle_{\mathbf{I}}=\frac{1}{2}\Re\{\mathrm{trace}\{\mathbf{G}_k\mathbf{G}_k^H\}\}$. If it is sufficiently small, then STOP.
5	Determine the rotation matrices: $\mathbf{P}_k=\exp(-\mu\mathbf{G}_k),\quad \mathbf{Q}_k=\mathbf{P}_k\mathbf{P}_k$
6	While $\mathcal{J}(\mathbf{W}_k)-\mathcal{J}(\mathbf{Q}_k\mathbf{W}_k)\geq \mu\langle \mathbf{G}_k,\mathbf{G}_k\rangle_{\mathbf{I}}$ $\mathbf{P}_k:=\mathbf{Q}_k,\quad \mathbf{Q}_k=\mathbf{P}_k\mathbf{P}_k,\quad \mu:=2\mu$
7	While $\mathcal{J}(\mathbf{W}_k)-\mathcal{J}(\mathbf{P}_k\mathbf{W}_k)<(\mu/2)\langle \mathbf{G}_k,\mathbf{G}_k\rangle_{\mathbf{I}}$ $\mathbf{P}_k=\exp(-\mu\mathbf{G}_k),\ \mu:=\mu/2$
8	Update: $\mathbf{W}_{k+1}$=$\mathbf{P}_k\mathbf{W}_k$, k:=k+1, and go to step **2**.

as long as the step size is out of a range whose limits are set by two inequalities. It is known that in a stationary scenario (i.e., the matrices involved in the cost function are time invariant) the SD algorithm together with the Armijo step size rule [1] almost always converges *to a local minimum* if not initialized at a stationary point. The convergence properties of the geodesic SD algorithm using the Armijo rule have been established in [29], [46] for general Riemannian manifolds, provided that the cost function is continuously differentiable and has bounded level sets. The first condition is an underlying assumption in this paper and the second one is ensured by the compactness of $U(n)$.

In [2], a SD algorithm is coupled with the Armijo rule for optimizing the step size. Geodesic motion is not used. Nevertheless, in the general framework proposed in [2] it could be used. We show that by using the Armijo rule together with the generic SD algorithm along geodesics, the computational complexity is reduced by exploiting the properties of the exponential map, as it will be shown later.

The generic SD algorithm with adaptive step size selection is summarized in Table II. The choice for computing the matrix exponential is explained in Section V.

Algorithm Description: The algorithm consists of the following steps.

- Step 1—Initialization: A typical initial value is $\mathbf{W}_0=\mathbf{I}$. If the gradient $\mathbf{G}_0=\mathbf{0}$, then the identity element is a stationary point. In that case a different initial value $\mathbf{W}_0\in U(n)$ may be chosen.
- Steps 2–3—Gradient computation: The Euclidean gradient and Riemannian gradient are computed.
- Step 4—Setting the threshold for the final error: Evaluate the squared norm of the Riemannian gradient $\langle \mathbf{G}_k,\mathbf{G}_k\rangle_{\mathbf{I}}$ in order to check if we are sufficiently close to the minimum of the cost function. The residual error may be set to a value closest to the smallest value available in the limited-precision environment, or the highest value which can be tolerated in task at hand.
- Step 5—Rotation matrix computation: This step requires the computation of the rotation matrix $\mathbf{P}_k=\exp(-\mu\mathbf{G}_k)$. The rotation matrix $\mathbf{Q}_k=\exp(-2\mu\mathbf{G}_k)$ may be computed just by squaring $\mathbf{P}_k$, because $\exp(-2\mu\mathbf{G}_k)=[\exp(-\mu\mathbf{G}_k)]^2$. Therefore, when doubling the step size, instead of computing a new matrix exponential only a matrix squaring operation is needed. It is important to mention that the squaring operation is a very stable operation [32] being also used in software packages for computing the matrix exponential.
- Steps 6 and 7—Step size evaluation: In every iteration k we check if the step size is in the appropriate range determined by the two inequalities. The step size evolves in a dyadic basis. If it is too small it will be doubled and if it is too high it will be halved.
- Step 8—Update: The new update is obtained in a multiplicative manner and a new iteration is started with step 2 if the residual error is not sufficiently small.

Remark: The SD algorithm in Table II may be easily converted into a steepest ascent algorithm. The only difference is that the step size μ would be negative and the inequalities in steps 6 and 7 need to be reversed.

IV. RELATIONS AMONG DIFFERENT LOCAL PARAMETRIZATIONS ON THE UNITARY GROUP $U(n)$

The proposed SD algorithms search for the minimum by moving along geodesics, i.e., the local parametrization is the exponential map. Other local parametrizations used to describe a small neighborhood of a point in the group have been proposed in [2] and [11]. In this section, we establish equivalence relationships among some different local parametrizations of the unitary group $U(n)$. The first one is the exponential map used in the proposed SD algorithms, the second one is the Cayley transform [31], the third one is the Euclidean projection operator [2], and the fourth one is a parametrization based on the QR-decomposition. The four parametrizations lead to different update rules for the basic SD algorithm on $U(n)$. The update expressions may be described in terms of Taylor series expansion, and we prove the equivalence among all of them up to a certain approximation order.

A. The Exponential Map

The rotation matrix $\mathbf{P}_k$ used in the update expression (14) of the proposed algorithm can be expressed as a Taylor series expansion of the matrix exponential, i.e., $\mathbf{P}_k=\mathbf{I}-\mu\mathbf{G}_k+(\mu^2/2)\mathbf{G}_k^2+(\mu^3/6)\mathbf{G}_k^3+\cdots$. The update $\mathbf{W}_{k+1}=\mathbf{P}_k\mathbf{W}_k$ is equivalent to

$$\mathbf{W}_{k+1}=\mathbf{W}_k-\mu\mathbf{G}_k\mathbf{W}_k+\frac{\mu^2}{2}\mathbf{G}_k^2\mathbf{W}_k-\frac{\mu^3}{6}\mathbf{G}_k^3\mathbf{W}_k+\ldots. \tag{15}$$

B. The Cayley Transform

If the rotation matrix $\mathbf{P}_k$ in the update is computed by using the CT [31] instead of the matrix exponential, then $\mathbf{P}_k=(\mathbf{I}-t\mathbf{G}_k)^{-1}(\mathbf{I}+t\mathbf{G}_k)$. The corresponding Taylor series is $\mathbf{P}_k=\mathbf{I}+2\sum_{m=1}^{\infty}(t\mathbf{G}_k)^m$. For $t=-\mu/2$ the update equation is

$$\mathbf{W}_{k+1}=\mathbf{W}_k-\mu\mathbf{G}_k\mathbf{W}_k+\frac{\mu^2}{2}\mathbf{G}_k^2\mathbf{W}_k-\frac{\mu^3}{4}\mathbf{G}_k^3\mathbf{W}_k+\ldots. \tag{16}$$

Obviously, (16) is equivalent to (15) up to the second order. Notice also that for $t=-\mu/2$ the CT equals the first order

diagonal Padé approximation of $\exp(-\mu\mathbf{G}_k)$ (i.e., $p = q = 1$, see [32]).

