Neural Learning by Geometric Integration of Reduced 'Rigid-Body' Equations*

Elena Celledoni† Simone Fiori‡

January 27, 2004

Abstract

In previous contributions, the second author of this paper presented a new class of algorithms for orthonormal learning of linear neural networks with p inputs and m outputs, based on the equations describing the dynamics of a massive rigid frame on the Stiefel manifold. These algorithms exhibit good numerical stability, strongly binding to the submanifold of constraints, and good controllability of the learning dynamics, but are not completely satisfactory from a computational-complexity point of view. In the recent literature, efficient methods of integration on the Stiefel manifold have been proposed by various authors, see for example [6, 9, 10, 15]. Inspired by these approaches, in this paper, we propose a new and efficient representation of the mentioned learning equations, and a new way to integrate them. The numerical experiments show how the new formulation leads to significant computational savings especially when $p \gg m$. The effectiveness of the algorithms is substantiated by numerical experiments concerning Principal Subspace Analysis and Independent Component Analysis. These experiments were carried out with both synthetic and real-world data.

1 Introduction

During the last years, several contributions appeared in the neural network literature as well as in other research areas regarding neural learning and optimisation involving flows on special sets (such as the Stiefel manifold).

*This work is part of the activities of the special year on Geometric Integration at the Center for Advanced Study in Oslo.

†SINTEF Applied Mathematics, Sem Sælandsvei 5, 7465 Trondheim, Norway, Email: Elena.Celledoni@math.ntnu.no, URL: http://www.math.ntnu.no/~elenac/

‡University of Perugia, Polo Didattico e Scientifico del Ternano, I-05100 Terni, Italy, Email: fiori@unipg.it, URL: http://www.unipg.it/sfr/

The analysis of these contributions has raised the idea that geometric concepts (such as the theory of Lie groups) give the fundamental instruments for gaining a deep insight into the mathematical properties of several learning and optimisation paradigms.

The interest displayed by the scientific community about this research topic is also testified by several activities such as the organisation of the special issue on "Non-Gradient Learning Techniques" of the International Journal of Neural Systems (guest editors A. de Carvalho and S.C. Kremer), the Post-NIPS*2000 workshop on "Geometric and Quantum Methods in Learning", organised by S.-i. Amari, A. Assadi and T. Poggio (Colorado, December 2000), the workshop "Uncertainty in Geometric Computations" held in Sheffield, England, in July 2001, organised by J. Winkler and M. Niranjan, the special session of the IJCNN'02 conference on "Differential & Computational Geometry in Neural Networks" held in Honolulu, Hawaii (USA), in May 2002 and organised by E. Bayro-Corrochano, and the workshop "Information Geometry and its Applications", held in Pescara (Italy), in July 2002, organised by P. Giblisco.

Understanding the underlying geometric structure of a network parameter space is extremely important for designing systems that can effectively navigate the space while learning.

Over the last decade or so, driven greatly by the work on information geometry, we are seeing the merging of the fields of statistics and geometry applied to neural networks and learning. Research topics include differential geometrical methods for learning, the Lie group learning algorithms [23], and the natural (Riemannian) gradient techniques [2, 25, 31, 35].

Some specific exemplary applied topics that can be addressed under the mentioned general methodology are for example principal component/subspace analysis [21, 43], neural independent component analysis and blind source separation [21, 44], information geometry [2], geometric Clifford algebra for the generalisation of neural networks [3], geometrical methods of unsupervised learning for blind signal processing [21, 23], eigenvalue and generalised eigenvalue problems, optimal linear compression, noise reduction and signal representation [14, 17, 36, 42, 43], simulation of the physics of bulk materials [16], minimal linear system realization from noise-injection measured data and invariant subspace computation [16, 33], optimal de-noising by sparse coding shrinkage [37], steering of antennas arrays [1], linear programming and sequential quadratic programming [7, 16], optical character recognition by transformation-invariant neural networks [40], analysis of geometric constraints on neural activity for nat-

ural three-dimensional movement [45], electrical networks fault detection [32], synthesis of digital filters by improved total least-squares technique [26], speaker verification [41], adaptive image coding [34] and dynamic texture recognition [39].

From the numerical point of view, the solution of matrix-type differential equations on Lie groups and homogeneous spaces has been vastly investigated in the last years in the context of Geometric Integration [9, 10, 15, 28, 30]. Geometric Integration (GI) is a recent branch of numerical analysis. The traditional efforts of numerical analysis and computational mathematics have been to render physical phenomena into algorithms that produce sufficiently precise, affordable and robust numerical approximations. Geometric integration is concerned also with producing numerical approximations preserving the qualitative attributes of the solution to the possible extent. Examples of GI algorithms for differential equations include Lie group integrators, volume and energy preserving integrators, integrators preserving first integrals and Lyapunov functions, Lagrangian and variational integrators, integrators respecting Lie symmetries and integrators preserving contact structures [28].

As a contribution to the research field of geometric methods in neural learning, a new learning theory derived from the study of the dynamics of an abstract system of masses, moving in a multidimensional space under an external force field, was presented and studied in details in [22, 23]. The set of equations describing the dynamics of the system was interpreted as a learning algorithm for neural layers termed MEC[1]. Relevant properties of the proposed learning theory were discussed, along with results of computer-based experiments performed in order to assess its effectiveness in applied fields. In particular, some applications of the proposed approach were suggested and cases of orthonormal independent component analysis and principal component analysis were tackled through computer simulations.

However, an open question about the mentioned algorithm concerned the computational complexity of the numerical approach, which seemed not optimal due to the necessity of heavy matrix computations, such as the repeated evaluation of the matrix exponential.

In the present article, we address this last issue reformulating the learning equations and obtaining new integration algorithms. The new numerical approach preserves the geometric features of the underlying equations in the spirit

[1]The name given to the algorithm stems from the initials of the word 'mechanical', as the abstract system of masses constitutes a classical problem in rational mechanics.

of Geometric Integration and, at the same time, the computational effort is significantly reduced.

We start by deriving a new characterisation of second order differential equations on the Stiefel manifold (Theorem 4, Section 2.3). This characterisation is well suited and essential for the application of low-complexity numerical integration preserving orthogonality. From the result of Theorem 4, a new and different formulation of the MEC learning equations arises. A partitioned integrator preserving orthogonality is then considered for the numerical solution of the MEC learning equations. The method requires the repeated approximation of the matrix exponential of skew-symmetric matrices with special structure, which we implement in a new low-complexity algorithm (Section 3). The new formulation allows us to achieve significantly reduced computational complexity and execution times in the performed simulations.

2 Summary of the MEC Theory and Proposed Improvement

In orthonormal learning, the target of the adaptation rule for a neural network is to learn a weight-matrix with orthonormal columns related in some way to the input signal. Let us denote by $\mathrm{St}(p, m, \mathbb{R})$ the set of the real-valued $p \times m$ matrices with orthonormal columns (usually termed compact Stiefel manifold [5]). When $p = m$, the manifold $\mathrm{St}(p, p, \mathbb{R})$ coincides to the group of the $p \times p$ orthogonal matrices, i.e. the orthogonal group $O(p, \mathbb{R})$. Since it is a prior knowledge that the final neural state must belong to $\mathrm{St}(p, m, \mathbb{R})$, the evolution of the weight-matrix could be strongly bounded to always belong to $\mathrm{St}(p, m, \mathbb{R})$.

We solved this strongly-binding problem by adopting as columns of the weight-matrix the position-vectors of some masses of a rigid system. Because of the intrinsic rigidity of the system, the required constraint is always respected.

By recalling that a (dissipative) mechanical system reaches the equilibrium when its own potential energy function (PEF) is at its minimum or local minima, a PEF may be assumed proportional to a cost function to be minimised, or proportional to an objective function to be maximised, both *under the constraint of orthonormality*.

In the following sections we briefly recall the mentioned theory, its principal features and the drawbacks related to its computational complexity. We then describe the proposed improvement based on an advantageous parameterisation

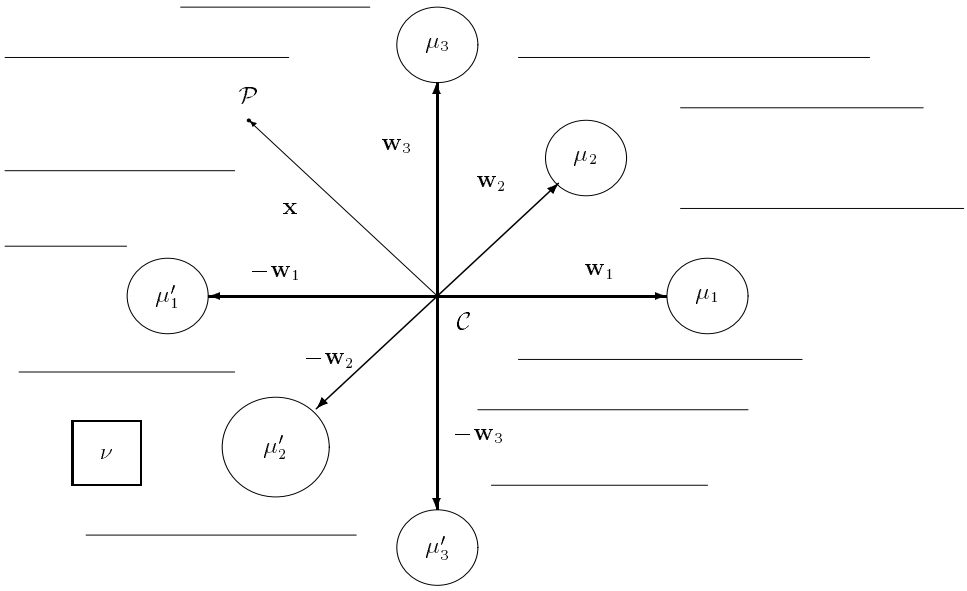

Figure 1: A configuration of $\mathcal{S}_m$ for $p = 3$ and $m = 3$. The μ_i and μ'_i represent the masses, the vectors $\mathbf{w}_i$ represent their coordinates and $\mathbf{x}$ is the coordinate-vector of the external point $\mathcal{P}$.

of the angular-velocity space.

