An implementation of the steepest descent method using retractions on riemannian manifolds

Ever F. Cruzado Quispe * and Erik A. Papa Quiroz †

Abstract

In 2008 Absil et al. [1] published a book with optimization methods in Riemannian manifolds. The authors developed steepest descent, Newton, trust-region and conjugate gradients methods using an approximation of the geodesic called retraction.

In this paper we present implementations of the of steepest descent method of Absil et al. [1] using Matlab software. We show the implementation and numerical results to minimize the Rayleigh quotient on the unit sphere and the Brockett function on the Stiefel manifold that are strongly related to eigenvalue problems in computational Linear Algebra. Moreover, we introduce nonconvex functions studied in the paper of Papa Quiroz et al. [7], verifying that for these functions the method also has good performance.

Keywords: Optimization on manifolds, riemannian manifolds, retractions, steepest descent method.

1 Introduction

We are interested in studying the problem:

$$\begin{cases} \min f(x) \\ s.to: \\ x \in \mathcal{M} \end{cases} \tag{1.1}$$

where $\mathcal{M}$ is a Riemannian manifold and f is a function defined on $\mathcal{M}$. The problem (1.1) belongs to the branch of Mathematical Optimization on Riemannian manifolds wich arises as a natural extension of the theory and optimization methods in Euclidian space for more general spaces. One of the advantages of using Riemannian geometry tools is that constrained optimization problems can be seen as unconstrained ones considering the intrinsic properties of the manifold; another advantage is that optimization problems with nonconvex objetive functions become convex choosing a appropriate Riemannian metric.
We will study the generalization of steepest descent method from Euclidian spaces to Riemannian manifolds, this method generates the following sequence of poinTS x_k give by:

$$x_{k+1} = R_{x_k}(t_k \eta_k),$$

where $\eta_k \in T_{x_k}\mathcal{M}$ ($T_{x_k}\mathcal{M}$ is the tangent space to the manifold $\mathcal{M}$ in the point x_k), t_k is a scalar for which we will make a Armijo search to determine the next point x^{k+1} and R is a application called **retraction**, this ones is defined as a smooth application of tangent bundle on the manifold, i.e., $R : T\mathcal{M} \to \mathcal{M}$; further if we denote for R_x the restriction of R to $T_x\mathcal{M}$ it must satisfy the following conditions:

*Callao National University, evercmath@gmail.com
†Federal University of Rio de Janeiro, erikpapa@gmail.com

- $R_x(0_x) = x$, where $0_x \in T_x\mathcal{M}$ is the null vector.
- With the canonical identification $T_{0_x}T_x\mathcal{M} \simeq T_x\mathcal{M}$, R_x satisfies $DR_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}$, where, $\mathrm{id}_{T_x\mathcal{M}}$ denotes the identity application on $T_x\mathcal{M}$.

Many key ideas for line search methods were developed by LUEMBERGER (1972) [6], who proposed to use a search direction along geodesic obtained by projecting the gradient vector in $\mathbb{R}^n$ on the tangent space of the set of constraints. Other contributions in optimization on manifolds can be found in GABAY(1982) [5], who generalized Quasi-Newton methods obtaining superlinear convergence, there are also contributions of UDRISTE (1994) [8] and of YANG (2007) [9], who proposed a strategy of Armijo line search along geodesic.
In this work, we study the retraction application of a practical point of view, performing implementations of the steepest descent method on Riemannian manifolds, to minimize the Rayleigh quotient on the unit sphere and to minimize the Brockett function on the Stiefel manifold. Later, we present some computational experiments showing that the proposed method also works for nonconvex functions, wich were taken of the paper of Papa Quiroz et al. [7]. Our main contribution is to implement the respective algorithms using MATLAB.
The paper is divided as follows. In Section 2 we give some basic results of manifolds and retractions that we will use along the paper. In Section 3 we introduce the method of steepest descent. In Section 4 we give some convergence results of the sequence generated by the method. In Section 5 we show the applications with their respective algorithms. Finally, in Section 6 we realize some computational experiments using our method for nonconvex functions of which we also implement their algorithms and then we show the computational results.

2 Basic results

2.1 Differentiable manifolds

The abstract definition of a manifold relies on the concepts of charts and atlases, so first we define these concepts.

Definition 2.1 (Chart) *Let $\mathcal{M}$ be a set, a chart d-dimensional of the set $\mathcal{M}$, is a bijection φ of a subset $\mathcal{U}$ of $\mathcal{M}$ onto a open subset of $\mathbb{R}^d$, i.e.,*

$$\begin{array}{rccc} \varphi: & \mathcal{U} \subset \mathcal{M} & \to & \mathcal{S} \subset \mathbb{R}^d \\ & x & \mapsto & \varphi(x) \end{array}$$

A chart is denoted by $(\mathcal{U}, \varphi)$, if there is no risk of confusion, we will simply write φ.

Definition 2.2 (Atlas) *An atlas C^∞ of $\mathcal{M}$ into $\mathbb{R}^d$ is a collection of charts $\{(\mathcal{U}_\alpha, \varphi_\alpha) : \alpha \in I\}$ of the set $\mathcal{M}$ such that:*

1. $\bigcup_{\alpha \in I} \mathcal{U}_\alpha = \mathcal{M}$,
2. *for any pair $\alpha, \beta \in I$ with $\mathcal{U}_\alpha \cap \mathcal{U}_\beta \neq \emptyset$, the sets $\varphi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)$ and $\varphi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)$ are open sets in $\mathbb{R}^d$ and the change of coordinates*

$$\varphi_\beta \circ \varphi_\alpha^{-1} : \varphi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta) \subset \mathbb{R}^d \to \varphi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta) \subset \mathbb{R}^d$$

is smooth with its inverse also smooth (i.e., is a diffeomorphism).