C. The Euclidean Projection Map

Another possibility is to use an Euclidean projection map as a local parametrization as in [2]. This map projects an arbitrary matrix $\mathbf{X} \in \mathbb{C}^{n\times n}$ onto $U(n)$ at a point $\mathbf{W}$ which is the closest point to $\mathbf{X}$ in terms of Euclidean norm, i.e., $\boldsymbol{\pi}(\mathbf{X}) = \arg\min_{\mathbf{W}} ||\mathbf{X} - \mathbf{W}||, \mathbf{W} \in U(n)$. The unitary matrix $\mathbf{W}_{\text{opt}}$ minimizing the above norm can be obtained from the polar decomposition of $\mathbf{X}$ as in [47]

$$\mathbf{W}_{\text{opt}} = \boldsymbol{\pi}(\mathbf{X}) = \mathbf{U}_{\mathbf{X}}\mathbf{V}_{\mathbf{X}}^H \tag{17}$$

where $\mathbf{U}_{\mathbf{X}}$ and $\mathbf{V}_{\mathbf{X}}$ are the left and the right singular vectors of $\mathbf{X}$, respectively. Equivalently

$$\mathbf{W}_{\text{opt}} = \boldsymbol{\pi}(\mathbf{X}) = \mathbf{X}(\mathbf{X}^H\mathbf{X})^{-1/2}. \tag{18}$$

Equation (18) is also known as "the symmetric orthogonalization" procedure [12]. In [2], the local parametrizations are more general in the sense that they are chosen for the Stiefel and Grassmann manifolds. The projection operation is computed via SVD as in (17). For the SD algorithm on the Stiefel manifold [2], the update is of form $\mathbf{W}_{k+1} = \boldsymbol{\pi}(\mathbf{W}_k + \gamma\mathbf{Z}_k)$, where $\mathbf{Z}_k$ is the SD direction on the manifold and γ is the step size. According to (18) the update is equivalent to $\mathbf{W}_{k+1} = (\mathbf{W}_k + \gamma\mathbf{Z}_k)[(\mathbf{W}_k + \gamma\mathbf{Z}_k)^H(\mathbf{W}_k + \gamma\mathbf{Z}_k)]^{-1/2}$. By expanding the above expression in a Taylor series we get

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \gamma\mathbf{Z}_k - \frac{\gamma^2}{2}\mathbf{W}_k\mathbf{Z}_k^H\mathbf{Z}_k - \frac{\gamma^3}{2}\mathbf{Z}_k\mathbf{Z}_k^H\mathbf{Z}_k + \cdots. \tag{19}$$

Considering that $\mathbf{Z}_k = -\mathbf{G}_k\mathbf{W}_k$ and $\mathbf{G}_k^H = -\mathbf{G}_k$, for $\gamma = \mu$ the three update expressions (15), (16), and (19) become equivalent up to the second order.

D. The Projection Based on the QR-Decomposition

A computationally inexpensive way to approximate the optimal projection of an arbitrary matrix onto $U(n)$ (18) is the QR-decomposition. We show that this is not the optimal projection in terms of minimum Euclidean distance, but is accurate enough to be used in practical applications. We establish a connection between the QR-based projection and the optimal projection. Let us consider the QR-decomposition of the arbitrary nonsingular matrix $\mathbf{X} \in \mathbb{C}^{n\times n}$ given by $\mathbf{X} = \mathbf{Q}_{\mathbf{X}}\mathbf{R}_{\mathbf{X}}$, where $\mathbf{R}_{\mathbf{X}}$ is an upper triangular matrix and $\mathbf{Q}_{\mathbf{X}}$ is a unitary matrix. The unitary matrix $\mathbf{Q}_{\mathbf{X}}$ is an approximation of the optimal projection of $\mathbf{X}$ onto $U(n)$, i.e., $\mathbf{W}_{\text{opt}} \approx \mathbf{Q}_{\mathbf{X}}$. The connection between this projection and the optimal projection can be established by using the polar decomposition of the upper-triangular matrix $\mathbf{R}_{\mathbf{X}} = \boldsymbol{\Phi}_{\mathbf{R}_{\mathbf{X}}}\mathbf{H}_{\mathbf{R}_{\mathbf{X}}}$. We obtain $\mathbf{W}_{\text{opt}} = \mathbf{Q}_{\mathbf{X}}\boldsymbol{\Phi}_{\mathbf{R}_{\mathbf{X}}}$, where $\boldsymbol{\Phi}_{\mathbf{R}_{\mathbf{X}}} = \mathbf{R}_{\mathbf{X}}(\mathbf{R}_{\mathbf{X}}^H\mathbf{R}_{\mathbf{X}})^{-1/2}$ and $\mathbf{H}_{\mathbf{R}_{\mathbf{X}}} = \mathbf{H}_{\mathbf{R}_{\mathbf{X}}}^H$. Therefore, the matrix $\mathbf{Q}_{\mathbf{X}}$ is an approximation of the optimal projection $\mathbf{W}_{\text{opt}}$ and it includes an additional rotation $\boldsymbol{\Phi}_{\mathbf{R}_{\mathbf{X}}}^H$ from $\mathbf{W}_{\text{opt}}$. The update of the SD algorithm is equal to the unitary factor from the QR-decomposition. In other words, if we have $\mathbf{W}_k + \gamma\mathbf{Z}_k = \mathbf{Q}_{\mathbf{W}_k+\gamma\mathbf{Z}_k}\mathbf{R}_{\mathbf{W}_k+\gamma\mathbf{Z}_k}$, then

$$\mathbf{W}_{k+1} = \mathbf{Q}_{\mathbf{W}_k+\gamma\mathbf{Z}_k}. \tag{20}$$

The equivalence up to the first order between the update (20) and the other update expressions (15), (16), and (19) is obtained by expanding the columns of the matrix $\mathbf{Q}_{\mathbf{W}_k+\gamma\mathbf{Z}_k}$ from the Gram-Schmidt process separately in a Taylor series (proof available on request)

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \gamma\mathbf{G}_k\mathbf{W}_k + \ldots. \tag{21}$$

The equivalence may extend to higher orders, but this remains to be studied.

V. Computational Complexity

In this section, we evaluate the computational complexity of the SD algorithms on $U(n)$ by considering separately the *the geodesic* and *the nongeodesic* SD algorithms. The proposed geodesic SD algorithms use the rotational update of form $\mathbf{W}_{k+1} = \mathbf{P}_k\mathbf{W}_k$, and the rotation matrix $\mathbf{P}_k$ is computed via matrix exponential. We review the variety of algorithms available in the literature for calculating the matrix exponential in the proposed algorithms. Details are given for the matrix exponential algorithms with the most appealing properties. The nongeodesic SD algorithms are based on an update expression of form $\mathbf{W}_{k+1} = \boldsymbol{\pi}(\mathbf{W}_k + \gamma\mathbf{Z}_k)$, and the computational complexity of different cases is described. The cost of adapting to the step size using the Armijo rule is also evaluated. The SD method on $U(n)$ involves overall complexity of $\mathcal{O}(n^3)$ flops[1] per iteration. Algorithms like conjugate gradient or Newton algorithm are expected to provide a faster convergence, but also their complexity is expected to be higher. Moreover, a Newton algorithm is more likely to converge to stationary points other than local minima.