2.1 Summary of rigid-body learning theory

Let $\mathcal{S}_m = \{(\mu_i, \mathbf{w}_i), (\mu'_i, -\mathbf{w}_i)\}_{i\in\{1,\ldots,m\}}$ be a *rigid* system of masses, where the m vectors $\mathbf{w}_i \in \mathbb{R}^p$ represent the instantaneous positions of $2m$ masses $\mu_i, \mu'_i \in \mathbb{R}^+_0$ in a coordinate system. Such masses are positioned at constant (unitary) distances from the origin $\mathcal{C}$ fixed in the space $\mathbb{R}^p$, and over mutually orthogonal immaterial axes. In [22] we assumed the values of the masses μ_i and μ'_i constant at 1. In Figure 1 an exemplary configuration of $\mathcal{S}_m$ for $p = 3$ and $m = 3$ is illustrated.

Note that by definition the system has been assumed rigid with the axes origin $\mathcal{C}$ fixed in space, thus the masses are allowed only to instantaneously rotate around this point, while they cannot translate with respect to it. Also note that, thanks to its structural symmetry, the massive system is also inertially equilibrated.

The masses and a physical point $\mathcal{P}$ (endowed with a negligible mass) move in $\mathbb{R}^p$. The position of $\mathcal{P}$ with respect to $\mathcal{C}$ is described by an independent vector

$\mathbf{x}$. The point $\mathcal{P}$ exerts a force on each mass and the set of forces generated causes the motion of the global system $\mathcal{S}_m$. Furthermore, masses move in a homogeneous and isotropic fluid endowed with a non-negligible viscosity. The corresponding resistance brakes the motion, makes the system dissipative and stabilises its dynamics.

The equations describing the motion of such abstract system are summarised in the following result.

Theorem 1 *([22].) Let $\mathcal{S}_m \subset \mathbb{R}_0^+ \times \mathbb{R}^p$ be the physical system described above. Let us denote with $\mathbf{F}$ the $p \times m$ matrix of the* active forces*, with $\mathbf{P}$ the $p \times m$ matrix of the* viscosity resistance*, with $\mathbf{B}$ the $p \times p$* angular speed *matrix and with $\mathbf{W}$ the $p \times m$ matrix of the* instantaneous positions *of the masses. The dynamics of the system obeys the following equations:*

$$\frac{d\mathbf{W}}{dt} = \mathbf{B}\mathbf{W} \ , \tag{1}$$

$$\frac{d\mathbf{B}}{dt} = (\mathbf{F}+\mathbf{P})\mathbf{W}^T - \mathbf{W}(\mathbf{F}+\mathbf{P})^T \ , \tag{2}$$

$$\mathbf{P} = -\nu\mathbf{B}\mathbf{W} \ , \tag{3}$$

with ν being a positive parameter termed viscosity coefficient*.* □

The set of equations (1)-(3) may be assumed as a learning rule (briefly referred to as *MEC*) for a neural layer with weight-matrix $\mathbf{W}$. The MEC adaptation algorithm applies to any neural network described by the input-output transference $\mathbf{y} = \mathbf{S}[\mathbf{W}^T\mathbf{x} + \mathbf{w}_0]$, where $\mathbf{x} \in \mathbb{R}^p$, $\mathbf{W}$ is $p \times m$, with $m \leq p$, $\mathbf{w}_0$ is a generic biasing vector in $\mathbb{R}^m$ and $\mathbf{S}[\cdot]$ is an arbitrarily-chosen $m \times m$ diagonal activation operator.

Provided that initial conditions $\mathbf{B}(0) = \mathbf{B}_0$ and $\mathbf{W}(0) = \mathbf{W}_0$, and the expression of $\mathbf{F}$ as a function of $\mathbf{W}$ are given, the equations (1)-(3) represent an initial-value problem in the matrix-variables $(\mathbf{B}, \mathbf{W})$, whose asymptotic solution $\mathbf{W}_\star$ represents the neural network connection pattern after learning.

The basic properties of this algorithm may be summarised as follows.

- Let us denote by $\mathbf{so}(p, \mathbb{R})$ the linear space of skew-symmetric matrices, which has also a Lie-algebra structure. It is immediate to verify that if $\mathbf{B}(0) \in \mathbf{so}(p, \mathbb{R})$ then $\mathbf{B}(t) \in \mathbf{so}(p, \mathbb{R})$, since equation (2) provides $\dot{\mathbf{B}}(t) \in \mathbf{so}(p, \mathbb{R})$.

- Because of the skew-symmetry of $\mathbf{B}(t)$, from equation (1) it follows that if $\mathbf{W}(0) \in \mathrm{St}(p, m, \mathbb{R})$ then $\mathbf{W}(t) \in \mathrm{St}(p, m, \mathbb{R})$ for all $t > 0$.

- As a mechanical system, stimulated by a conservative force field, tends to *minimise* its potential energy, the set of learning equations (1)-(3), for a neural network with connection pattern $\mathbf{W}$, may be regarded as a non-conventional (second-order, non-gradient) optimisation algorithm.

The MEC learning rule possesses a fixed structure, the only modifiable part is the computation rule of the active forces applied to the masses. Here we suppose that the forcing terms derive from a potential energy function (PEF) U, which yields force

$$\mathbf{F} \stackrel{\text{def}}{=} -\frac{\partial U}{\partial \mathbf{W}} \ . \tag{4}$$

Generally, we may suppose to be interested in statistical signal processing, therefore we may regard the potential energy function U as dependent upon $\mathbf{W}$, $\mathbf{w}_0$ and on the statistics of $\mathbf{x}$; more formally $U = E_{\mathbf{x}}[u(\mathbf{W}, \mathbf{w}_0, \mathbf{x})]$, where $u(\cdot, \cdot, \cdot)$ represents a network's performance index and $E_{\mathbf{x}}[\cdot]$ denotes the statistical expectation (ensemble average) operator. Namely, we have

$$U(\mathbf{W}, \mathbf{w}_0) = \int_{\mathbb{R}^p} u(\mathbf{W}, \mathbf{w}_0, \mathbf{x}) q_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \ , \tag{5}$$

where $q_{\mathbf{x}}(\mathbf{x})$ denotes the joint probability density function of the scalar random variables in $\mathbf{x}$. Recalling that a (dissipative) mechanical system reaches an equilibrium state when its own potential energy U is at its minimum (or local minima), we can use as PEF any arbitrary smooth function to be optimised. Vector $\mathbf{w}_0$ may be arbitrarily adapted and, in the following, it will be ignored, so that we only have the dependencies

$$U = U(\mathbf{W}) \text{ and } \mathbf{F} = \mathbf{F}(\mathbf{W}) \ . \tag{6}$$

If we regard the above learning rule as a minimisation algorithm, the following observations might be worth noting.

- The searching space is considerably reduced. In fact, the set of matrices belonging to $\mathbb{R}^{p \times m}$, with $p \geq m$, has pm degrees of freedom, while the subset of $p \times m$ matrices with orthonormal columns has $pm - m(m+1)/2$ degrees of freedom.

- Non-orthonormal local (sub-optimal) solutions are inherently avoided as they do not belong to the search-space.

- The searching algorithm may be geodesic. The Stiefel manifold is a Riemannian manifold. A geodesic connecting two points of a Riemannian

manifold is the shortest pathway between them. A geodesic algorithm follows the geodesics between any pair of searching steps, thus providing the best local search-path.

To conclude the summary of MEC theory, it is worthwhile to mention two results about the stationary points of the algorithm and on their stability, see [23] for further details.

Theorem 2 *([23].) Let us consider the dynamical system (1)-(3) where the initial state is chosen so that* $\mathbf{W}(0) \in St(p, m, \mathbb{R})$ *and* $\mathbf{B}(0)$ *is skew-symmetric. Let us also define the matrix function* $\mathbf{F} \stackrel{\text{def}}{=} -\frac{\partial U}{\partial \mathbf{W}}$*, and denote as* $\mathbf{F}_\star$ *the value of* $\mathbf{F}$ *at* $\mathbf{W}_\star$*. A state* $\mathbf{X}_\star = (\mathbf{B}_\star, \mathbf{W}_\star)$ *is stationary for the system if* $\mathbf{F}_\star^T \mathbf{W}_\star$ *is symmetric and* $\mathbf{B}_\star \mathbf{W}_\star = \mathbf{0}$*. These stationary points are among the extremes of learning criterion* U *over* $St(p, m, \mathbb{R})$*.*

Recall that $\mathrm{SO}(p, \mathbb{R})$ is the set of real-valued square orthogonal matrices of dimension p, with determinant equal to one.

Theorem 3 *([23].) Let* U *be a real-valued function of* $\mathbf{W}$*,* $\mathbf{W} \in SO(p, \mathbb{R})$*, bounded from below with a minimum in* $\mathbf{W}_\star$*. Then the equilibrium state* $\mathbf{X}_\star = (\mathbf{0}, \mathbf{W}_\star)$*, is asymptotically (Lyapunov) stable for system (1)-(3) if* $\nu > 0$*, while simple stability holds if* $\nu \geq 0$*.*

2.2 Present study motivation

The discussed equations describing the MEC learning rule are based on two matrix state-variables, namely $\mathbf{B}$ and $\mathbf{W}$, whose dimensions are $p \times p$ and $p \times m$, respectively, where p denotes the number of neural network's inputs and m denotes the number of network's outputs. As a consequence, even if the dimension pm of the network is of reduced size, namely $m \ll p$, the state-matrix $\mathbf{B}$ assumes the largest possible dimension. An extreme example is represented by the one-unit network case, in which in order to train a single neuron ($m = 1$) with many inputs ($p \gg 1$) a full $p \times p$ angular-velocity matrix is required. Also, it is useful noting that $\mathbf{B}$ is a $p \times p$ matrix with only $p(p-1)/2$ distinct entries, because of the skew-symmetry.