Two atlases $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent or compatible if $\mathcal{A}_1 \cup \mathcal{A}_2$ is an atlas, in other words, for every chart $(\mathcal{U}, \varphi)$ in $\mathcal{A}_2$, the set of charts $\mathcal{A}_1 \cup \{(\mathcal{U}, \varphi)\}$ is still an atlas.
Given an atlas $\mathcal{A}$, let $\mathcal{A}^+$ be the set of all charts $(\mathcal{U}, \varphi)$ such that $\mathcal{A} \cup \{(\mathcal{U}, \varphi)\}$ is also an atlas; the atlas $\mathcal{A}^+$ is called the maximal atlas or complete generated by the atlas $\mathcal{A}$.
An maximal atlas, is called differentiable structure.

Definition 2.3 (Differentiable manifold) *A differentiable manifold (d-dimensional) is a pair $(\mathcal{M}, \mathcal{A}^+)$, where $\mathcal{M}$ is a set and $\mathcal{A}^+$ is a maximal atlas of $\mathcal{M}$ into $\mathbb{R}^d$, such that the topology induced by $\mathcal{A}^+$ is Hausdorff y second countable.*

2.2 Submanifolds

Let $(\mathcal{M}, \mathcal{A}^+)$ and $(\mathcal{N}, \mathcal{B}^+)$ be two differential manifolds such that $\mathcal{N} \subset \mathcal{M}$. The manifold $(\mathcal{N}, \mathcal{B}^+)$ is called an **immersed submanifold** of $(\mathcal{M}, \mathcal{A}^+)$ if the inclusion application $i : \mathcal{N} \to \mathcal{M}$ defined by $i(x) = x$ is an immersion (see Absil et al. [1], page 24).
Let $(\mathcal{N}, \mathcal{B}^+)$ be a submanifold of $(\mathcal{M}, \mathcal{A}^+)$. Since $\mathcal{M}$ and $\mathcal{N}$ are also topological spaces with the topology of the manifold, if the topology of the manifold $\mathcal{N}$ coincide with the topology of the induced subspace by the topological space $\mathcal{M}$, then $\mathcal{N}$ is called a **embedded submanifold, regular submanifold** or simply **submanifold** of the manifold $\mathcal{M}$.

2.3 Riemannian manifolds

A Riemannian manifold is a differentiable manifold in which every tangent space $T_x\mathcal{M}$ is endowed with an inner product $\langle \cdot, \cdot \rangle_x$ (i.e., a bilinear, Symmetric positive-definite form) and, moreover, this inner product is smoothly varying. This fact allows us to define several metric notions, as length of curves, gradient functions, etc.
The smoothly varying inner product is called **Riemannian metric**. We will use the following notations to denote the inner product of two elements ξ_x and ζ_x of $T_x\mathcal{M}$

$$g(\xi_x, \zeta_x) = g_x(\xi_x, \zeta_x) = \langle \xi_x, \zeta_x \rangle = \langle \xi_x, \zeta_x \rangle_x .$$

Strictly speaking, a Riemannian manifold is a pair $(\mathcal{M}, g)$, where $\mathcal{M}$ is a differentiable manifold and g is a Riemannian metric on $\mathcal{M}$.

Definition 2.4 *Given a smooth scalar function f defined on a Riemannian manifold $\mathcal{M}$, the gradient of f at x denoted by* $\operatorname{grad} f(x)$*, is defined as the unique element of $T_x\mathcal{M}$ that satisfies*

$$\langle \operatorname{grad} f(x), \xi \rangle_x = \mathrm{D}\, f(x)[\xi], \ \forall\, \xi \in T_x\mathcal{M}. \tag{2.2}$$

The gradient of a function has the following remarkable properties of steepest ascent:

- The direction of $\operatorname{grad} f(x)$ is the direction of steepest ascent of f at x:

$$\frac{\operatorname{grad} f(x)}{\|\operatorname{grad} f(x)\|} = \underset{\xi \in T_x\mathcal{M}:\|\xi_x\|=1}{\arg\max}\ \mathrm{D}\, f(x)[\xi].$$

- The norm of $\operatorname{grad} f(x)$ gives the steepest slope of f at x:

$$\|\operatorname{grad} f(x)\| = \mathrm{D}\, f(x) \left[\frac{\operatorname{grad} f(x)}{\|\operatorname{grad} f(x)\|} \right].$$

These two properties are very important in the scope of optimization methods.

2.3.1 Riemannian submanifolds

Let $\mathcal{M}$ be a regular submanifold of a Riemannian submanifold $\overline{\mathcal{M}}$, since every tangent space $T_x\mathcal{M}$ can be considered as a subspace of $T_x\overline{\mathcal{M}}$, the Riemannian metric $\overline{g}$ of $\overline{\mathcal{M}}$ induces a Riemannian metric g on $\mathcal{M}$ according to

$$g_x(\xi,\zeta) = \overline{g}_x(\xi,\zeta);\ \xi,\ \zeta \in T_x\mathcal{M},$$

where ξ and ζ of right-hand side of the equality are view as elements of $T_x\overline{\mathcal{M}}$. These turns to $\mathcal{M}$ into a Riemannain manifold of $\overline{\mathcal{M}}$. Endowed with this Riemannian metric, $\mathcal{M}$ is called Riemannian submanifold of $\overline{\mathcal{M}}$.
The orthogonal complement of $T_x\mathcal{M}$ at $T_x\overline{\mathcal{M}}$ is called the normal space to $\mathcal{M}$ at x and is denoted by $(T_x\mathcal{M})^\perp$:

$$(T_x\mathcal{M})^\perp = \{\xi \in T_x\overline{\mathcal{M}} : \overline{g}_x(\xi,\zeta) = 0,\ \forall\, \zeta \in T_x\mathcal{M}\}.$$

Any element $\xi \in T_x\overline{\mathcal{M}}$ can be uniquely decomposed into the sum of an element of $T_x\mathcal{M}$ and an element of $(T_x\mathcal{M})^\perp$:

$$\xi = P_x\xi + P_x^\perp\xi,$$

where, P_x denotes the orthogonal projection onto $T_x\mathcal{M}$ and $P_x^\perp$ denotes the orthogonal projection onto $(T_x\mathcal{M})^\perp$.