A. Geodesic SD Algorithms on $U(n)$

In general, the geodesic motion on manifolds is computationally expensive. In the case of $U(n)$, the complexity is reduced even though it requires the computation of the matrix exponential. We are interested in the special case of matrix exponential of form $\mathbf{W} = \exp(t\mathbf{G})$, where $t \in \mathbb{R}$ and $\mathbf{G}$ is a skew-Hermitian matrix. Obviously, finding the exponential of a skew-Hermitian matrix has lower complexity than of general matrix. The matrix exponential operation maps the skew-Hermitian matrices from the Lie algebra $\mathfrak{u}(n)$ into unitary matrices which reside on the Lie group $U(n)$. Several alternatives for approximating the matrix exponential have been proposed in [30], [32], [48], and [49]. In general the term "approximation" may refer to two different things. The first kind of approximation maps the elements of the Lie algebra exactly into the Lie group, and the approximation takes place only in terms of deviation "within the constrained surface." Among the most efficient methods from this category are: the diagonal Padé approximation [32], Generalized Polar Decomposition [30], [48], technique of coordinates of the second kind [49]. The second category includes methods for which the resulting elements do not reside on the group anymore, i.e., $\mathbf{W}^H\mathbf{W} \approx \mathbf{I}$. The most popular methods belonging to

[1]One "flop" is defined as a complex addition or a complex multiplication. An operation of form $ab + c, a, b, c \in \mathbb{C}$ is equivalent to two flops. A simple multiplication of two $n \times n$ matrices requires $2n^3$ flops. This is a quick evaluation of the computational complexity, not necessarily proportional to the computational speed.

this category is the truncated Taylor series and the nondiagonal Padé approximation. They do not preserve the algebraic properties, but they still provide reasonable performance in some applications [50]. Their accuracy may be improved by using them together with the scaling and squaring procedure.

1) Padé Approximation of the Matrix Exponential: This is [32, Method 2], and together with *scaling and squaring* [32, Method 3] is considered to be one of the most efficient methods for approximating a matrix exponential. For normal matrices (i.e., matrices $\mathbf{A}$ which satisfy $\mathbf{A}^H\mathbf{A} = \mathbf{A}\mathbf{A}^H$), the Padé approximation prevents the round-off error accumulation. The skew-Hermitian matrices are normal matrices, therefore, they enjoy this benefit. Because we deal with a SD algorithm on $U(n)$ we are also concerned about preserving the Lie algebraic properties. The *diagonal* Padé approximation preserves the unitarity property accurately. The Padé approximation together with scaling and squaring supposes the choice of the Padé approximation order q and the scaling and squaring exponent j to get the best approximant given the approximation accuracy. See [32] for information of choosing the pair (q, j) optimally. The complexity of this approximation is $2(q + j + (1/3))n^3$ flops. The drawback of Padé method when used together with the scaling and squaring procedure is that if the norm of the argument is large the computational efficiency decreases due to the repeated squaring.

2) Approximation of the Matrix Exponential via Generalized Polar Decomposition (GPD): The GPD method, recently proposed in [30] is consistent with the Lie group structure as it maps the elements of the Lie algebra exactly into the corresponding Lie group. The method lends itself to implementation in parallel architectures and it requires about $5n^3$ flops [30] regardless of the approximation order. It may not be the most efficient implementation in terms of flop count, but the algorithm has potential for highly parallel implementation. GPD algorithms based on splitting techniques have also been proposed in [48]. The corresponding approximation is less complex than the one in [30] for the second and the third order. The second-order approximation requires only $2(2/3)n^3$ flops. This is the same amount of computation needed to perform the CT. Other efficient approximations in a Lie-algebraic setting have been considered in [49] by using the technique of coordinates of the second kind (CSK). A second-order CSK approximant requires $3n^3$ flops.

B. Nongeodesic SD Algorithms on $U(n)$

This category includes local parametrizations derived from a projection operator which is used to map arbitrary matrices into $U(n)$. The optimal projection and an approximation of it are considered.

1) Optimal Projection: The projection that minimizes the Euclidean distance between the arbitrary matrix $\mathbf{X}$ and a matrix $\mathbf{W} \in U(n)$ may be computed in different ways. By using the SVD the computation of the projection requires $23n^3$ flops and by using the procedure (18) it requires about $12n^3$ flops.

2) Approximation of the Optimal Projection: This method is the most inexpensive approximation of the optimal projection, being based on the QR-decomposition of the matrix $\mathbf{X}$. It requires only the unitary matrix $\mathbf{Q}$ which is an orthonormal basis in the range space of $\mathbf{X}$. This can be done by using Householder reflections, Givens rotations or the Gram-Schmidt procedure [32]. The most computationally efficient and numerically stable approach is the *modified Gram-Schmidt procedure* which requires only $2n^3$ flops.

TABLE III
THE COMPLEXITY (IN FLOPS) OF COMPUTING THE LOCAL PARAMETRIZATION IN $U(n)$

geodesic SD			non-geodesic SD	
DPA+SS	GPD-IZ	GPD-ZMK	OP	AOP
$2\frac{2}{3}n^3$	$5n^3$	$2\frac{2}{3}n^3$	$12n^3$	$2n^3$

In Table III, we summarize the complexity[2] of computing the local parametrizations for the geodesic and the nongeodesic methods, respectively. The geodesic methods include: the diagonal Padé approximation with scaling and squaring (DPA + SS) of type (1, 0) [32] (CT), the Generalized Polar Decomposition with reduction to tridiagonal form (GPD-IZ) [30] and without reduction to the tridiagonal form (GPD-ZMK) [48]. All methods have an approximation order of two. The nongeodesic methods include the optimal projection (OP) and its approximation (AOP).

C. The Cost of Using an Adaptive Step Size

In this subsection, we analyze the computational complexity of adapting the step size with Armijo rule. The total computational cost is given by the complexity of computing the local parametrization and the additional complexity of selecting the step size. Therefore, the step size adaptation is a critical aspect to be considered. We consider again the geodesic SD algorithms and the nongeodesic SD algorithms, respectively. We show that the geodesic methods may reduce the complexity of the step size adaptation.

1) The Geodesic SD Algorithms: Since the step size evolves in a dyadic basis, the geodesic methods are very suitable for the Armijo step. This is due to the fact that doubling the step size does not require any expensive computation, just squaring the rotation matrix as in the scaling and squaring procedure. For normal matrices, the computation of the matrix exponential via matrix squaring *prevents the round-off error accumulation* [32]. An Armijo type of geodesic SD algorithm enjoys this benefit, since the argument of the matrix exponential is skew-Hermitian. Moreover, when the step size is halved, the corresponding rotation matrix may be available from the scaling and squaring procedure which is often combined with other methods for approximating the matrix exponential. This allows reducing the complexity because the expensive operation may often be avoided.

2) The Nongeodesic SD Algorithms: The nongeodesic methods compute the update by projecting a matrix into $U(n)$. Unfortunately, the Armijo step size adaptation is relatively expensive in this case. The main reason is that the update $\mathbf{W}_{k+1}(\gamma) = \boldsymbol{\pi}(\mathbf{W}_k + \gamma\mathbf{Z}_k)$ and the one corresponding to the double step size $\mathbf{W}_{k+1}(2\gamma) = \boldsymbol{\pi}(\mathbf{W}_k + 2\gamma\mathbf{Z}_k)$ do not have a straightforward relationship as squaring the rotation matrix for the geodesic methods. Thus, the projection operation needs to be computed multiple times. Moreover, even keeping the step size constant involves the computation of the projection twice

[2]Only dominant terms are reported, i.e., $\mathcal{O}(n^3)$.

TABLE IV
THE COMPLEXITY OF ADAPTING THE STEP SIZE

Step size adjustment	flops/iteration	
	geodesic SD	non-geodesic SD
keep μ constant	$C_{\mathrm{r}} + 3n^3$	$2C_{\mathrm{p}}$
double μ	$C_{\mathrm{r}} + (t_{\mathrm{d}} + 3)n^3$	$(t_{\mathrm{d}} + 1)C_{\mathrm{p}}$
halve μ	$t_{\mathrm{h}}C_{\mathrm{r}} + 3n^3$	$(t_{\mathrm{h}} + 2)C_{\mathrm{p}}$

since both inequalities 6 and 7 in Table II need to be tested even if they fail. In this case both projections $\boldsymbol{\pi}(\mathbf{W}_k + 2\gamma\mathbf{Z}_k)$ and $\boldsymbol{\pi}(\mathbf{W}_k + \gamma\mathbf{Z}_k)$ need to be evaluated.