In order to overcome this representation problem, we propose to recast the learning equations into the following system of differential equations:

$$\begin{cases} \dot{\mathbf{W}} &= \mathbf{V} \,, \\ \dot{\mathbf{V}} &= g(\mathbf{V}, \mathbf{W}) \,, \end{cases} \tag{7}$$

where $\mathbf{V} \in \mathbb{R}^{p\times m}$ replaces $\mathbf{B}$ and $g : \mathbb{R}^{p\times m} \times \mathbb{R}^{p\times m} \to \mathbb{R}^{p\times m}$ is describing the dynamics of the considered rigid-body mechanical system.

It is worth noting that the new formulation of the equations is completely advantageous only if they are integrated numerically in a proper and efficient way. A twofold explanation of this statement is given here below.

- **Preservation of the underlying structure**. The rigid-body dynamics differential equations should be integrated in a way that preserves the rigidity of the system. This is crucial in order to ensure the quality of the signal processing solution provided by the neural system, and it is important in order to preserve the intrinsic stability of the learning theory. The last point is relevant for the long-time simulation in on-line signal processing tasks. The results of extensive numerical simulations, presented recently in [24], clearly show that the majority of existing algorithms are unable to tackle long signal processing tasks. A reason for this is that the network-state eventually looses the adherence to the manifold of constraints pertaining to the learning theories.

- **Efficiency**: It has been observed (see e.g. [23]) that certain classes of learning algorithms involve rectangular matrices whose column/row ratio (m/p) is quite low. This suggests that an integration method that takes into account the structure of matrix-type expressions involved in the learning equations might achieve contained computational complexity. This is the case of algorithms proposed for example in [6, 9, 10, 15]. In the following, we suggest a low-complexity integration scheme especially adapted for the new formulation of the MEC learning equations of complexity $\mathcal{O}(pm^2)$.

2.3 New equations for the MEC algorithm

With the proposed representation, the learning dynamics is described by the new pair of state-variables $(\mathbf{V}, \mathbf{W})$ representing a generic point on the tangent bundle of the Stiefel manifold $\mathrm{TSt}(p, m, \mathbb{R})$. In order to derive the new equations for these state-variables, we consider the following characterisation of second order differential equations on $\mathrm{St}(p, m, \mathbb{R})$.

Theorem 4 *Any second order differential equation on $St(p, m, \mathbb{R})$ can be ex-*

pressed in the form

$$\begin{array}{rcl} \dot{\mathbf{W}} & = & \mathbf{V} = (\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T)\mathbf{W} \ , \\ \dot{\mathbf{V}} & = & (\mathbf{L}\mathbf{W}^T - \mathbf{W}\mathbf{L}^T)\mathbf{W} + (\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T)\mathbf{V} \ , \end{array} \tag{8}$$

with $\mathbf{G} = \mathbf{V} + \mathbf{W}(-\mathbf{W}^T\mathbf{V}/2 + \mathbf{S})$, $\mathbf{S}$ *arbitrary* $m \times m$ *symmetric matrix, and* $\mathbf{L} = \dot{\mathbf{G}} - \mathbf{G}\mathbf{W}^T\mathbf{G}$.

Proof. It has been proven in [9] that any vector $\mathbf{V}$ tangent to the Stiefel manifold at a point $\mathbf{W}$ can be written in the form

$$\mathbf{V} = (\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T)\mathbf{W} \ , \tag{9}$$

where $\mathbf{G} = \mathbf{V} + \mathbf{W}(-\mathbf{W}^T\mathbf{V}/2 + \mathbf{S})$ and $\mathbf{S}$ is an arbitrary symmetric $m \times m$ matrix. $\mathbf{S}$ can be chosen arbitrarily because of the skew-symmetry of $\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T$. In fact by substituting the expression for $\mathbf{G}$ in $\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T$ we obtain

$$\mathbf{V}\mathbf{W}^T - \frac{\mathbf{W}\mathbf{W}^T\mathbf{V}\mathbf{W}^T}{2} + \mathbf{W}\mathbf{S}\mathbf{W}^T - \mathbf{W}\mathbf{V}^T - \frac{\mathbf{W}\mathbf{V}^T\mathbf{W}\mathbf{W}^T}{2} - \mathbf{W}\mathbf{S}\mathbf{W}^T \ ,$$

which is independent of $\mathbf{S}$.

By differentiating (9) with respect to time we obtain

$$\ddot{\mathbf{W}} = \dot{\mathbf{V}} = (\dot{\mathbf{G}}\mathbf{W}^T + \mathbf{G}\mathbf{V}^T - \mathbf{V}\mathbf{G}^T - \mathbf{W}\dot{\mathbf{G}}^T)\mathbf{W} + (\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T)\mathbf{V}. \tag{10}$$

Now, by multiplying out $\mathbf{V} = (\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T)\mathbf{W}$, and using the property $\mathbf{W}^T\mathbf{W} = \mathbf{I}_m$, we obtain $\mathbf{V} = \mathbf{G} - \mathbf{W}\mathbf{G}^T\mathbf{W}$, which, substituted into the first term of the right hand side of (10), gives

$$\dot{\mathbf{V}} = ((\dot{\mathbf{G}} - \mathbf{G}\mathbf{W}^T\mathbf{G})\mathbf{W}^T - \mathbf{W}(\dot{\mathbf{G}} - \mathbf{G}\mathbf{W}^T\mathbf{G})^T)\mathbf{W} + (\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T)\mathbf{V} \ ,$$

which concludes the proof. □

By comparing the equations (1) and (2) with equation (8), and recalling that $\dot{\mathbf{W}} = \mathbf{V} = \mathbf{B}\mathbf{W}$ which implies $\dot{\mathbf{V}} = \dot{\mathbf{B}}\mathbf{W} + \mathbf{B}\mathbf{V}$, we recognise that $\mathbf{B} = \mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T$, and $\mathbf{L} = \mathbf{F} + \mathbf{P}$.

The matrix $\mathbf{S}$ plays a role in the computation of $\mathbf{G}$, but not in the evaluation of the rigid-body learning equations. In this paper for simplicity we choose $\mathbf{S} = \mathbf{0}$. It is worth noting that other choices for the matrix $\mathbf{S}$ could be useful, e.g. for reasons of numerical stability [9].

The final expressions for the new MEC learning equations is then

$$\left\{ \begin{array}{rcl} \dot{\mathbf{W}} & = & \mathbf{V} \ , \ \mathbf{P} = -\nu\mathbf{V}, \ \mathbf{G} = \mathbf{G}(\mathbf{V}, \mathbf{W}) \stackrel{\text{def}}{=} \mathbf{V} - \frac{1}{2}\mathbf{W}(\mathbf{W}^T\mathbf{V}) \ , \\ \dot{\mathbf{V}} & = & \mathbf{F} + \mathbf{P} - \mathbf{W}(\mathbf{F} + \mathbf{P})^T\mathbf{W} + (\mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T)\mathbf{V} \ . \end{array} \right. \tag{11}$$

In order to limit the computational complexity of the above expressions, it is important to compute the matrix products in the right order. For instance, the function $g(\mathbf{V}, \mathbf{W})$ should be computed as indicated by the priority suggested by the parentheses in the following expression

$$g(\mathbf{V}, \mathbf{W}) = \mathbf{F} + \mathbf{P} - \mathbf{W}((\mathbf{F} + \mathbf{P})^T \mathbf{W} + (\mathbf{G}^T \mathbf{V})) + \mathbf{G}(\mathbf{W}^T \mathbf{V}) \ .$$

In this way, the matrix products involve $p \times m$ and $m \times m$ matrices only, making the complexity burden pertaining to the evaluation of the function $g(\cdot, \cdot)$ of about $10pm^2 + 3pm + \mathcal{O}(m^2)$ floating point operations plus the cost of evaluating $\mathbf{F}$.

3 Integration of the Learning Equations

In order to implement the new MEC algorithm on a computer platform, it is necessary to discretise the learning equations (11) in the time domain. We here use a geometric numerical integrator based on the classical forward Euler method. The method preserves orthogonality in the numerical integration.

3.1 Geometric integration of the learning equations

In the present case, we partition the learning equations solving the equation for $\mathbf{V}$ with a classical forward Euler method and the equation for $\mathbf{W}$ with a Lie group method.

We integrate the differential equation for $\mathbf{W}$ using the Lie-Euler method which advances the numerical solution by using the left transitive action of $\mathrm{O}(n, \mathbb{R})$ on the Stiefel manifold. The action is lifted to the Lie algebra $\mathbf{so}(n, \mathbb{R})$ using the exponential map. This guarantees that $\mathbf{W}_n^T \mathbf{W}_n = \mathbf{I}_m$ for all n. In formulas, we thus get

$$\left\{ \begin{array}{lcl} \mathbf{V}_{n+1} & = & \mathbf{V}_n + hg(\mathbf{V}_n, \mathbf{W}_n) \ , \\ \mathbf{G}_n & = & \mathbf{V}_n - \frac{1}{2}\mathbf{W}_n(\mathbf{W}_n^T \mathbf{V}_n) \ , \\ \mathbf{W}_{n+1} & = & \exp(h(\mathbf{G}_n \mathbf{W}_n^T - \mathbf{W}_n \mathbf{G}_n^T))\mathbf{W}_n \ , \end{array} \right. \tag{12}$$

where h denotes the time step of the numerical integration, $n \geq 0$ denotes the discrete-time index, and proper initial conditions are fixed. In the present article, we always consider $\mathbf{W}_0 = \mathbf{I}_{p \times m}$ and $\mathbf{V}_0 = \mathbf{0}_{p \times m}$. Provided the matrix exponential is computed to machine accuracy, $\mathbf{W}_n \in \mathrm{St}(p, m)$ for all n, whenever $\mathbf{W}_0 \in \mathrm{St}(p, m)$. However the proposed method does not guarantee that $(\mathbf{W}_n, \mathbf{V}_n) \in \mathrm{TSt}(p, m)$ for all n, and the consequences of this have not been investigated yet.