Example 2.1 *The unit sphere S^{n-1} is considered a Riemannian submanifold of $\mathbb{R}^n$, the inner product is inherited from standard inner product at $\mathbb{R}^n$ is given by*

$$\langle \xi, \eta \rangle := \xi^T\eta \tag{2.3}$$

The normal space is $(T_xS^{n-1})^\perp = \{\alpha x : \alpha \in \mathbb{R}\}$, and the projection are given by $P_x\xi = (I - xx^T)\xi$ y $P_x^\perp\xi = xx^T\xi$, for $x \in S^{n-1}$.

Remark 2.1 *Let $\overline{f}$ be a function defined on a Riemannian manifold $\overline{\mathcal{M}}$ and let f denote the restriction of $\overline{f}$ to a Riemannian submanifold $\mathcal{M}$. The gradient of f is equal to the projection of the gradient of $\overline{f}$ onto $T_x\mathcal{M}$, that is,*

$$\operatorname{grad} f(x) = P_x \operatorname{grad} \overline{f}(x). \tag{2.4}$$

In fact, $P_x \operatorname{grad} \overline{f}(x)$ belongs to $T_x\mathcal{M}$ and the equation (2.2) is satisfied since, $\forall \zeta \in T_x\mathcal{M}$ we have:

$$\left\langle P_x \operatorname{grad} \overline{f}(x), \zeta \right\rangle = \left\langle \operatorname{grad} \overline{f}(x) - P_x^\perp \operatorname{grad} \overline{f}(x), \zeta \right\rangle = \left\langle \operatorname{grad} \overline{f}(x), \zeta \right\rangle = D\overline{f}(x)[\zeta] = Df(x)[\zeta].$$

2.4 Retractions

A retraction on a manifold $\mathcal{M}$ is a smooth application R from the tangent bundle $T\mathcal{M}$ onto $\mathcal{M}$ with the following property: if for any $x \in \mathcal{M}$, R_x denote the restriction of R to $T_x\mathcal{M}$, i.e.,

$$\begin{array}{rccc} R_x: & T_x\mathcal{M} & \to & \mathcal{M} \\ & \xi & \mapsto & R_x(\xi), \end{array}$$

then it must satisfy the followings properties:

(i) $R_x(0_x) = x$, where 0_x denotes the zero element of $T_x\mathcal{M}$.

(ii) With the canonical identification $T_{0_x}T_x\mathcal{M} \simeq T_x\mathcal{M}$, R_x satisfies

$$\mathrm{D}\, R_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}, \tag{2.5}$$

where $\mathrm{id}_{T_x\mathcal{M}}$ denotes the identity application on $T_x\mathcal{M}$.

Concerning condition (2.5) notice that, since R_x es an aplication from $T_x\mathcal{M}$ to $\mathcal{M}$ sending 0_x to x, it follows that $\mathrm{D}\, R_x(0_x)$ is an aplication from $T_{0_x}(T_x\mathcal{M})$ to $T_x\mathcal{M}$. Since $T_x\mathcal{M}$ is a vectorial space, there exists a natural identification $T_{0_x}(T_x\mathcal{M}) \simeq T_x\mathcal{M}$. The condition $\mathrm{D}\, R_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}$ is called local rigidity condition.

Proposition 2.1 *Let $\mathcal{M}$ be a regular submanifold of a vector space $\mathcal{E}$ and let $\mathcal{N}$ be an abstract manifold such that $\dim(\mathcal{M}) + \dim(\mathcal{N}) = \dim(\mathcal{E})$. Assume that exists a diffeomorphism*

$$\begin{array}{rccc} \phi: & \mathcal{M}\times\mathcal{N} & \to & \mathcal{E}_* \\ & (F,G) & \mapsto & \phi(F,G) \end{array}$$

where $\mathcal{E}_*$ is an open subset of $\mathcal{E}$ (thus $\mathcal{E}_*$ is an open submanifold of $\mathcal{E}$), with a neutral element $I \in \mathcal{N}$ satisfying $\phi(F, I) = F,\ F \in \mathcal{M}$.
Under the above assumptions on ϕ, the aplication

$$R_X(\mathcal{E}) := \pi_1(\phi^{-1}(X + \xi));$$

where, $\pi_1 : \mathcal{M}\times\mathcal{N} \to \mathcal{M}$ defined as $\pi_1(F, G) = F$ is the proyection onto the first component; defines a retraction on $\mathcal{M}$.

Proof. See Proposition 4.1.2 de Absil et al. (2008), [1]. ■

Example 2.2 (Retraction on the Stiefel manifold) *Consider the Stiefel manifold $\mathrm{St}(p, n) = \{X \in \mathbb{R}^{n\times p} : X^T X = I_p\}$, a retraction is,*

$$R_X(\xi) := \mathrm{qf}(X + \xi),$$

where $\mathrm{qf}(A)$ denotes the Q factor of the decomposition of $A \in \mathbb{R}^{n\times p}_*$; as $A = QR$, where $Q \in \mathrm{St}(p, n)$ and R is an upper triangular $p \times p$ matrix with strictly positive diagonal elements.