We compare the proposed geodesic SD algorithms to the nongeodesic SD algorithms by considering the complexity of adapting to the step size. We also take into account the cost C_{r} of computing the rotation matrix for the geodesic SD algorithms and the cost C_{p} of computing the projection operation for the nongeodesic algorithms. These costs are given in Table III for different local parametrizations. We denote by t_{d} and t_{h} the number of times we double, respectively, the number of times we halve the step size during one iteration k. The complexity of adapting the step size is summarized in Table IV.

We may conclude that the local parametrization may be chosen based on the requirements of the application. Often, preserving the algebraic structure is important. On the other hand, the implementation complexity may be a limiting factor. Most of the parametrizations presented here are equivalent up to the second order. Therefore, the difference in convergence speed is not expected to be significant. The cost function to be minimized plays a role in this difference as also stated in [2]. An SD algorithm with adaptive step size is more suitable in practice. Consequently, the geodesic algorithms are a good choice. In this case, the matrix exponential is employed and it may be computed either by using the CT [31] or the GPD-ZMK method [48]. They require equal number of flops, therefore the choice remains upon the numerical stability. Even though the GPD-IZ method recently proposed in [30] is sensibly less efficient in terms of flop count, it may be *faster* in practice if implemented in parallel architectures. Moreover, it provides good numerical stability as it will be seen in the simulations. As a final conclusion, we would opt for the GPD-IZ method [30] if the algorithm is implemented in a parallel fashion and for the CT if the parallel computation is not an option.

VI. NUMERICAL STABILITY

In this section, we focus on the numerical stability of the proposed SD algorithms on $U(n)$. Taking into account the recursive nature of the algorithms, we analyze the deviation of each new update $\mathbf{W}_{k+1}$ from the unitary constraint, i.e., the departure from $U(n)$. The nongeodesic SD algorithms do not experience this problem due to nature of local parametrization. In that case, the error does not accumulate because the projection operator maps the update into the manifold at every new iteration. Therefore, we consider only the geodesic SD algorithms.

The methods proposed here for approximating the matrix exponential map the elements of the Lie algebra exactly into the Lie group, therefore they do not cause deviation from the unitary constraint. However, the rotation matrix $\mathbf{P}_k$ may be affected by round-off errors, and the error may accumulate in the update (14) due to the repeated matrix multiplications.

We provide a closed-form expression for the expected value of the deviation form the unitary constraint after a certain number of iterations. The theoretical value derived here predicts the error accumulation with high accuracy, as it will be shown in the simulations. We show that the error accumulation is negligible in practice.

We assume that at each iteration k, the rotation matrix $\mathbf{P}_k$ is affected additively by the quantization error $\mathbf{E}_k$, i.e., $\tilde{\mathbf{P}}_k = \mathbf{P}_k + \mathbf{E}_k$, where $\mathbf{P}_k = \exp(-\mu\mathbf{G}_k) \in U(n)$ is the true rotation matrix. The real and imaginary parts of the entries of the matrix $\mathbf{E}_k$ are mutually independent and independent of the entry indices. They are assumed to be uniformly distributed within the quantization interval of width δ. The deviation of the quantized update $\tilde{\mathbf{W}}_{k+1}$ from the unitary constraint is measured by

$$\Delta_{k+1} = ||\tilde{\mathbf{W}}_{k+1}^H \tilde{\mathbf{W}}_{k+1} - \mathbf{I}||_F^2. \tag{22}$$

The closed-form expression of the expected value of the deviation at iteration $k+1$ is given by (derivation available on request)

$$E[\Delta_{k+1}] = n\left[1 + \frac{2n}{3}\delta^2 + \frac{(10n+4)}{360}\delta^4\right]^{k+1} - 2n\left[1 + \frac{n}{6}\delta^2\right]^{k+1} + n. \tag{23}$$

The theoretical value (23) depends on the matrix dimension n and the width of the quantization interval δ. Often, the convergence is reached in just few iterations, as in the practical example presented in Section VII. Therefore, the error accumulation problem is avoided. We show that even if the convergence is achieved after a large number of iterations, the expected value of the deviation from the unitary constraint is negligible. This is due to the fact that the dominant term in (23) is driven by the factor $\delta^{2(k+1)}$. The error is increasing very slowly and the increasing rate decays rapidly with k, as it will be shown in Section VII.

VII. SIMULATION RESULTS AND APPLICATIONS

In this section, we test how the proposed method performs in signal processing applications. An example of separating independent signals in a MIMO system is given. Applications to array signal processing, ICA, BSS, for MIMO systems may be found in [5]–[8], [10], [14]–[17], [19]–[23], [51]. A recent review of the applications of differential geometry to signal processing may be found in [52].

A. Blind Source Separation for MIMO Systems

Separating signals blindly in a MIMO communication systems may be done by exploiting the statistical information of the transmitted signals. The JADE algorithm [3] is a reliable alternative for solving this problem. The JADE algorithm consists of two stages. First, a *prewhitening* of the received signal is performed. The second stage is a *unitary rotation*. This second stage is formulated as an optimization problem under unitary matrix constraint, since no closed form solution can be given except for simple cases such as 2-by-2 unitary matrices. This may be efficiently solved by using the proposed SD on the unitary group. It should be noted that the first stage can also be formulated as a unitary optimization problem [50], and the algorithms

proposed in this paper could be used to solve it. However, here we only focus on the second stage.

The JADE approach has been recently considered on the oblique [53] and Stiefel [19] manifolds. The SD algorithm in [19] has complexity of $\mathcal{O}(n^3)$ per iteration as the original JADE [3], but in general it converges in fewer iterations. This is true especially for large matrix dimensions, where JADE seems to converge slowly due to its pairwise processing approach. Therefore, the overall complexity of algorithm in [19] is lower than in the original JADE. It operates on the Stiefel manifold of $n \times n$ unitary matrices, but still without taking into account the additional Lie group structure of the manifold. Our proposed algorithm is designed specifically for the case of $n \times n$ unitary matrices, and for this reason the complexity per iteration is lower compared to the SD in [19]. The convergence speed is identical as it will be shown later. The algorithms in [2] and [19] are more general than the proposed one, in the sense that the parametrization is chosen for the Stiefel and the Grassmann manifolds. The reduction in complexity for the proposed algorithm is achieved by exploiting the additional group structure of $U(n)$. The SD along geodesics is more suitable for Armijo step size.

A number of m independent zero-mean signals are sent by using m transmit antennas and they are received by r receive antennas. The frequency flat MIMO channel matrix $\mathbf{H}$ is in this case an $r \times m$ mixing matrix $(r \geq m)$. We use the classical signal model used in source separation. The $r \times N$ matrix $\mathbf{X}$ corresponding to the received signal may be written as $\mathbf{X} = \mathbf{HS} + \mathbf{V}$, where $\mathbf{S}$ is an $m \times N$ matrix corresponds to the m transmitted signals and $\mathbf{V}$ is the additive white noise. In the prewhitening stage the received signal is decorrelated based on the eigendecomposition of the correlation matrix. The prewhitened received signal is given by $\mathbf{Y} = \mathbf{\Lambda}_m^{-1/2}\mathbf{U}_m^H\mathbf{X}$, where $\mathbf{U}_m$ and $\mathbf{\Lambda}_m$ contain the m eigenvectors and the m eigenvalues corresponding to the signal subspace, respectively.