3.2 Efficient computation of the matrix exponential

In this section, we will discuss the importance of the new formulation of the MEC system for deriving efficient implementations of the method (12).

The computation of the matrix exponential is a task that should be treated with care in the implementations of (12). Computing $\exp(\mathbf{A}) \stackrel{\text{def}}{=} \sum_{j=0}^{\infty} \mathbf{A}^j/j!$, for a $p \times p$ matrix $\mathbf{A}$, requires typically $\mathcal{O}(p^3)$ floating point operations complexity. The numerical methods for computing the matrix exponential are in fact either based on factorisations of the matrix $\mathbf{A}$, e.g. reduction to triangular or diagonal form [38], or on the use of the powers of $\mathbf{A}$. Among these are for example techniques based on the Taylor expansion or on the Cayley-Hamilton theorem, like, e.g., the Putzer algorithm ([29], p. 506).

Matrix factorisations and powers of matrices require by themselves $\mathcal{O}(p^3)$ floating point operations implying immediately a similar complexity for the mentioned algorithms (see e.g. [27] for an overview).

The complexity is $\mathcal{O}(p^2 m)$ if instead we want to compute $\exp(\mathbf{A})\mathbf{X}$ where $\mathbf{X}$ is a $p \times m$ matrix. In this case, methods are available which are based on the computation of $\mathbf{AX}$ and the successive powers $\mathbf{A}^j\mathbf{X}$ each one involving $\mathcal{O}(p^2 m)$ floating point operations.

In any event, since the first two equations of (12) require $\mathcal{O}(pm^2)$ floating point operations, one would hope to get the same type of complexity for computing $\mathbf{W}_{n+1}$, instead of $\mathcal{O}(p^3)$ or $\mathcal{O}(p^2 m)$, especially when p is large and much larger than m.

At the same time, it is very important in our context to obtain approximations $\tilde{\mathbf{X}}$ of $\exp(\mathbf{A})\mathbf{X}$ with the crucial property that $\tilde{\mathbf{X}}$ is an element of the Stiefel manifold. In fact, if this requirement is not fulfilled the geometric properties of the method (12) would be compromised. For this reason the use of approximations of $\exp(\mathbf{A})\mathbf{X}$ based on truncated Taylor expansions is not advisable, because in this case the approximation of $\exp(\mathbf{A})$ is not guaranteed to be an orthogonal matrix, and the resulting approximation of $\exp(\mathbf{A})\mathbf{X}$ is not on the Stiefel manifold.

Since in the method (12) the matrix we want to exponentiate has the special form $\mathbf{A} = \mathbf{G}_n\mathbf{W}_n^T - \mathbf{W}_n\mathbf{G}_n^T = [\mathbf{G}_n, -\mathbf{W}_n][\mathbf{W}_n, \mathbf{G}_n]^T$, the computational costs for $\exp(\mathbf{A})\mathbf{X}$ can be further reduced to $\mathcal{O}(pm^2)$ floating point operations.

In fact, in order to compute $\exp(\mathbf{A})\mathbf{X}$ exactly – up to rounding errors –, it is possible to adopt the strategy proposed in [11] and proceed as follows. Let us consider the $2m \times 2m$ matrix defined by $\mathbf{D} \stackrel{\text{def}}{=} [\mathbf{W}_n, \mathbf{G}_n]^T[\mathbf{G}_n, -\mathbf{W}_n]$, and the

analytic function $\phi(z) \stackrel{\text{def}}{=} \frac{e^z-1}{z}$. Then, it can be shown that

$$\exp(\mathbf{A})\mathbf{X} = \mathbf{X} + [\mathbf{G}_n, -\mathbf{W}_n]\phi(\mathbf{D})[\mathbf{W}_n, \mathbf{G}_n]^T\mathbf{X} \ . \tag{13}$$

Under the assumption that m is not too large, $\phi(\mathbf{D})$ is easy to compute exactly (up to rounding errors) in $\mathcal{O}(m^3)$ floating point operations. For this purpose, we can suggest techniques based on diagonalising $\mathbf{D}$ or on the use of the Putzer algorithm. The cost of computing $\exp(\mathbf{A})\mathbf{W}_n$ with this formula is $8pm^2 + pm + \mathcal{O}(m^3)$ floating point operations. This results in an algorithm of $\mathcal{O}(pm^2)$ computational complexity, therefore leading to a considerable advantage in the case that $p \gg m$.

Here we propose a variant of formula (13) particularly suited to the case of Stiefel manifolds. We consider a QR-factorisation of the $p \times (2m)$ matrix $[\mathbf{W}_n, \mathbf{G}_n]$. Since $\mathbf{W}_n$ has orthonormal columns, we have

$$[\mathbf{W}_n, \mathbf{G}_n] = [\mathbf{W}_n, \mathbf{W}_n^\perp] \begin{bmatrix} \mathbf{I} & \mathbf{C} \\ \mathbf{O} & \mathbf{R} \end{bmatrix},$$

and $[\mathbf{W}_n, \mathbf{W}_n^\perp]$ has $2m$ orthonormal columns. Here since $\mathbf{G}_n = \mathbf{W}_n\mathbf{C} + \mathbf{W}_n^\perp\mathbf{R}$ we have $\mathbf{C} = \mathbf{W}_n^T\mathbf{G}_n$ and $\mathbf{W}_n^\perp\mathbf{R}$ is the QR-factorisation of $\mathbf{G}_n - \mathbf{W}_n\mathbf{C}$. To find $\mathbf{C}$, $\mathbf{W}_n^\perp$ and $\mathbf{R}$ we use about $4pm^2 + pm$ floating point operations plus the cost of a QR-factorisation for a $p \times m$ matrix, which we assume requires about pm^2 floating point operations.

Next we see that the QR-factorisation of $[\mathbf{G}_n, -\mathbf{W}_n]$ can be easily obtained from the QR-factorisation of $[\mathbf{W}_n, \mathbf{G}_n]$, in fact

$$[\mathbf{G}_n, -\mathbf{W}_n] = [\mathbf{W}_n, \mathbf{G}_n] \begin{bmatrix} \mathbf{O} & -\mathbf{I} \\ \mathbf{I} & \mathbf{O} \end{bmatrix} = [\mathbf{W}_n, \mathbf{W}_n^\perp] \begin{bmatrix} \mathbf{C} & -\mathbf{I} \\ \mathbf{R} & \mathbf{O} \end{bmatrix} \ .$$

By putting together the two decomposed factors we obtain the following factorisation for $\mathbf{A}$,

$$\mathbf{A} = [\mathbf{W}_n, \mathbf{W}_n^\perp] \begin{bmatrix} \mathbf{C} - \mathbf{C}^T & -\mathbf{R}^T \\ \mathbf{R} & \mathbf{O} \end{bmatrix} [\mathbf{W}_n, \mathbf{W}_n^\perp]^T \ . \tag{14}$$

Using the obtained factorisation we find that

$$\exp(\mathbf{A}) = [\mathbf{W}_n, \mathbf{W}_n^\perp] \exp\left(\begin{bmatrix} \mathbf{C} - \mathbf{C}^T & -\mathbf{R}^T \\ \mathbf{R} & \mathbf{O} \end{bmatrix} \right) [\mathbf{W}_n, \mathbf{W}_n^\perp]^T, \tag{15}$$

in other words we have reduced the computation of the exponential of the $p \times p$ matrix $\mathbf{A}$ to the computation of the exponential of a $2m \times 2m$ skew-symmetric

matrix. The new formula (15) has in general better stability proprieties compared to (13). In fact, the exponentiation of the block skew-symmetric matrix of size $2m$ in (15), is an easier computational task compared to the computation of the analytic function $\phi(\mathbf{D})$, for a non-normal matrix $\mathbf{D}$ in formula (13). This approach implies however the extra cost of computing a QR-factorisation. We can compute $\exp(\mathbf{A})\mathbf{W}_n$ with this formula with about $9pm^2 + pm + \mathcal{O}(m^3)$ floating point operations. We estimate that the total cost of performing one step of the method (12), using (15) to compute the exponential in the third line of the algorithm, is of about $21pm^2 + 6pm + \mathcal{O}(m^3)$ floating point operations.

4 Results of Numerical Experiments

In this section, we consider some computer-based experiments to illustrate the benefits of the new formulation of the MEC equations. In particular, we report the results of numerical experiments performed on synthetic as well as real-world data in order to show the numerical behaviour of the introduced algorithm. In the following experiments, the matrix exponential in the method (12) is implemented by formula (15).