Proof. See Example 4.1.3 de Absil et al. (2008), [1]. ■

Example 2.3 (Retracction on the sphere S^{n-1})
Let $\mathcal{M} = S^{n-1}$ and let $\mathcal{N} = \{x \in \mathbb{R} : x > 0\}$, and consider the aplication

$$\begin{array}{rccc} \phi: & \mathcal{M}\times\mathcal{N} & \to & \mathbb{R}^n_* \\ & (x,r) & \mapsto & xr. \end{array}$$

It is straightforward to verify that ϕ is a diffeomorphism; then by the Proposition 2.1 is obtained the retraction

$$R_x(\xi) = \frac{x+\xi}{\|x+\xi\|},$$

defined for all $\xi \in T_x S^{n-1}$. The point $R_x(\xi)$ is the point of S^{n-1} that minimizes the distance to $x + \xi$.

3 Steepest descent method

This method on Riemannian manifolds is based on the iteration formula

$$x_{k+1} = R_{x_k}(t_k \eta_k),$$

where $\eta_k \in T_{x_k}\mathcal{M}$, t_k is a scalar and R_{x_k} is a retraction application. To obtain the global convergence of the method we should impose some conditions on η_k and t_k.

Definition 3.1 (Gradient related sequence) *Given a function f on a Riemannian manifold $\mathcal{M}$, a sequence $\{\eta_k\}$, $\eta_k \in T_{x_k}\mathcal{M}$, is gradient related if, for any subsequence $\{x_k\}_{k\in\mathcal{K}}$ de $\{x_k\}$ that converges to a noncritic point of f, the correspondence subsequence $\{\eta_k\}_{k\in\mathcal{K}}$ is bounded and satisfies*

$$\limsup_{k\to\infty,\, k\in\mathcal{K}} \langle \operatorname{grad} f(x_k), \eta_k \rangle < 0.$$

Definition 3.2 (Armijo point) *Given a function f on a Riemannian manifold $\mathcal{M}$ with retraction R, a point $x \in \mathcal{M}$, a tangent vector $\eta \in T_x\mathcal{M}$, and scalars $\overline{\alpha} > 0$, β, $\sigma \in (0,1)$, the Armijo point is $\eta^A = t^A\eta = \beta^m\overline{\alpha}\eta$, where m is the smallest nonnegative integer such that*

$$f(x) - f(R_x(\beta^m\overline{\alpha}\eta)) \geq -\sigma \langle \operatorname{grad} f(x), \beta^m\overline{\alpha}\eta \rangle_x .$$

The real t^A is the Armijo step size.

Now we propose the algorithm 1 that describe a Armijo line search.

Algorithm 1 Armijo line search

Require: A Riemannian manifold $\mathcal{M}$, a continuously differentiable function f on $\mathcal{M}$, a retraction R from $T\mathcal{M}$ to $\mathcal{M}$, scalars $\overline{\alpha} > 0; c$, β, $\sigma \in (0,1)$ and a initial point $x_0 \in \mathcal{M}$.

Ensure: Sequence of iterates $\{x_k\}$.

1: **for** $k = 0, 1, 2, \ldots$ **do**

2: Pick $\eta_k \in T_{x_k}\mathcal{M}$ such that the sequence $\{\eta_k\}_{k=0,1,\ldots}$ is gradient related.

3: Select x_{k+1} such that

$$f(x_k) - f(x_{k+1}) \geq c(f(x_k) - f(R_{x_k}(t_k^A\eta_k))), \tag{3.6}$$

where, t_k^A is the Armijo step size for the given $\overline{\alpha}$, β, σ and η_k.

4: **end for**

4 Convergence analysis

First, we define the notions of convergence and limit points on manifolds, then we give a global convergence result for Algorithm 1.

4.1 Convergence on manifolds

The notion of convergence on manifolds is a generalization of the $\mathbb{R}^n$ case.

Definition 4.1 *An infinite sequence $\{x_k\}_{k=0,1,\ldots}$ of points of a manifold $\mathcal{M}$ is said to be convergent if there exists a chart $(\mathcal{U}, \psi)$ of $\mathcal{M}$, a point $x_* \in \mathcal{U}$, and a $k > 0$ such that x_k is in $\mathcal{U}$ for all $k \geq K$ and such that the sequence $\{\psi(x_k)\}_{k=K,K+1,\ldots}$ converges to $\psi(x_*)$.*

Definition 4.2 *Given a sequence $\{x_k\}_{k=0,1,...}$ we say that x is an accumulation point or limit point of the sequence, if there exists a subsequence $\{x_{j_k}\}_{k=0,1,...}$ that converges a x. The set of accumulation points of a sequence is called the limit set of the sequence.*

4.2 Convergence of the method

Theorem 4.1 *Let $\{x_k\}$ be an infinite sequence of iterations generated by Algorithm 1. Then every accumulation point $\{x_k\}$ is a critical point of the cost function f.*

Proof. See Theorem 4.3.1 de Absil et al. (2008), [1]. ■

Corollary 4.1 *Let $\{x_k\}$ be an infinite sequence of iterations generated by Algorithm 1. Assume that the level set $\mathcal{L} = \{x \in \mathcal{M} : f(x) \leq f(x_0)\}$ is compact (which holds in particular when $\mathcal{M}$ itself is compact). Then $\lim_{k\to\infty} \|\text{grad}\, f(x_k)\| = 0$.*

Proof. See Corollary 4.3.2 de Absil et al. (2008), [1]. ■

4.3 Stability of fixed points

The Theorem 4.1 states that only critical points of a cost function f can be accumulation points of sequences $\{x_k\}$ generated by Algorithm 1; but it does not specify whether the acumulation points are local minimizers, local maximizers or saddle points.
In practice, if a initial point x_0 is carefully crafted, the Algoritm 1 generates sequences whose acumulation points are local minimal of the cost function. This observation is supported by the following stability analysis of critical points.