In the second stage, the goal is to determine a unitary matrix $\mathbf{W}$ such that the estimated signals $\hat{\mathbf{S}} = \mathbf{WY}$ are the transmitted signals up to a phase and a permutation ambiguity, which are inherent to any blind methods. The unitary matrix may be obtained by exploiting the information provided by the fourth-order cumulants of the whitened signals. The JADE algorithm minimizes the following criterion:

$$\mathcal{J}_{\mathrm{JADE}}(\mathbf{W}) = \sum_{i=1}^{m} \mathrm{off}\{\mathbf{W}^H\hat{\mathbf{M}}_i\mathbf{W}\} \tag{24}$$

with respect to $\mathbf{W}$, under the unitarity constraint on $\mathbf{W}$, i.e., we deal with a minimization problem on $U(m)$. The eigenmatrices $\hat{\mathbf{M}}_i$ which are estimated from the fourth-order cumulants need to be diagonalized. The operator $\mathrm{off}\{\cdot\}$ computes the sum of the squared magnitudes of the off-diagonal elements of a matrix, therefore, the criterion penalizes the departure of all eigenmatrices from the diagonal property. The Euclidean gradient of the JADE cost function is $\mathbf{\Gamma}_{\mathbf{W}} = 2\sum_{i=1}^{m} \hat{\mathbf{M}}_i\mathbf{W}\big[\mathbf{W}^H\hat{\mathbf{M}}_i\mathbf{W} - \mathbf{I} \odot (\mathbf{W}^H\hat{\mathbf{M}}_i\mathbf{W})\big]$, where $\odot$ denotes the elementwise matrix multiplication.

The performance is studied in terms of convergence speed considering the *JADE criterion* and the *Amari distance* (performance index) [12]. This JADE criterion (24) is a measure of how well the eigenmatrices $\hat{\mathbf{M}}_i$ are jointly diagonalized. This characterizes the goodness of the optimization solution, i.e., the unitary rotation stage of the BSS. The Amari distance d_{A} is a good performance measure for the entire blind source separation problem since it is invariant to permutation and scaling. In terms of deviation from the unitary constraint the performance is measured by using a *unitarity criterion* (22), in a logarithmic scale.

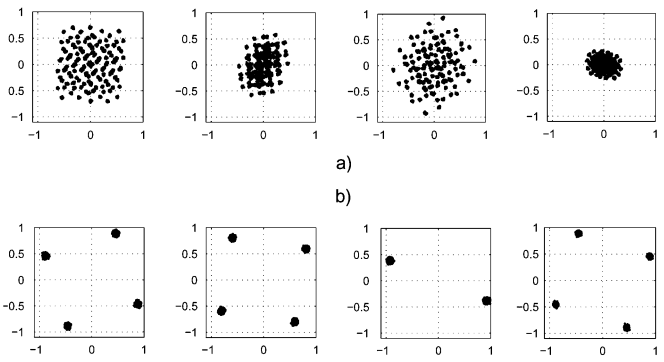

Fig. 4. The constellation patterns corresponding to (a) four of the six received signals and (b) the four recovered signals by using JADE with the algorithm proposed in Table I. There is an inherent phase ambiguity which may be noticed as a rotation of the constellation, as well as a permutation ambiguity.

A number of $m = 4$ signals are transmitted, three QPSK signals and one BPSK signal. The signal-to-noise ratio (SNR) is $\mathrm{SNR} = 20$ dB and the channel taps are independent random coefficients with power distributed according to a Rayleigh distribution. The results are averaged over 100 random realizations of the (4×6) MIMO matrix $\mathbf{H}$ and (4×1000) signal matrix.

In the first simulation, we compare three optimization algorithms: the classical Euclidean SD algorithm which enforces the unitarity of $\mathbf{W}$ after every iteration, the Euclidean SD with extra-penalty similar to [16] stemming from the Lagrange multipliers method and the proposed Riemannian SD algorithm from Table II. The update rule for the classical SD algorithm is $\mathbf{W}_{k+1} = \mathbf{W}_k - \eta\mathbf{\Gamma}_{\mathbf{W}_k}$. The unitarity property is enforced by symmetric orthogonalization after every iteration [12], i.e., $\mathbf{W}_{k+1} := \mathbf{W}_{k+1}(\mathbf{W}_{k+1}^H\mathbf{W}_{k+1})^{-1/2}$. The extra-penalty SD method uses an additional term $\mathcal{J}_u(\mathbf{W}) = \|\mathbf{W}^H\mathbf{W} - \mathbf{I}\|_F^2$ added to the original cost function (24) similarly to the bigradient method in [16]. The corresponding update rule is $\mathbf{W}_{k+1} = \mathbf{W}_k - \eta_L[\mathbf{\Gamma}_{\mathbf{W}_k} + 2\lambda\mathbf{W}_k(\mathbf{W}_k^H\mathbf{W}_k - \mathbf{I})]$. A weighting factor $\lambda = 0.5$ is used to weight the importance of the unitarity constraint. The third method is the SD algorithm summarized in Table II. Armijo [1] step size selection rule is used for all four methods. Fig. 4 shows received signal mixtures, and separated signals by using JADE with the proposed algorithm. The performance of the three algorithms in terms of convergence speed and accuracy of satisfying the unitary constraint are presented in Fig. 5. The JADE criterion (24) versus the number of iterations is shown in subplot a) of Fig. 5. Subplot b) of Fig. 5 shows the evolution of the Amari distance with the number of iterations. We may notice that the accuracy of the optimization solution described by the value of the JADE cost function is very related to the accuracy of solving the entire source separation, i.e., the Amari distance. The Riemannian SD algorithm (Table I) converges faster compared to the classical

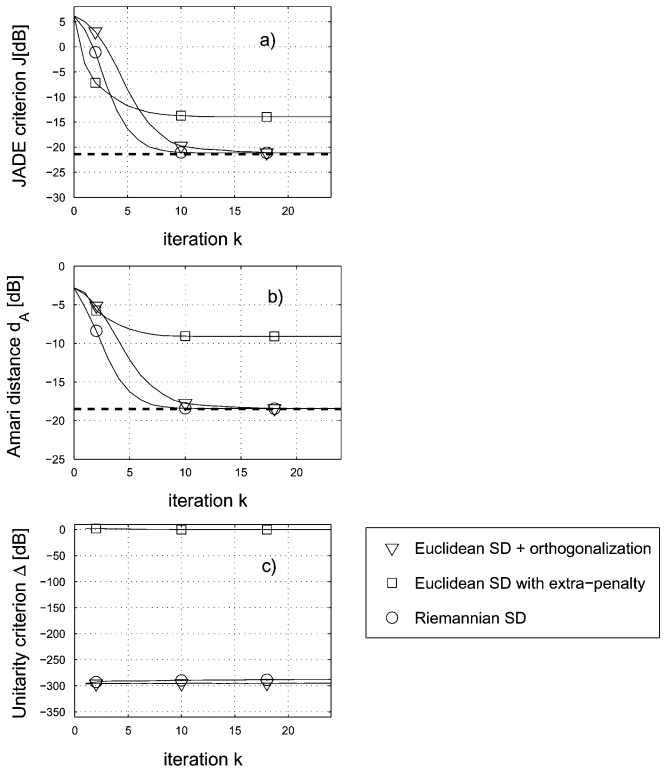

Fig. 5. A comparison between the conventional optimization methods operating on the Euclidean space (the classical SD algorithm with enforcing unitarity, the extra-penalty SD method) and the Riemannian SD algorithm from Table II. The horizontal thick dotted line in subplots (a) and (b) represents the solution of the original JADE algorithm [3]. The performance measures are the JADE criterion $\mathcal{J}_{\mathrm{JADE}}(\mathbf{W}_k)$(24), Amari distance d_A and the unitarity criterion Δ_k(22) versus the iteration step. The Riemannian SD algorithm outperforms the conventional methods.