In order to objectively evaluate the computational saving achievable through the use of the new algorithm, we consider for comparison the old version of the MEC algorithm, previously introduced and studied in [22, 23]. The old MEC algorithm was based on equations (1)-(3), which gave rise to the following method

$$\begin{cases} \mathbf{B}_{n+1} &= \mathbf{B}_n + h((\mathbf{F}_n + \mathbf{P}_n)\mathbf{W}_n^T - \mathbf{W}_n(\mathbf{F}_n + \mathbf{P}_n)^T) \ , \\ & \mathbf{P}_n = -\nu \mathbf{B}_n \mathbf{W}_n, \ \ \mathbf{F}_n = \mathbf{F}(\mathbf{W}_n) \ , \\ \mathbf{W}_{n+1} &= \exp(h\mathbf{B}_n)\mathbf{W}_n \ . \end{cases} \tag{16}$$

The exponential, in this case, may be computed through a Taylor series truncated to the third-order term, namely, for a matrix $\mathbf{X} \in \mathbf{so}(p, \mathbb{R})$, we invoke the approximation

$$\exp(\mathbf{X}) \approx \mathbf{I}_p + \mathbf{X} + \frac{1}{2}\mathbf{X}^2 + \frac{1}{6}\mathbf{X}^3 \ . \tag{17}$$

It is important to note that if we first compute an approximation to $\exp(h\mathbf{B}_n)$ through the above formula and then post-multiply by $\mathbf{W}_n$, namely we employ the updating rule

$$\mathbf{W}_{n+1} \approx \left[\mathbf{I}_p + h\mathbf{B}_n + \frac{1}{2}(h\mathbf{B}_n)^2 + \frac{1}{6}(h\mathbf{B}_n)^3\right]\mathbf{W}_n \ , \tag{18}$$

the cost of the learning algorithm is of about $4p^3 + 6p^2 + 4p^2m + 3pm + p$ floating point operations. The above implementation is the one already used in [22]. However, with a careful implementation, that is by computing $\exp(\mathbf{X})\mathbf{Y}$ by finding successively $\mathbf{XY}, \mathbf{X}(\mathbf{XY}), \mathbf{X}(\mathbf{X}^2\mathbf{Y})$, the operation count for the overall algorithm can be reduced to about $8p^2(m+1/4)+8pm$ floating point operations. This approach gives rise to the computationally cheaper updating rule

$$\begin{cases} \mathbf{V^{(1)}}_n = h\mathbf{B}_n\mathbf{W}_n \ , \\ \mathbf{V^{(2)}}_n = \frac{h}{2}\mathbf{B}_n\mathbf{V^{(1)}}_n \ , \\ \mathbf{V^{(3)}}_n = \frac{h}{3}\mathbf{B}_n\mathbf{V^{(2)}}_n \ , \\ \mathbf{W}_{n+1} \approx \mathbf{W}_n + \mathbf{V^{(1)}}_n + \mathbf{V^{(2)}}_n + \mathbf{V^{(3)}}_n \ . \end{cases} \tag{19}$$

The computational complexity of algorithm (16)+(18) is then of the order $\mathcal{O}(p^3)$, and the computational burden pertaining to algorithm (16)+(19) is of order $\mathcal{O}(p^2m)$, while, according to the estimate given in the previous section, the new algorithm (12) has a computational complexity growing as $\mathcal{O}(pm^2)$. We therefore expect the new algorithm to perform significantly better than the old one when $p \gg m$.

We note, moreover, that the third order truncation of the Taylor series produces accurate approximations of $\exp(h\mathbf{B}_n)\mathbf{W}_n$ only for small values of h. Furthermore, the orthogonality constraint $\mathbf{W}_n^T\mathbf{W}_n = \mathbf{I}_m$ is not fulfilled exactly, but only with an error depending on h. Conversely, formula (15) approximates $\exp(h\mathbf{B}_n)\mathbf{W}_n$ to machine accuracy, and also the orthogonality constraint is satisfied with the same precision.

It is worth mentioning that the particular form of the involved matrices suggests the use of the sparse-matrix representation of the equation (14).

In the following numerical experiments, we may consider four performance indices:

- A specific index tied to the numerical problem at hand.
- The potential energy function associated to the problem, computed at each iteration step, that is $U_n \stackrel{\text{def}}{=} U(\mathbf{W}_n)$, and the *kinetic energy* associated to the mechanical system, which is defined by $K_n \stackrel{\text{def}}{=} \frac{1}{2}\text{tr}[\mathbf{V}_n^T\mathbf{V}_n]$. Note that for the old MEC rule (16), the general expression of the kinetic energy simplifies into $K_n = -\frac{1}{2}\text{tr}[\mathbf{B}_n^2]$, which is used for computation[2].
- The floating point operation count provided by MATLAB 5.2 as empirical index of computational complexity.

[2]Note that the matrix $\mathbf{B}_n^2$ is negative semi-definite.

It might be interesting to note that, besides the specific performance indices, that can be defined after a particular learning task has been specified, the neural network's 'kinetic energy' plays the role of a general-purpose indicator of the network's internal state, whose behaviour may be roughly subdivided in three phases: 1) at the beginning of learning the neurons are quiet and the kinetic energy has a low value, which begins to grow as soon as the network starts learning; 2) in the full-learning phase, we may observe an intense internal state of activation, which is testified by high values of the kinetic energy; 3) after that a task has been learnt by the network, the neurons progressively abandon the state of activity, as testified by the progressive loss of network's kinetic energy.

4.1 Largest eigenvalue computation of a sparse matrix

In the first experiment, we want to compare the complexity of the methods (12) and (16), through a purely-numerical experiment.

The proposed experiment consists in finding the largest eigenvalue of the tri-diagonal, symmetric, $p \times p$ matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & \ddots & -1 & \vdots \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{bmatrix} .$$

To this aim we let $\mathbf{W} = \mathbf{w}$ be a vector, and maximise $J(\mathbf{w}) = \mathbf{w}^T \mathbf{A} \mathbf{w}$, that is equivalent to assuming $U(\mathbf{w}) = -J(\mathbf{w})$, under the constraint $\|\mathbf{w}\|_2 = 1$. We use the different formulations of the MEC learning equations to solve such maximisation problem. As initial conditions, we take $\mathbf{w}(0)$ equal to the first canonical vector in $\mathbb{R}^p$, and $\mathbf{V}(0) = \mathbf{v}(0)$ the zero vector. The viscosity parameter is $\nu = 0.5$, and the stepsize of integration is $h = 0.01$. The force corresponding to the chosen potential energy function is $\mathbf{F} = 2\mathbf{A}\mathbf{w}$.

In order to measure the complexity of the considered methods, we performed 100 learning steps. This because the floating point operation count grows linearly with the number of learning steps.

In this experiment, we consider different values of the dimension p of the matrix $\mathbf{A}$, i.e. $p = 2^j$ and $j = 1, \dots, 8$. Table 1 reports the floating point operation count per iteration (i.e. the total floating point operation count pertaining to every experiment divided by 100) for the different approaches, against

SIZE OF $\mathbf{A}$	ALG. (12)	ALG. (16)+(19)	ALG. (16)+(18)
$p = 2$	2.83×10^3	5.27×10^3	4.77×10^4
$p = 4$	9.00×10^3	2.08×10^4	3.54×10^5
$p = 8$	3.21×10^4	8.25×10^4	2.73×10^6
$p = 16$	1.21×10^5	3.29×10^5	2.14×10^7
$p = 32$	4.72×10^5	1.31×10^6	1.69×10^8
$p = 64$	1.86×10^6	5.25×10^6	1.35×10^9
$p = 128$	7.39×10^6	2.10×10^7	—
$p = 256$	2.94×10^7	8.39×10^7	—

Table 1: Computational complexity (in terms of floating point operations per iteration) versus the size of the problem. Comparison of the new and old versions of the MEC algorithm on the first problem.

the increasing value of the problem dimension p. The computational burden of the algorithm (16)+(18) is much higher than the computational complexity of the algorithm (16)+(19), and beyond the order 10^9 we stopped measuring the complexity of the first one to save time. The counted numbers of floating point operations for both these algorithms are higher than the counted number for the new algorithm (12), but the comparison is more striking with algorithm (16)+(18). In order to appreciate the computational gain of algorithm (12) compared to (16)+(19), we repeated the previous experiment in MATLAB 6 considering matrices $\mathbf{A}$ of larger size. In Table 2 we report the CPU times in seconds per iteration for the two methods. The advantage of the new algorithm is clear in the case of $\mathbf{A}$ of large size. The times are obtained averaging over 500 iterations.

4.2 Application to principal subspace analysis

The discussed algorithm has been applied to a statistical signal processing problem, referred to as Principal Subspace Analysis. The description of this problem and the simulation results are reported in the following two subsections.

4.2.1 Principal subspace analysis description

Data reduction techniques aim at providing an efficient representation of the data. We consider the compression procedure consisting in mapping the higher dimensional input space into a lower dimensional representation space by means

SIZE OF $\mathbf{A}$	ALG. (12)	ALG. (16)+(19)
$p = 32$	0.0004	0.0002
$p = 64$	0.0005	0.0005
$p = 128$	0.0014	0.0065
$p = 256$	0.0148	0.0431
$p = 512$	0.0643	0.0825
$p = 1024$	0.1551	0.3074
$p = 2048$	0.6331	1.2443
$p = 4096$	2.9686	68.1223

Table 2: Average CPU time (in seconds) per iteration versus the size of the problem. Comparison of the new and old versions of the MEC algorithm on the first problem.

of a linear transformation, as in the Karhunen-Loéve Transform (KLT). The classical approach for evaluating the KLT requires the computation of the input data covariance matrix, and then the application of a numerical procedure to extract the eigenvalues and the corresponding eigenvectors. Compression is obtained representing the data in the basis of the eigenvectors associated with the dominant eigenvalues. When large data sets are handled, this approach can not be used in practice because the eigenvalues/eigenvectors calculation becomes too onerous. In addition, the whole set of eigenvectors has to be evaluated even though only some of them are used.