Definitions 4.3

1. *Let F be a application from $\mathcal{M}$ a $\mathcal{M}$. A point $x_* \in \mathcal{M}$ is a fixed point of F if $F(x_*) = x_*$. Let $F^{(n)}$ denote the result of n applications of F to x, i.e.,*

$$F^{(1)}(x) = F(x),\ F^{(i+1)}(x) = F(F^{(i)}(x)),\ i = 1, 2, ...$$

2. *A fixed point x_* of F is a stable point of F if, for every neighborhood $\mathcal{U}$ of x_*, there exists a neighborhood $\mathcal{V}$ of x_* such that, for all $x \in \mathcal{V}$ and all positive integer n, it holds that $F^{(n)}(x) \in \mathcal{U}$.*

3. *The fixed point x_* is asymptotically stable if it is stable, and, moreover $\lim_{n\to\infty} F^{(n)}(x) = x_*$ for all x sufficiently close to x_*.*

4. *The fixed point x_* is unstable if it is not stable; in other words, there exists a neighborhood $\mathcal{U}$ of x_* such that, for all neighborhood $\mathcal{V}$ of x_*, there is a point $x \in \mathcal{V}$ such that $F^{(n)}(x) \notin \mathcal{U}$ for some n.*

5. *We say that F is a descent application for a cost function f if*

$$f(F(x)) \leq f(x),\ \forall x \in \mathcal{M}.$$

Theorem 4.2 (Unstable fixed points) *Let $F : \mathcal{M} \to \mathcal{M}$ be a descent application for a smooth cost function f and assume that for every $x \in \mathcal{M}$, all the acumulation points of $\{F^{(k)}(x)\}_{k=1,2,\dots}$ are critical points of f. Let x_* be a fixed point of F (thus x_* is a critical point of f). Assume that x_* is not a local minimum of f. Further assume that there is a compact neighborhood $\mathcal{U}$ of x_* where, for every critical point y of F in $\mathcal{U}$, $f(y) = f(x_*)$. Then x_* is an unstable point for F.*

Proof. See Theorem 4.4.1 de Absil et al. (2008), [1]. ■

We now give a stability result.

Theorem 4.3 (Capture theorem) *Let $F : \mathcal{M} \to \mathcal{M}$ be a descent application for a smooth cost function f and assume that, for every $x \in \mathcal{M}$, all the acumulation points of $\{F^k(x)\}_{k=1,2,\dots}$ are critical points of f. Let x_* be a local minimizer and an isolated critical point of f. Assume further that* $\operatorname{dist}(F(x), x)$ *goes to zero as x goes to x_*. Then x_* is an asymptotically stable point of F.*

Proof. See Theorem 4.4.2 de Absil et al. (2008), [1]. ■

5 Applications

5.1 Rayleigh quotient minimization on the unit sphere

Consider the function

$$\begin{array}{rccl} \overline{f}: & \mathbb{R}^n & \to & \mathbb{R} \\ & x & \mapsto & \overline{f}(x) = x^T A x \end{array}$$

whose restriction to the unit sphere S^{n-1} is the function

$$\begin{array}{rccl} f: & S^{n-1} & \to & \mathbb{R} \\ & x & \mapsto & f(x) = x^T A x, \end{array} \tag{5.7}$$

where the matrix A is symmetric ($A = A^T$) but not necessarily positive-definite.
We view S^{n-1} as a Riemannian submanifold of Euclidean space $\mathbb{R}^n$ endowed with the canonical Riemannian metric. $\overline{g}(\xi, \zeta) = \xi^T \zeta$.

The formulas that we use are summarized in the following table.

	Manifold (S^{n-1})	Embedding space ($\mathbb{R}^n$)
function	$f(x) = x^T A x$, $x \in S^{n-1}$	$\overline{f}(x) = x^T A x$, $x \in \mathbb{R}^n$
metric	induced metric	$\overline{g}(\xi, \zeta) = \xi^T \zeta$
tangent space	$\xi \in \mathbb{R}^n : x^T \xi = 0$	$\mathbb{R}^n$
projection onto tangent space	$P_x \xi = (I - x x^T)\xi$	identity
gradient	$\operatorname{grad} f(x) = P_x \operatorname{grad} \overline{f}(x)$	$\operatorname{grad} \overline{f}(x) = 2Ax$
retracction	$R_x(\xi) = \operatorname{qf}(x + \xi)$	$R_x(\xi) = x + \xi$

Hereafter we consider that $\eta_k := -\operatorname{grad} f(x_k)$, therefore for the Rayleigh quotient we have that $\eta_k = -2(Ax_k - x_k x_k^T A x_k)$. It is clear that this choice of search direction is gradient related.

Next we have to pick a retraction; a reasonable choice is

$$R_x(\xi) = \frac{x+\xi}{\|x+\xi\|} \tag{5.8}$$

We now show the algorithm for the problem of minimizing the Rayleigh quotient on the sphere S^{n-1}; the which is an adaptation of Algorithm 1, considering $c = 1$.

Algorithm 2 Armijo line search for the Rayleigh quotient on S^{n-1}

Require: Symetric matrix A, scalars $\overline{\alpha} > 0$, β, $\sigma \in (0,1)$ and an initial point $x_0 \in \mathbb{R}^n$ such that $\|x_0\| = 1$.

Ensure: Secuence of iterations $\{x_k\}$.

1: **for** $k = 0, 1, 2, ...$ **do**

2: Compute $\eta_k = -2(Ax_k - x_k x_k^T A x_k)$.