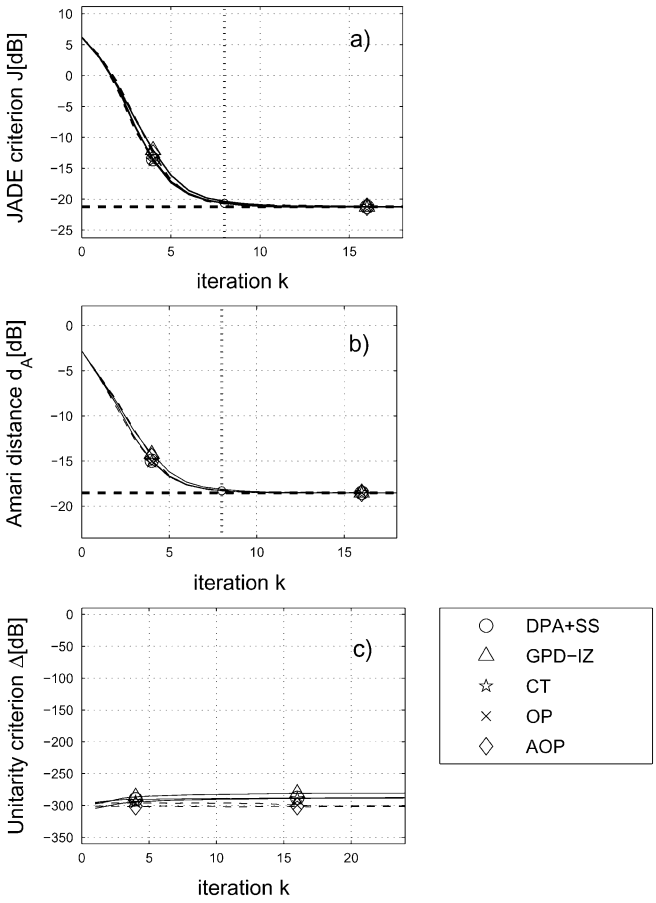

Fig. 6. The JADE criterion $\mathcal{J}_{\mathrm{JADE}}(\mathbf{W}_k)$(24), the Amari distance and the unitarity criterion Δ_k(22) versus the iteration step. A comparison between different local parametrizations on $U(n)$: the geodesic algorithms (continuous line) versus the nongeodesic algorithms (dashed line). For the geodesic algorithms the exponential map is computed by using three different methods: Matlab's diagonal Padé approximation with scaling and squaring (DPA+SS), GPD-IZ [30], CT. For the nongeodesic algorithms the projection operation is computed with two different methods: the OP via Matlab's `svd` function and the approximation of it (AOP) via QR decomposition. The horizontal thick dotted line in subplots (a) and (b) represents the solution of the original JADE algorithm [3].

methods, i.e., the Euclidean SD with enforcing unitarity and the extra-penalty method. The Euclidean SD and extra-penalty method do not operate in an appropriate parameter space, and the convergence speed is decreased. All SD algorithms satisfy perfectly the unitary constraint, except for the extra-penalty method which achieves also the lowest convergence speed. This is due to the fact that an optimum *scalar* weighting parameter λ may not exist. The unitary matrix constraint is equivalent to n^2 smooth real Lagrangian constrains, therefore more parameters could be used. However, computing these parameters may computationally expensive and/or non-trivial even in the case of $n = 1$, like the example presented in Section II-B. The accuracy of the solution in terms of unitary constraint is shown in subplot c) of Fig. 5 considering the criterion Δ_k (22) versus the number of iterations.

We will next analyze how the choice of the local parametrization affects the performance of the SD algorithms. The results are compared in terms of convergence speed and the accuracy of satisfying the unitary constraint. Two classes of Riemannian SD algorithms are considered. The first one includes the geodesic SD algorithms and the second one include the nongeodesic SD algorithms. For the geodesic SD algorithms the exponential map is computed by three different methods: the Matlab's `expm` function which uses the diagonal Padé approximation with scaling and squaring $(\mathrm{DPA} + \mathrm{SS})$ [32], the Generalized Polar Decomposition of order four by Iselres and Zanna (GPD-IZ) [30] and the CT. The nongeodesic SD algorithms are based on a projection type of local parametrization. The OP is computed via SVD [2] by using the Matlab `svd` function and the approximation of the optimal projection (AOP) based on the QR algorithm is computed via modified Gram-Schmidt procedure [54]. In terms of convergence speed, both the geodesic and the nongeodesic SD algorithms such as [2] and [19] have similar performance, regardless of the local parametrization, as shown in subplots a) and b) of Fig. 6. Also in terms of unitarity criterion, all algorithms provide good performance. The solution of the original JADE algorithm (represented by the horizontal thick dotted line in Fig. 6) is achieved in less than 10 iteration for all SD algorithms, regardless of the local parametrization.

In conclusion, the choice of the local parametrization in made according to the computational complexity and numerical stability. An Armijo step size rule is very suitable to the geodesic algorithms, i.e., using the exponential map as a local parametrization. If implemented in parallel architecture the

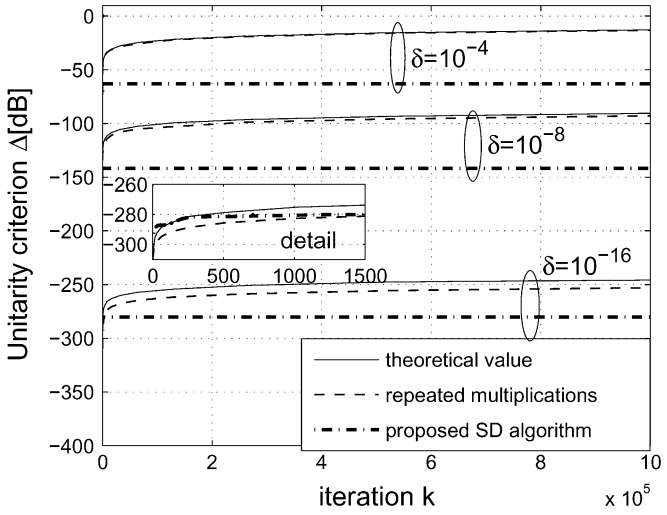

Fig. 7. The deviation from the unitary constraint for different values of the quantization errors δ on the rotation matrices $\mathbf{P}_k$ after one million iterations. The theoretical value $E[\Delta_k]$ in (23) is represented by continuous black line. The unitarity criterion Δ_k(22) obtained by repeatedly multiplying unitary matrices is represented by dashed gray line. The value Δ_k obtained by using the proposed SD algorithm in Table II is represented by dot-dashed thick black line. The theoretical expected value of the deviation from the unitary constraint predicts accurately the value obtained in simulations. The proposed algorithm produces an error lower that the theoretical bound due to the fact that the error does not accumulate after the convergence has been reached.

GPD-IZ method [30] for computing the matrix exponential is a reliable choice from the point of view of efficiency and computational speed. Otherwise, the CT provides a reasonable performance at a low computational cost. Finally, the proposed algorithm has lower computational complexity per iteration compared to the nongeodesic SD in [2], [19] at the same convergence speed. This reduction is achieved by exploiting the Lie group structure of the Stiefel manifold of $n \times n$ unitary matrices.