In order to overcome these problems, neural-network-based approaches can be used. Neural principal component analysis (PCA) is a second-order adaptive statistical data processing technique introduced by Oja [36] that helps removing the second-order correlation among given random processes. In fact, let us consider the stationary multivariate random process $\mathbf{x}(t) \in \mathbb{R}^p$, and suppose its covariance matrix $\mathbf{\Phi} \stackrel{\text{def}}{=} E_{\mathbf{x}}[(\mathbf{x} - E_{\mathbf{x}}[\mathbf{x}])(\mathbf{x} - E_{\mathbf{x}}[\mathbf{x}])^T]$ exists and is bounded. If $\mathbf{\Phi}$ is not diagonal, then the components of $\mathbf{x}(t)$ are statistically correlated. This second-order redundancy may be partially (or completely) removed by computing a linear operator $\mathbf{L}$ such that the new random signal defined by $\mathbf{y}(t) \stackrel{\text{def}}{=} \mathbf{L}^T(\mathbf{x}(t) - E_{\mathbf{x}}[\mathbf{x}]) \in \mathbb{R}^m$ has uncorrelated components, with $m \leq p$ arbitrarily selected. The operator $\mathbf{L}$ is known to be the matrix formed by the eigenvectors of $\mathbf{\Phi}$ corresponding to its dominant eigenvalues [36]. The elements of $\mathbf{y}(t)$ are termed *principal components of* $\mathbf{x}(t)$. Their importance is proportional to the absolute value of the corresponding eigenvalues $\sigma_i^2 \stackrel{\text{def}}{=} E_{\mathbf{x}}[y_i^2]$, which

are arranged in decreasing order ($\sigma_i^2 \geq \sigma_{i+1}^2$).

The data-stream $\mathbf{y}(t)$ represents a compressed version of the data-stream $\mathbf{x}(t)$. At some point the processed (i.e. stored, retrieved, transmitted), reduced-size data-stream will be recovered, that is brought back to its original size. However, the principal-component based data reduction technique is not lossless, thus only an approximation $\hat{\mathbf{x}}(t)$ of the original data-stream may be recovered. As $\mathbf{L}$ is an orthonormal operator, an approximation of $\mathbf{x}(t)$ is given by $\hat{\mathbf{x}}(t) = \mathbf{L}\mathbf{y}(t) + E_{\mathbf{x}}[\mathbf{x}]$. This approximation minimises the reconstruction error $E_{\mathbf{x}}[||\mathbf{x} - \hat{\mathbf{x}}||_2^2]$ which is equal to $\sum_{k=m+1}^{p} \sigma_k^2$.

A simpler – yet interesting – application is principal subspace analysis (PSA), which focuses on the estimation of an (orthonormal) basis of the subspace spanned by the principal eigenvectors, without computing the eigenvectors themselves. The dual case of Minor Subspace Analysis is discussed in details in [24].

To this aim, we may define a criterion U as an Oja's criterion, that is $J(\mathbf{W}) \stackrel{\text{def}}{=} k \cdot \text{tr}[\mathbf{W}^T \mathbf{\Phi} \mathbf{W}]$, where $k > 0$ is a scaling factor. The criterion is maximised under the constraint of orthonormality of the connection matrix $\mathbf{W}$. In real-world applications, the covariance matrix is unknown in advance, thus we may resort to its instantaneous approximation by replacing $\mathbf{\Phi}$ with $\mathbf{x}\mathbf{x}^T$. This also applies when we have a finite number of input samples, in fact the mentioned approximation allows us to avoid occupying memory storage with the whole samples-batch, and the computational effort is also reduced. It is well-known that the learning algorithm performs a kind of low-pass filtering on the input signal, which approximates temporal averaging.

In PSA estimation, the quantity $J(\mathbf{W})$ itself is a valid index of system performance, thus, we assumed $U(\mathbf{W}) = -J(\mathbf{W})$.

4.2.2 Computer-based experiment on PSA

In the experiment on PSA, a synthetic random process $\mathbf{x}$ with $p = 50$ components has been generated. The random process has zero-mean Gaussian statistics with covariance matrix $\mathbf{\Phi} = \frac{1}{2}(\mathbf{H}_p + \mathbf{H}_p^T)$, where $\mathbf{H}_p$ denotes the pth-order Hilbert matrix. In this way $\mathbf{\Phi}$ is symmetric and positive definite. We used the Hilbert matrix to generate the signal's covariance matrix, because it is ill-conditioned, i.e. its eigenvalues are quite spread.

We wish to estimate (an orthonormal basis of) the principal subspace associated to the input signal of dimension $m = 3$, and we suppose to have $2,000$ samples of the input signal available.

In order to iterate with the new discretised MEC algorithm we chose pa-

rameters values $\nu = 0.8$, $k = 0.8$, and $h = 0.05$. The result of the iterations is illustrated in the upper row of Figure 2. The obtained numerical results in the

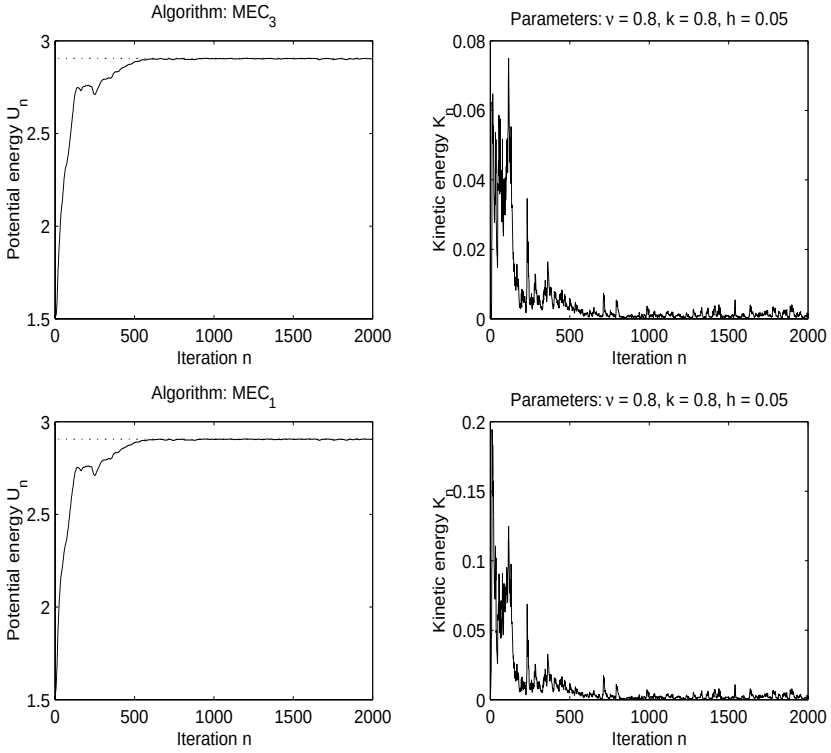

Figure 2: Results of the iteration in the PSA problem. Top: Results pertaining the algorithm (12). Bottom: Results pertaining to the old MEC algorithm (16). Left panels: Behaviour of potential energy function (solid line) compared to its known exact optimal value (dashed line). For graphical convenience, the potential energy function is plotted with reversed sign (that is equivalent to plot the criterion function J). Right panels: Kinetic energy during network's learning.

approximated-covariance (stochastic) case are in excellent agreement with the expected result.

For comparison purpose, the same Figure 2 shows the behaviour of old MEC algorithm (16). The numerical results are quite similar, but the old algorithm achieves them at a larger computational expense, as it is evidenced by the comparative Table 3. In this case, the ratio among the floating point operations consumption for the two algorithms is about 3.4.

Algorithm	Total flops
New MEC (12)	1.95×10^8
Old MEC (16)+(19)	6.65×10^8

Table 3: Complexity comparison of the new and old versions of the MEC algorithm on the PSA problem. The older MEC is considered with the computationally cheapest implementation of the left-action.

4.3 Application to Independent Component Analysis

Also, the discussed algorithm has been applied to a statistical signal processing problem referred to as Independent Component Analysis. The description of this problem and the simulation results are reported in the following two subsections.

4.3.1 Independent component analysis (ICA) description

As another application, we consider a neural-network based statistical signal processing technique that allows to recover unknown signals by processing their observable mixtures, which are the only available data. In particular, under the hypothesis that the source signals to separate out are statistically independent and are mixed by a linear full-rank operator, the neural independent component analysis (ICA) theory may be employed. The idea is to re-mix the observed mixtures in order to make the resulting signals *as independent as possible* [4, 12, 18, 19, 20]. In practice, a suitable measure of statistical dependence is exploited as an optimisation criterion which drives network's learning.

In the following we use the well-known result of [12] stating that a ICA stage can be decomposed into two subsequent stages, o re-whitening stage, and a orthonormal-ICA stage. Therefore the signal $\mathbf{z} = \mathbf{M}^T\mathbf{s}$, at the sensors, can be first standardised, and then *orthonormally separated* by a three-layer network as depicted in Figure 3. Here we assume the source signal stream $\mathbf{s} \in \mathbb{R}^p$, the observed linear mixture stream $\mathbf{z} \in \mathbb{R}^p$, and the mixing matrix $\mathbf{M}^T \in \mathbb{R}^{p\times p}$.