3: Find the smallest integer $m \geq 0$ such that

$$f(R_{x_k}(\overline{\alpha}\beta^m \eta_k)) \leq f(x_k) - \sigma\overline{\alpha}\beta^m \eta_k^T \eta_k,$$

with f defined in (5.7) and R defined in (5.8).

4: Use the updating formula $x_{k+1} = R_{x_k}(\overline{\alpha}\beta^m \eta_k)$.

5: **end for**

5.1.1 Implementation of Algorithm 2 in MATLAB

We now present the algorithm in Matlab code that resolves the problem of minimize the Rayleigh quotient on the unit sphere.

```
1- Y=input('\n Enter a vector Y: ');
   % must be a column vector of i components
2- fprintf('\n The norm N of Y is: ')
3- N=norm(Y)
4- fprintf('\n The unit vector X of Y is: ');
5- X=Y/norm(Y)
6- A=input('\n Enter a symetric matrix A: ')
   % must be a matrix of order i*i
7- alpha=1;
8- sigma=0.1;
9- beta=0.5;
10- epsilon=0.00001; % approximation error allowed
11- Nmax=100; % maximum number of iterations to be performed
12- k=1;
13- testArmijo=0;
14- fprintf('\n The initial point X(i) is: % d',X)
   % i varies from 1 until the number of components that have  X
15- while k<=Nmax,
16- fprintf('\n\n ==================');
17- fprintf('\n Iteration N: % d',k)
18- fprintf('\n   ==================\n');
19- fprintf('\n The point X(i) for this iteration is : % d',X)
20- gradf=2*(A*X-X*X'*A*X) % this is the gradient on the manifold
21- Normadf=norm(gradf)
22-    if Normadf >= epsilon
23-       m=0;
24-        factual=X'*A*X
```

```
25-       eta=-2*(A*X-X*X'*A*X); % search direction
26-       R=(X+eta)/norm(X+eta); % this is the taken retraction
27-       fnuevo=R'*A*R ;
28-       dif=factual-fnuevo
29-       t=1;
30-       test=sigma*t*eta'*eta
31-       testArmijo=testArmijo+1;
32-        while dif < test,
33-          m = m+1
34-           testArmijo=testArmijo+1;
35-           t=beta^m*alpha; % Armijo step size
36-           r=(X+t*eta)/norm(X+t*eta)
             % the retraction is updated at each iteration
37-           dif=factual-r'*A*r
38-           test=sigma*t*eta'*eta
39-         end
40-       X=(X+t*eta)/norm(X+t*eta)
          % updating formula for each iteration
41-       fprintf('\n New point x(i) is=: % d',X)
42-       k=k+1;
43-     else
44-       fprintf('\n Error criterion satisfied Norm-gradient=:% d',Normadf)
45-       fprintf('\n The approximate optimum point x(i) is:% d',X)
46-       fprintf('\n The approximate optimum value is:% d', factual)
47-       fprintf('\n Iteration number is:% d',k-1)
48-       fprintf('\n Number of Armijo test:% d',testArmijo)
49-       k=1000000;
50-     end
51- end
52- if k >999999
53-      fprintf('\n Solved the problem: % d')
54- else
55-    fprintf('\n : Iterations number  exceeded to % d',k-1)
56-    fprintf('\n Number of Armijo test: % d',testArmijo)
57- end
```

Remark 5.1 *The numbers before each line of the algorithm were placed for didactic questions, so for run the algorithm in Matlab only must to copy the Matlab code but not the number.*

We now will make a brief comment on the performance of the algorithm, line 1 to 9 are the input data as the initial point and the parameters used in the Armijo test, Normadf denotes the gradient norm since this is the stop criterion and can see that if the gradient norm is greater than the approximation error, then the algorithm performs the first iteration, where factual denotes the function evaluated at the initial point, thereafter the Armijo test is performed for it always considers $m = 0$ but if not satisfies the test is increased at one unit until that the test is satisfied. Once that the Armijo test is satisfied, the retraction "r" leads to the next point and with this point is realized the second iteration and later the algorithm is repeated until that the stop condition is satisfied.

Now we will show the computational results of applying the algorithm implemented in MATLAB to solve the problem (5.7) for the particular case when

$$A = \begin{bmatrix} 2 & 5 \\ 5 & 1 \end{bmatrix},$$

In Algorithm 2 are considered $\sigma = 0.1$, $\beta = 0.5$, $\overline{\alpha} = 1$, also we consider an approximation error $\epsilon = 0.00001$ and as initial point we take $\begin{bmatrix} 0.6 & 0.8 \end{bmatrix}^T$ which is the unit vector obtained from $\begin{bmatrix} 3 & 4 \end{bmatrix}^T$. With these data the computational results are presented in the following table.

x_k	Iteration	Obtained point	Obtained value	$\|\text{grad}\, f(x_k)\|$
x_0	1	(0.600000, 0.800000)	6.160000	3.760000
x_1	2	(-0.618911, 0.785461)	-3.478254	1.366731
x_2	3	(-0.683516, 0.729935)	-3.522032	0.341732
x_3	4	(-0.667774, 0.744364)	-3.524748	0.087431
x_4	5	(-0.671831, 0.740704)	-3.524925	0.022401
x_5	6	(-0.670794, 0.741644)	-3.524937	0.005740
x_6	7	(-0.671060, 0.741403)	-3.524938	0.001471
x_7	8	(-0.670991, 0.741465)	-3.524938	3.7685e-004
x_8	9	(-0.671009, 0.741449)	-3.524938	9.6562e-005
x_9	10	(-0.671004, 0.741453)	-3.524938	2.4743e-005
x_{10}	11	(-0.671006, 0.741452)	-3.524938	6.3399e-006

In the table we can see that taking the initial point $x_0 = (0.6, 0.8)$, it is need to perform 11 iterations for reach the optimal point, being the optimal point $x^* = (-0.671006, 0.741452)$ and the optimal value $f^* = f(x^*) = -3.524938$.