The last simulation shows how the round-off errors caused by finite numerical precision affect the proposed iterative algorithm. In Fig. 7, different values of the quantization error δ are considered for the rotation matrices $\tilde{\mathbf{P}}_k$, which are obtained by using the $\mathrm{DPA} + \mathrm{SS}$ method [32]. Similar results are obtained by using the other approximation methods of the matrix exponential presented in Section V-A. The theoretical value of the unitarity criterion $E[\Delta_k]$ in (23) is represented by continuous lines. The value Δ_k (22) obtained by repeatedly multiplying unitary matrices is represented by dashed lines and the value obtained by using the proposed algorithm in Table II is represented by dot-dashed thick lines. The theoretical value (23) predicts accurately the value (22) obtained by repeated multiplications of unitary matrices. The proposed algorithm exhibits an error below the theoretical value due to the fact that the convergence is reached in few steps. Even if the convergence would be reached after a much larger number of iterations, the error accumulation is negligible for reasonable values of the quantization errors ($\delta < 10^{-8}$), as shown in Fig. 7. In practice, a much smaller number of iterations need to be performed.

VIII. Conclusion

In this paper, Riemannian optimization algorithms on the Lie group of $n \times n$ unitary matrices $U(n)$ have been introduced. Expression for Riemannian gradient needed in the optimization has been derived. The proposed algorithms move towards the optimum along geodesics and the local parametrization is the exponential map. We exploit the recent developments in computing the matrix exponential needed in the multiplicative update on $U(n)$. This operation may be efficiently computed in a parallel fashion by using the GPD-IZ method [30] or in a serial fashion by using the CT. We also address the numerical issues and show that the geodesic algorithms together with the Armijo rule [1] are more efficient in practical implementations. Nongeodesic algorithms have been considered as well, and equivalence up to a certain approximation order has been established.

The proposed geodesic algorithms are suitable for practical applications where a closed form solution does not exist, or to refine estimates obtained by classical means. Such an example is the joint diagonalization problem presented in the paper. We have shown that the unitary matrix optimization problem encountered in the JADE approach for blind source separation [3] may be efficiently solved by using the proposed algorithms. Other possible applications include: smart antenna algorithms, wireless communications, biomedical measurements and signal separation, where unitary matrices play an important role in general. The algorithms introduced in this paper provide significant advantages over classical Euclidean gradient with enforcing unitary constraint and Lagrangian type of methods in terms convergence speed and accuracy of the solution. The unitary constraint is automatically maintained at each iteration, and consequently, undesired suboptimal solutions may be avoided. Moreover, for the specific case of $U(n)$, the proposed algorithm has lower computational complexity than the nongeodesic SD algorithms in [2].

References

[1] E. Polak, *Optimization: Algorithms and Consistent Approximations*. New York: Springer-Verlag, 1997.

[2] J. H. Manton, "Optimization algorithms exploiting unitary constraints," *IEEE Trans. Signal Process.*, vol. 50, pp. 635–650, Mar. 2002.

[3] J. Cardoso and A. Souloumiac, "Blind beamforming for non Gaussian signals," *Inst. Elect. Eng. Proc.-F*, vol. 140, no. 6, pp. 362–370, 1993.

[4] S. T. Smith, "Subspace tracking with full rank updates," in *Conf. Rec. 31st Asilomar Conf. Signals, Syst. Comp.*, Nov. 2–5, 1997, vol. 1, pp. 793–797.

[5] D. R. Fuhrmann, "A geometric approach to subspace tracking," in *Conf. Rec. 31st Asilomar Conf. Signals, Syst., Comp.*, Nov. 2–5, 1997, vol. 1, pp. 783–787.

[6] J. Yang and D. B. Williams, "MIMO transmission subspace tracking with low rate feedback," in *Int. Conf. Acoust., Speech Signal Process.*, Philadelphia, PA, Mar. 2005, vol. 3, pp. 405–408.

[7] W. Utschick and C. Brunner, "Efficient tracking and feedback of DL-eigenbeams in WCDMA," in *Proc. 4th Europ. Pers. Mobile Commun. Conf.*, Vienna, Austria, 2001.

[8] M. Wax and Y. Anu, "A new least squares approach to blind beamforming," in *IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 21–24, 1997, vol. 5, pp. 3477–3480.

[9] P. Stoica and D. A. Linebarger, "Optimization result for constrained beamformer design," in *IEEE Signal Process. Lett.*, Apr. 1995, vol. 2, pp. 66–67.

[10] Y. Nishimori, "Learning algorithm for independent component analysis by geodesic flows on orthogonal group," in *Int. Joint Conf. Neural Netw.*, Jul. 10–16, 1999, vol. 2, pp. 933–938.

[11] Y. Nishimori and S. Akaho, "Learning algorithms utilizing quasi-geodesic flows on the Stiefel manifold," *Neurocomputing*, vol. 67, pp. 106–135, Jun. 2005.