In the following experiments, the aim is to separate out m independent signals from their linear mixtures. A convenient optimisation principle, that leads to extracting the independent components of a multivariate random signal, is the *kurtosis optimisation* rule. We recall that a possible definition of kurtosis of a zero-mean random variable x is

$$kur(x) \stackrel{\text{def}}{=} \frac{E_x[x^4]}{E_x^2[x^2]} \,. \tag{20}$$

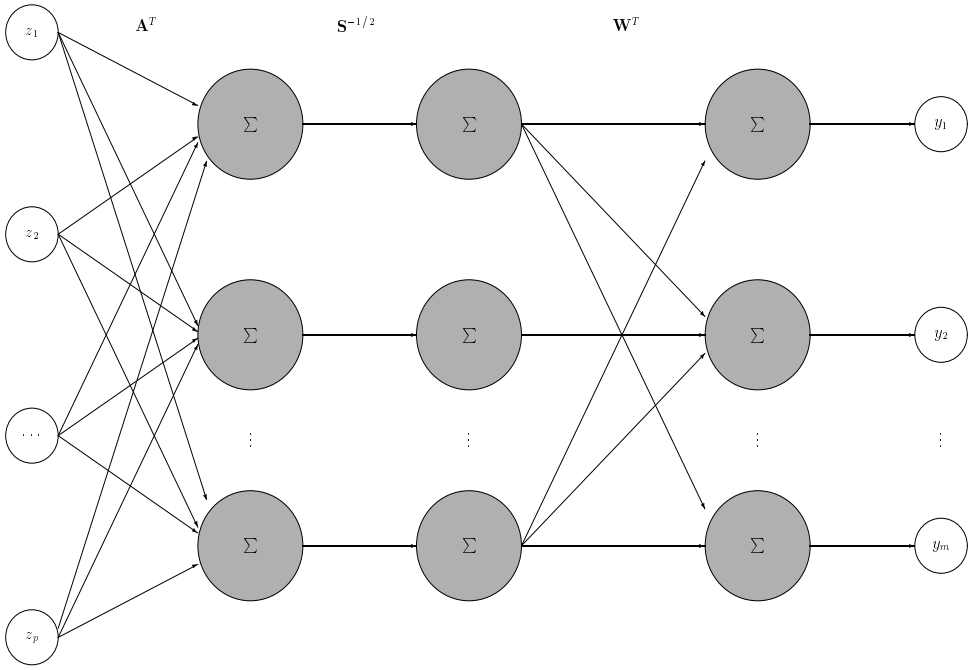

Figure 3: Three-layer neural architecture for blind source separation.

Roughly speaking, the kurtosis is a statistical feature of a random signal that is able to measure its deviation from Gaussianity. A linear combination of random variables tend to approach a Gaussian distribution, while kurtosis optimisation tends to construct variables that deviates from a Gaussian. This briefly gives the rationale of kurtosis extremisation in the context of independent component analysis.

With the above premises, the following simple potential energy function may be used as optimisation criterion, [8, 13],

$$U(\mathbf{W}) = \frac{k}{4} \sum_{i=1}^{m} E_{\mathbf{x}}[y_i^4] \ , \tag{21}$$

where k is a scaling factor. Note that the addenda on the right-hand side of the above expressions are the kurtoses of the network's outputs. In fact, thanks to whitening, it holds $E_{\mathbf{x}}[y_i^2] = 1$ for every i. The resulting active force has the expression

$$\mathbf{F} = -k \cdot E_{\mathbf{x}}[\mathbf{x}(\mathbf{x}^T\mathbf{W})^3] \ , \tag{22}$$

where the $(\cdot)^3$-exponentiation acts component-wise.

The whitening matrix pair $(\mathbf{S}, \mathbf{A})$ computes as follows. If $\mathbf{\Phi}_{zz}$ denotes the covariance matrix of the multivariate random vector $\mathbf{z}$, then $\mathbf{S}$ contains the

eigenvalues, and $\mathbf{A}$ contains (as columns) the corresponding eigenvectors of the covariance matrix. The whitened version of $\mathbf{z}$ is thus $\mathbf{x} = \mathbf{S}^{-1}\mathbf{A}^T\mathbf{z}$.

In the ICA, since the overall source-to-output matrix $\mathbf{Q} \stackrel{\text{def}}{=} \mathbf{W}^T\mathbf{S}^{-1/2}\mathbf{A}^T\mathbf{M}^T \in \mathbb{R}^{m\times p}$ should become as quasi-diagonal (i.e. such that only one entry per row and column differs from zero) as possible, we might take as convergence measure the general Comon time-index [12]. The Comon index measures the distance between the source-to-output separation matrix, and a quasi-identity and is able to measure also degeneracy, that is the case where the same source signals get encoded by two or more neurons.

However, in the present context degeneracy is impossible, because pre-whitening and orthonormal ICA inherently prevent the different neurons from sharing the same source signals. Consequently, we may employ the following reduced criterion, normally referred to as *signal-to-interference ratio*, [22],

$$SIR \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{m}\sum_{j=1}^{p} Q_{ij}^2 - \sum_{i=1}^{m} \max_k\{Q_{ik}^2\}}{\sum_{i=1}^{m} \max_k\{Q_{ik}^2\}} \,. \tag{23}$$

This is a proper measure of distance between $\mathbf{Q}$ and an unspecified quasi-diagonal matrix at any time.

4.3.2 Computer-based experiment on ICA

In the experiment on independent component analysis we considered $p = 6$ grey-scale natural images as source signals. The original images (equipped with their kurtoses) as well as their mixtures are shown in the Figure 4. The images have size 100×100 pixels, thus we have a total of $10,000$ samples. The potential energy function is computed as the (scaled) ensemble-average of the fourth-power of the neural network's output signal. By properly selecting the sign of the constant k it is possible to extract the groups of images from their linear mixtures. In the present case we chose $m = 1$, and $k = 0.5$, so that the algorithm should be able to extract the image that exhibits the smallest kurtosis value in the Figure 4. The other learning parameters were $\nu = 1$, and $h = 0.1$.

Figure 5 shows the value of the performance index SIR, as well as the value of the potential energy function U_n (in units of $k/4$), and of the kinetic energy K_n, during iteration. The same Figure also shows the appearance of the single-neuron's output signal, which closely resembles the image in Figure 4 corresponding to the smallest kurtosis. It is also worth noting that the reached value of the potential energy function U, expressed in units of $k/4$, closely resembles such kurtosis value.

Figure 4: Original images, along with their kurtoses (top six pictures) and their linear mixtures in the ICA problem.

In the present experiment, the size of the involved matrices is not very high, so we expect that the achieved computational gain pertaining to the new MEC with respect to the computationally cheaper one (16)+(19), is not evident – or completely absent. So, we made use of the sparse-matrix representation of the equation (14) which is permitted by the particular form of the involved matrices.

The numerical results provided by the old MEC algorithm (16) on this problem are completely equivalent and are not shown here. The old algorithm achieves them at a larger computational expense, as it is evidenced by the comparative Table 4. In this case, the ratio among the floating point operation consumption between the new old and new version of the MEC algorithm is about 1.25.

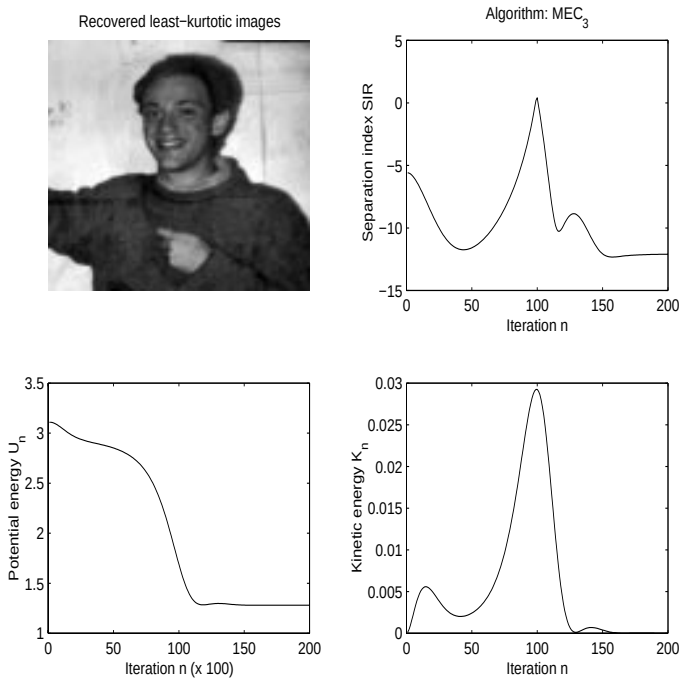

Figure 5: Top-left: Extracted independent component corresponding to the least-kurtotic source. Top-right: Behaviour of separation index SIR during learning. Bottom-left: Behaviour of potential energy function during learning. Bottom-right: Behaviour of kinetic energy during learning.

Algorithm	Total flops
New MEC (12)	6.97×10^4
Old MEC (16)+(19)	8.68×10^4

Table 4: Complexity comparison of the new and old versions of the MEC algorithm on the ICA problem. The new MEC algorithm is implemented by exploiting the sparse-matrix representation of Matlab. The older MEC is considered with the computationally cheapest implementation of the left-action.

5 Conclusion

It is known from the scientific literature that a class of learning algorithms for artificial neural networks may be formulated in terms of matrix-type differential equations of network's learnable parameters. Not infrequently, such differential equations are defined over parameter spaces endowed with a specific geometry (such as the general linear group $\mathrm{GL}(p, \mathbb{R})$, the compact Stiefel manifold or the orthogonal group [2, 21, 23]). Also, from an abstract viewpoint, the differential equations describing the internal dynamics of neural systems may be studied through the instruments of complex dynamical systems analysis. From a practical viewpoint, the mentioned differential equations should be implemented on a computer, so a discretisation method in the time-domain that allows converting them into discrete-time algorithms should be carefully developed, in order to retain (up to reasonable precision) the geometric properties that characterise the developed learning rules.

In previous contributions [21, 22, 23], the second author of this paper presented a new class of learning rules for neural network learning based on the equations describing the dynamics of massive rigid bodies, shortly referred to as MEC. In this context, the parameter space is the real compact Stiefel manifold $\mathrm{St}(p, m, \mathbb{R})$. The main drawbacks of the early formulation were the inefficient representation of the involved quantities, and the inefficient discretisation in the time-domain, which lead to unfaithful conversions between the continuous and the discrete time domains, and an unnecessary computational burden. Both effects evidently appear when there is a serious imbalance between the number of inputs and the number of outputs of the neural systems under analysis.