5.2 Brockett function minimization on the Stiefel manifold

We will minimize the following function, which admits a more friendly expression in matrix form:

$$\begin{array}{rccl} f: & \text{St}(p,n) & \to & \mathbb{R} \\ & X & \mapsto & \text{tr}(X^T A X N), \end{array} \tag{5.9}$$

where $N = \text{diag}(\mu_1, ..., \mu_p)$, with $0 \leq \mu_1 \leq ... \leq \mu_p$, and $\text{St}(p,n)$ denotes the orthogonal Stiefel manifold

$$\text{St}(p,n) = \{X \in \mathbb{R}^{n \times p} : X^T X = I_p\}.$$

We view $\text{St}(p,n)$ as a regular manifold of the Euclidean space $\mathbb{R}^{n \times p}$ endowed with the canonical inner product $\langle Z_1, Z_2 \rangle := \text{tr}\left(Z_1^T Z_2\right)$.
The necessary formulas are summarized in the following table.

	Manifold (S^{n-1})	Embedding space ($\mathbb{R}^n$)
cost function	$\text{tr}(X^T A X N)$, $X^T X = I_p$	$\text{tr}(X^T A X N)$, $X \in \mathbb{R}^{n \times p}$
metric	induced metric	$\langle Z_1, Z_2 \rangle = \text{tr}(Z_1^T Z_2)$
tangent space	$Z \in \mathbb{R}^{n \times p} : \text{sim}(X^T Z) = 0$	$\mathbb{R}^{n \times p}$
projection onto the tangent space	$P_X Z = Z - X\,\text{sim}(X^T Z)$	identity
gradient	$\text{grad}\, f(x) = P_x \,\text{grad}\, \overline{f}(X)$	$\text{grad}\, \overline{f}(x) = 2AXN$
retraction	$R_X(\xi) = \text{qf}(X + Z)$	$R_X(Z) = X + Z$

For the implementation we consider that

$$\eta_k = -\,\text{grad}\, f(X) = -(2AXN - XX^T AXN - XNX^T AX)$$

and we choose the proposed retraction of the example 2.2, i.e., $R_X(\xi) = \mathrm{qf}(X + \xi)$.

5.2.1 Implementation

The implementation is similar to the implementation of the Rayleigh quotient on the unit sphere, only we make the following changes.
Delete Line 1 until line 6 and replace by:

```
M=input('\n Enter a matrix M: ')
% must be a matrix of order n*p
[Q,R]=qr( M )% is the decomposition QR of M
A=input('\n Enter a symetric matrix A: ')
% must be a matrix of order n*n
X = Q;
% this is the orthogonal matrix that we use as initial point
u=input('\n Enter a vector u such that 0<= u_1 <=...<=u_p: ')
% must be a vector of order 1*p such that 0<= u_1 <=...<=u_p
N=diag(u)
```

Line 20 replace by:

```
gradf=2*(A*X*N)-(X*X'*A*X*N)-(X*N*X'*A*X)
```

Line 24 until 27 to delete and replace by:

```
factual=trace(X'*A*X*N)
eta=-2*(A*X*N)-(X*X'*A*X*N)-(X*N*X'*A*X); % this is the search direction
[S,T]=qr(X+eta);
Re=S % this is taken retraction
fnuevo=trace(Re'*A*Re*N)
```

Line 36 and 37 replace by:

```
[W,Z]=qr(X+t*eta)
r = W % the retraction is updated in each iteration
dif=factual-trace(r'*A*r*N)
```

Line 40 replace by:

```
[W,Z]=qr(X+t*eta);
X = W
```

Now we show the computational results that were obtained by applying the algorithm to solve the problem of minimize

$$\begin{array}{rccl} f: & \mathrm{St}(p,n) & \to & \mathbb{R} \\ & X & \mapsto & \mathrm{tr}(X^T A X N), \end{array}$$

for $A = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}$ *and* $N = \begin{bmatrix} 4 & 0 \\ 0 & 5 \end{bmatrix}$. Moreover, we consider as initial point to the orthogonal matrix $\begin{bmatrix} 0.8 & 0.6 \\ 0.6 & -0.8 \end{bmatrix}$ that were obtained of the decomposition QR of the matrix $\begin{bmatrix} 4 & -5 \\ 3 & -4 \end{bmatrix}$. For the necessary input data in Algorithm 1 are consider $\sigma = 0.1$, $\beta = 0.5$, $\overline{\alpha} = 1$, also, we consider an approximation error $\epsilon = 0.00001$.