[12] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.
[13] S. Fiori, A. Uncini, and F. Piazza, "Application of the MEC network to principal component analysis and source separation," in *Proc. Int. Conf. Artif. Neural Netw.*, 1997, pp. 571–576.
[14] M. D. Plumbley, "Geometrical methods for non-negative ICA: Manifolds, Lie groups, toral subalgebras," *Neurocomput.*, vol. 67, pp. 161–197, 2005.
[15] A. Cichocki and S.-I. Amari, *Adaptive Blind Signal and Image Processing*. New York: Wiley, 2002.
[16] L. Wang, J. Karhunen, and E. Oja, "A bigradient optimization approach for robust PCA, MCA and source separation," in *Proc. IEEE Conf. Neural Netw.*, 27 Nov.–1 Dec. 1995, vol. 4, pp. 1684–1689.
[17] S. C. Douglas, "Self-stabilized gradient algorithms for blind source separation with orthogonality constraints," *IEEE Trans. Neural Netw.*, vol. 11, pp. 1490–1497, Nov. 2000.
[18] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, 1998.
[19] M. Nikpour, J. H. Manton, and G. Hori, "Algorithms on the Stiefel manifold for joint diagonalisation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2002, vol. 2, pp. 1481–1484.
[20] C. B. Papadias, "Globally convergent blind source separation based on a multiuser kurtosis maximization criterion," *IEEE Trans. Signal Process.*, vol. 48, pp. 3508–3519, Dec. 2000.
[21] P. Sansrimahachai, D. Ward, and A. Constantinides, "Multiple-input multiple-output least-squares constant modulus algorithms," in *IEEE Global Telecommun. Conf.*, Dec. 1–5, 2003, vol. 4, pp. 2084–2088.
[22] C. B. Papadias and A. M. Kuzminskiy, "Blind source separation with randomized Gram-Schmidt orthogonalization for short burst systems," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 17–21, 2004, vol. 5, pp. 809–812.
[23] J. Lu, T. N. Davidson, and Z. Luo, "Blind separation of BPSK signals using Newton's method on the Stiefel manifold," in *IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2003, vol. 4, pp. 301–304.
[24] A. Edelman, T. Arias, and S. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Analysis Applicat.*, vol. 20, no. 2, pp. 303–353, 1998.
[25] S. Fiori, "Stiefel-Grassman Flow (SGF) learning: Further results," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw.*, Jul. 24–27, 2000, vol. 3, pp. 343–348.
[26] J. H. Manton, R. Mahony, and Y. Hua, "The geometry of weighted low-rank approximations," *IEEE Trans. Signal Process.*, vol. 51, pp. 500–514, Feb. 2003.
[27] S. Fiori, "Quasi-geodesic neural learning algorithms over the orthogonal group: A tutorial," *J. Mach. Learn. Res.*, vol. 1, pp. 1–42, Apr. 2005.
[28] D. G. Luenberger, "The gradient projection method along geodesics," *Manage. Sci.*, vol. 18, pp. 620–631, 1972.
[29] D. Gabay, "Minimizing a differentiable function over a differential manifold," *J. Optim. Theory Applicat.*, vol. 37, pp. 177–219, Jun. 1982.
[30] A. Iserles and A. Zanna, "Efficient computation of the matrix exponential by general polar decomposition," *SIAM J. Numer. Anal.*, vol. 42, pp. 2218–2256, Mar. 2005.
[31] I. Yamada and T. Ezaki, "An orthogonal matrix optimization by dual Cayley parametrization technique," in *Proc. ICA*, 2003, pp. 35–40.
[32] C. Moler and C. van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM Rev.*, vol. 45, no. 1, pp. 3–49, 2003.
[33] S. Douglas and S.-Y. Kung, "An ordered-rotation kuicnet algorithm for separating arbitrarily-distributed sources," in *Proc. IEEE Int. Conf. Independ. Compon. Anal. Signal Separat.*, Aussois, France, Jan. 1999, pp. 419–425.
[34] C. Udriste, *Convex Functions and Optimization Methods on Riemannian Manifolds. Mathematics and Its Applications*. Boston, MA: Kluwer Academic, 1994.
[35] M. P. do Carmo, *Riemannian Geometry. Mathematics: Theory and Applications*. Boston, MA: Birkhauser, 1992.
[36] A. Knapp, *Lie Groups Beyond an Introduction, Vol. 140 of Progress in Mathematics*. Boston, MA: Birkhauser, 1996.
[37] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.
[38] S. T. Smith, "Optimization techniques on Riemannian manifolds," *Fields Inst. Commun., Amer. Math. Soc.*, vol. 3, pp. 113–136, 1994.
[39] R. W. Brockett, "Least squares matching problems," *Linear Algebra Applicat.*, vol. 122/123/124, pp. 761–777, 1989.
[40] R. W. Brockett, "Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems," *Linear Algebra Applicat.*, vol. 146, pp. 79–91, 1991.
[41] B. Owren and B. Welfert, "The Newton iteration on Lie groups," *BIT Numer. Math.*, vol. 40, pp. 121–145, Mar. 2000.
[42] P.-A. Absil, R. Mahony, and R. Sepulchre, "Riemannian geometry of Grassmann manifolds with a view on algorithmic computation," *Acta Applicandae Mathematicae*, vol. 80, no. 2, pp. 199–220, 2004.
[43] S. G. Krantz, *Function Theory of Several Complex Variables*, 2nd ed. Pacific Grove, CA: Wadsworth and Brooks/Cole Advanced Books and Software, 1992.
[44] S. Smith, "Statistical resolution limits and the complexified Cramér-Rao bound," *IEEE Trans. Signal Process.*, vol. 53, pp. 1597–1609, May 2005.
[45] D. H. Brandwood, "A complex gradient operator and its applications in adaptive array theory," in *Inst. Elect. Eng. Proc., Parts F and H*, Feb. 1983, vol. 130, pp. 11–16.
[46] Y. Yang, "Optimization on Riemannian manifold," in *Proc. 38th Conf. Decision Contr.*, Phoenix, AZ, Dec. 1999, pp. 888–893.
[47] N. J. Higham, "Matrix nearness problems and applications," in *Applications of Matrix Theory*, M. J. C. Gover and S. Barnett, Eds. Oxford, U.K.: Oxford Univ. Press, 1989, pp. 1–27.
[48] A. Zanna and H. Z. Munthe-Kaas, "Generalized polar decomposition for the approximation of the matrix exponential," *SIAM J. Matrix Anal.*, vol. 23, pp. 840–862, Jan. 2002.
[49] E. Celledoni and A. Iserles, "Methods for approximation of a matrix exponential in a Lie-algebraic setting," *IMA J. Numer. Anal.*, vol. 21, no. 2, pp. 463–488, 2001.
[50] T. Abrudan, J. Eriksson, and V. Koivunen, "Optimization under unitary matrix constraint using approximate matrix exponential," in *Conf. Rec. 39th Asilomar Conf. Signals, Syst. Comp. 2005*, 28 Oct.–1 Nov. 2005.
[51] P. Sansrimahachai, D. Ward, and A. Constantinides, "Blind source separation for BLAST," in *Proc. 14th Int. Conf. Digital Signal Process.*, Jul. 1–3, 2002, vol. 1, pp. 139–142.
[52] J. H. Manton, "On the role of differential geometry in signal processing," in *Int. Conf. Acoust., Speech Signal Process.*, Philadelphia, PA, Mar. 2005, vol. 5, pp. 1021–1024.
[53] P. Absil and K. A. Gallivan, Joint diagonalization on the oblique manifold for independent component analysis DAMTP, Univ. Cambridge, U.K., Tech. Rep. NA2006/01, 2006 [Online]. Available: http://www.damtp.cam.ac.uk/user/na/reports.html
[54] G. H. Golub and C. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

Traian E. Abrudan (S'02) received the M.Sc. degree from the Technical University of Cluj-Napoca, Romania, in 2000.

Since 2001, he has been with the Signal Processing Laboratory, Helsinki University of Technology (HUT), Finland. He is a Ph.D. student with the Electrical and Communications Engineering Department, HUT. Since 2005, he has been a member of GETA, Graduate School in Electronics, Telecommunications and Automation. His current research interests include statistical signal processing and optimization algorithms for wireless communications with emphasis on MIMO and multicarrier systems.

Jan Eriksson (M'04) received the M.Sc. degree in mathematics from University of Turku, Finland, in 2000, and the D.Sc.(Tech) degree (with honors) in signal processing from Helsinki University of Technology (HUT), Finland, in 2004.

Since 2005, he has been working as a postdoctoral researcher with the Academy of Finland. His research interests are in blind signal processing, stochastic modeling, constrained optimization, digital communication, and information theory.

Visa Koivunen (S'87–M'93–SM'98) received the D.Sc. (Tech) degree with honors from the Department of Electrical Engineering, University of Oulu, Finland. He received the primus doctor (best graduate) award among the doctoral graduates during 1989–1994.

From 1992 to 1995, he was a visiting researcher at the University of Pennsylvania, Philadelphia. In 1996, he held a faculty position with the Department of Electrical Engineering, University of Oulu. From August 1997 to August 1999, he was an Associate Professor with the Signal Processing Labroratory, Tampere University of Technology, Finland. Since 1999 he has been a Professor of Signal Processing with the Department of Electrical and Communications Engineering, Helsinki University of Technology (HUT), Finland. He is one of the Principal Investigators in Smart and Novel Radios (SMARAD) Center of Excellence in Radio and Communications Engineering nominated by the Academy of Finland. Since 2003, he has also been an adjunct full professor with the University of Pennsylvania. During his sabbatical leave (2006–2007), he was the Nokia Visiting Fellow at the Nokia Research Center, as well as a Visiting Fellow at Princeton University, Princeton, NJ. His research interests include statistical, communications, and sensor array signal processing. He has published more than 200 papers in international scientific conferences and journals.

Dr. Koivunen coauthored the papers receiving the Best Paper award in IEEE PIMRC 2005, EUSIPCO 2006, and EuCAP 2006. He served as an Associate Editor for IEEE SIGNAL PROCESSING LETTERS. He is a member of the editorial board for the *Signal Processing* journal and the *Journal of Wireless Communication and Networking*. He is also a member of the IEEE Signal Processing for Communication Technical Committee (SPCOM-TC). He was the general chair of the IEEE SPAWC (Signal Processing Advances in Wireless Communication) conference held in Helsinki, June 2007.