The aims of this contribution were to introduce a new formulation of the learning equations that provide a better representation of the matrix quantities involved in the MEC learning equations, and to proposed a better numerical integration method for the obtained learning differential equations.

The first result was achieved through Theorem 4, which is the core of the reformulation of the old MEC equations, as it allows parameterising the solution of learning equations in the tangent space to the parameters space instead of the Lie algebra, (namely, it allows getting rid of the angular-speed matrix $\mathbf{B}$ in (1)-(3), to be replaced by the linear-velocity matrix $\mathbf{V}$ in (8)).

The second result was achieved through the use of a geometric integration method. The structure of the involved matrices is exploited and, in particular, the left transitive action of the orthogonal group on the Stiefel manifold,

represented by the third equation of (12), is efficiently implemented.

In order to assess the effectiveness of the proposed enhancements, we tested the new MEC algorithm over three problems, dealing with eigenvector/eigenvalue computation and with statistical signal processing. We also compared the numerical performance of the new algorithm (12), and of the old MEC algorithm (16). The obtained results may be summarized as follows: When the dimensionality of the problem is large enough, the advantages of the new formulation – in terms of computational savings – becomes well apparent.

References

[1] S. Affes and Y. Grenier, *A signal subspace tracking algorithm for speech acquisition and noise reduction with a microphone array*, Proc. of IEEE/IEE Workshop on Signal processing Methods in Multipath Environments, pp. 64 – 73, 1995

[2] S.-i. Amari, *Natural Gradient Works Efficiently in Learning*, Neural Computation, Vol. 10, pp. 251 – 276, 1998

[3] E. Bayro-Corrochano, *Geometric Neural Computation*, IEEE Trans. on Neural Networks, Vol. 12, No. 5, pp. 968 – 986, Sept. 2001

[4] A.J. Bell and T.J. Sejnowski, *An Information Maximisation Approach to Blind Separation and Blind Deconvolution*, Neural Computation, Vol. 7, No. 6, pp. 1129 – 1159, 1995

[5] G.E. Bredon, *Topology and Geometry*, New-York: Springer-Verlag, 1995

[6] T.J. Bridges and S. Reich, *Computing Lyapunov exponents on a Stiefel manifold*, Physica D, Vol. 156, pp. 219 – 238, 2001

[7] R.W. Brockett, *Dynamical Systems that Sort Lists, Diagonalize Matrices and Solve Linear Programming Problems*, Linear Algebra and Its Applications, Vol. 146, pp. 79 – 91, 1991

[8] J.F. Cardoso and B. Laheld, *Equivariant Adaptive Source Separation*, IEEE Trans. on Signal Processing, Vol. 44, No. 12, pp. 3017 – 3030, Dec. 1996

[9] E. Celledoni and B. Owren, *On the implementation of Lie group methods on the Stiefel manifold*, Numerical Algorithms, Vol. 32, pp. 163 – 183, 2003

[10] E. Celledoni and B. Owren, *A class of intrinsic schemes for orthogonal integration*, SIAM Journal on Numerical Analysis, Vol. 21, pp. 463 – 488, 2001

[11] E. Celledoni and A. Iserles, *Approximating the exponential form of a Lie algebra to a Lie group*, Mathematics of Computation, Vol. 69, pp. 1457 – 1480, 2000

[12] P. Comon, *Independent Component Analysis, A New Concept ?*, Signal Processing, Vol. 36, pp. 287 – 314, 1994

[13] P. Comon and E. Moreau, *Improved Contrast Dedicated to Blind Separation in Communications*, Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3453 – 3456, 1997

[14] S. Costa and S. Fiori, *Image Compression Using Principal Component Neural Networks*, Image and Vision Computing Journal (special issue on "Artificial Neural Network for Image Analysis and Computer Vision"), Vol. 19, No. 9-10, pp. 649 – 668, Aug. 2001

[15] L. Dieci and E. Van Vleck, *Computation of orthonormal factors for fundamental solution matrices*, Numerische Mathematik, Vol. 83, pp. 599 – 620, 1999

[16] A. Edelman, T.A. Arias and S.T. Smith, *The Geometry of Algorithms with Orthogonality Constraints*, SIAM Journal on Matrix Analysis Applications, Vol. 20, No. 2, pp. 303 – 353, 1998

[17] Y. Ephraim and L. Van Trees, *A Signal Subspace Approach for Speech Enhancement*, IEEE Trans. on Speech and Audio Processing, Vol. 3, No. 4, pp. 251 – 266, July 1995

[18] S. Fiori, *Entropy Optimization by the PFANN Network: Application to Independent Component Analysis*, Network: Computation in Neural Systems, Vol. 10, No. 2, pp. 171 – 186, May 1999

[19] S. Fiori, *Blind Separation of Circularly-Distributed Sources by Neural Extended APEX Algorithm*, Neurocomputing, Vol. 34, No. 1-4, pp. 239 – 252, Aug. 2000

[20] S. Fiori, *Blind Signal Processing by the Adaptive Activation Function Neurons*, Neural Networks, Vol. 13, No. 6, pp. 597 – 611, Aug. 2000

[21] S. Fiori, *A Theory for Learning by Weight Flow on Stiefel-Grassman Manifold*, Neural Computation, Vol. 13, No. 7, pp. 1625 – 1647, July 2001

[22] S. Fiori, *A Theory for Learning Based on Rigid Bodies Dynamics*, IEEE Trans. on Neural Networks, Vol. 13, No. 3, pp. 521 – 531, May 2002

[23] S. Fiori, *Unsupervised Neural Learning on Lie Group*, International Journal of Neural Systems, Vol. 12, No.s 3 & 4, pp. 219 – 246, 2002

[24] S. Fiori, *A Minor Subspace Algorithm Based on Neural Stiefel Dynamics*, International Journal of Neural Systems, Vol. 12, No. 5, pp. 339 – 350, 2002

[25] A. Fujiwara and S.-i. Amari, *Gradient systems in view of information geometry*, Physica D, Vol. 80, pp. 317 – 327, 1995

[26] K. Gao, M.O. Ahmed, and M.N. Swamy, *A Constrained Anti-Hebbian Learning Algorithm for Total Least-Squares Estimation with Applications to Adaptive FIR and IIR Filtering*, IEEE Trans. on Circuits and Systems – Part II, Vol. 41, No. 11, pp. 718 – 729, Nov. 1994

[27] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The John Hopkins University Press, 1996

[28] E. Hairer, C. Lubich and G. Wanner, *Geometric Numerical Integration*, Springer series in Computational Mathematics, Springer, 2002

[29] R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Reprinted edition, 1995

[30] A. Iserles, H.Z. Munthe-Kaas, S.P. Nørsett and A. Zanna: *Lie-group methods*, Acta Numerica, Vol. 9, pp. 215 – 365, 2000

[31] J. Kivinen and M. Warmuth, *Exponentiated gradient versus gradient descent for linear predictors*, Information and Computation, Vol. 132, pp. 1 – 64, 1997

[32] R.-w. Liu, *Blind Signal Processing: An Introduction*, Proc. of International Symposium on Circuits and Systems (IEEE-ISCAS), Vol. 2, pp. 81 – 84, 1996

[33] B.C. Moore, *Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction*, IEEE Trans. on Automatic Control, Vol. AC-26, No. 1, pp. 17 – 31, 1981

[34] H. Niemann and J.-K. Wu, *Neural Network Adaptive Image Coding*, IEEE Trans. on Neural Networks, Vol. 4, No. 4, pp. 615 – 627, July 1993

[35] Y. Nishimori, *Learning Algorithm for ICA by Geodesic Flows on Orthogonal Group*, Proc. of the International Joint Conference on Neural Networks (IJCNN'99), Vol. 2, pp. 1625 – 1647, 1999

[36] E. Oja, *Neural Networks, Principal Components and Subspaces*, Int. Journal of Neural Systems, Vol. 1, pp. 61 – 68, 1989

[37] E. Oja, A. Hyvärinen and P. Hoyer, *Image Feature Extraction and Denoising by Sparse Coding*, Pattern Analysis and Applications Journal, Vol. 2, Issue 2, pp. 104 – 110, 1999

[38] B. Parlett, *A recurrence among the elements of functions of triangular matrices*, Linear Algebra and its Applications, Vol. 14, pp. 117 – 121, 1976

[39] P. Saisan, G. Doretto, Y.N. Wu and S. Soatto, *Dynamic texture recognition*, Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, pp. 58 – 63, Dec. 2001

[40] D. Sona, A. Sperduti and A. Starita, *Discriminant Pattern Recognition Using Transformation Invariant Neurons*, Neural Computation, Vol. 12, No. 6, pp. 1355 – 1370, June 2000

[41] I.-T. Um, J.-J. Wom and M.-H. Kim, *Independent component based Gaussian mixture model for speaker verification*, Proc. of Second International ICSC Symposium on Neural Computation (NC), pp. 729 – 733, 2000

[42] L. Xu, E. Oja and C.Y. Suen, *Modified Hebbian learning for curve and surface fitting*, Neural Networks, Vol.5, pp. 393 – 407, 1992

[43] B. Yang, *Projection Approximation Subspace Tracking*, IEEE Trans. on Signal Processing, Vol. 43, No. 1, pp. 1247 – 1252, Jan. 1995

[44] H.H. Yang and S.-i. Amari, *Adaptive online learning algorithms for blind separation: maximum entropy and minimal mutual information*, Neural Computation, Vol. 9, pp. 1457 – 1482, 1997

[45] K. Zhang and T.J. Sejnowski, *A theory of geometric constraints on neural activity for natural three-dimensional movement*, Journal of Neuroscience, Vol. 19, No. 8, pp. 3122 – 3145, 1999