x_k	N iter.	Obtained point	Obtained value	$\|\text{grad}\, f(x_k)\|$
x_0	1	$\begin{bmatrix} 0.8 & 0.6 \\ 0.6 & -0.8 \end{bmatrix}$	31.920000	1.440000
x_1	2	$\begin{bmatrix} -4.421817e-01 & -8.969255e-01 \\ -8.969255e-01 & 4.421817e-01 \end{bmatrix}$	30.586574	1.189812
x_2	3	$\begin{bmatrix} 1.743662e-01 & 9.846809e-01 \\ 9.846809e-01 & -1.743662e-01 \end{bmatrix}$	30.586600	1.189800
x_3	4	$\begin{bmatrix} 1.743662e-001 & 9.846809e-001 \\ 9.846809e-001 & -1.743662e-001 \end{bmatrix}$	30.091200	0.515100
x_4	5	$\begin{bmatrix} -6.276994e-002 & -9.980280e-001 \\ -9.980280e-001 & 6.276994e-002 \end{bmatrix}$	30.011800	0.187900
x_5	6	$\begin{bmatrix} 2.229207e-002 & 9.997515e-001 \\ 9.997515e-001 & -2.229207e-002 \end{bmatrix}$	30.001500	0.066900
x_6	7	$\begin{bmatrix} -7.902974e-003 & -9.999688e-001 \\ -9.999688e-001 & 7.902974e-003 \end{bmatrix}$	30.000200	0.023700
x_7	8	$\begin{bmatrix} 2.801143e-003 & 9.999961e-001 \\ 9.999961e-001 & -2.801143e-003 \end{bmatrix}$	30.000000	0.008400
x_8	9	$\begin{bmatrix} -9.928140e-004 & -9.999995e-001 \\ -9.999995e-001 & 9.928140e-004 \end{bmatrix}$	30.000000	0.003000
x_9	10	$\begin{bmatrix} 3.518836e-004 & 9.999999e-001 \\ 9.999999e-001 & -3.518836e-004 \end{bmatrix}$	30.000000	0.001100
x_{10}	11	$\begin{bmatrix} -1.247183e-004 & -1.000000e+000 \\ -1.000000e+000 & 1.247183e-004 \end{bmatrix}$	30.000000	3.7415e-004
x_{11}	12	$\begin{bmatrix} 4.420394e-005 & 1.000000e+000 \\ 1.000000e+000 & -4.420394e-005 \end{bmatrix}$	30.000000	1.3261e-004
x_{12}	13	$\begin{bmatrix} -1.566722e-005 & -1.000000e+000 \\ -1.000000e+000 & 1.566722e-005 \end{bmatrix}$	30.000000	4.7002e-005
x_{13}	14	$\begin{bmatrix} 5.552938e-006 & 1.000000e+000 \\ 1.000000e+000 & -5.552938e-006 \end{bmatrix}$	30.000000	1.6659e-005
x_{14}	15	$\begin{bmatrix} -1.968130e-006 & -1.000000e+000 \\ -1.000000e+000 & 1.968130e-006 \end{bmatrix}$	30.000000	5.9044e-006

6 Computational experiments

In this section we give some computational experiments for minimize the following functions:

1. $f(x) = \sqrt{-\log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)}$
2. $f(x) = \log\left(1-\log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)\right)$
3. $f(x) = \arctan\left(-\log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)\right)$,

which were considered in PAPA QUIROZ et al. [7] and have been solved on the interior of the unitary hypercube, considered as a connected and complete Riemannian manifold $\left((0,1)^n, X^{-2}(I-X)^{-2}\right)$. For details see [7].

For our computational experiments we minimize the given functions on the unitary sphere using the method developed in this work, i.e.,

$$\min\{f(x) : x \in S^{n-1}\},$$

For the which we develop an analogous procedure to that made to minimize the quotient Rayleigh on the unitary sphere.

6.1 Experiment 1

In this experiment we minimize the following function

$$\begin{array}{rccl} \overline{f}: & (0,1)^2 & \to & \mathbb{R} \\ & x & \mapsto & \overline{f}(x) = \sqrt{-\log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)} \end{array}$$

restricted to the unitary disc $S^1 = \{(x_1, x_2) : x_1^2 + x_2^2 = 1\}$, except the points where the function not is defined, i.e., we minimize the function

$$\begin{array}{rccl} f: & S^1 - Z & \to & \mathbb{R} \\ & x & \mapsto & f(x) = \sqrt{-\log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)}, \end{array}$$

where $Z = \{(0,1),(0,-1),(1,0),(-1,0)\}$; for which compute the gradient of f and the retraction to use. From (2.4), we have that:

$$\operatorname{grad} f(x) = P_x \operatorname{grad} \overline{f}(x),$$

further, we know that

$$P_x\xi = \xi - xx^T\xi,$$

then the gradient of f will be

$$\operatorname{grad} f(x) = P_x\nabla\overline{f}(x) = [\nabla\overline{f}(x)]^T - xx^T[\nabla\overline{f}(x)]^T, \tag{6.10}$$

where $\nabla\overline{f}(x)$ is the classic gradient of $\overline{f}$.
The retraction that we use will be the same that we used for minimize the Rayleigh quotient, i.e.,

$$R_x\xi = \frac{x+\xi}{\|x+\xi\|}.$$

Implementation of the algorithm for Experiment 1

The implementation will be the same that we made for minimize Rayleigh quotient, only we must make the following changes:
The line 6 must be delete, since that in this example no matrix is required.
Line 8 replace by:

```
sigma=0.5;
```

Line 20 replace by:

```
D1f=(-(1-2*X(1))*(X(2)-X(2)^2))/(2*(sqrt(-log10(X(1)*(1-X(1))*
      X(2)*(1-X(2))))*log(10)*(X(1)*(1-X(1))*X(2)*(1-X(2))));
D2f=(-(1-2*X(2))*(X(1)-X(1)^2))/(2*(sqrt(-log10(X(1)*(1-X(1))*
      X(2)*(1-X(2))))*log(10)*(X(1)*(1-X(1))*X(2)*(1-X(2))));
gradf=[D1f D2f]'-X*X'*[D1f D2f]'
```

Lines 24 and 25 must be replace by:

```
factual=sqrt(-log(X(1)*(1-X(1))*X(2)*(1-X(2))))
eta= -gradf % search direction
```

Line 27 replace by:

```
fnuevo=sqrt(-log(R(1)*(1-R(1))*R(2)*(1-R(2))))
```

Line 37 replace by:

```
fn=sqrt(-log(r(1)*(1-r(1))*r(2)*(1-r(2))));
dif=factual-fn
```

Already having the implementation, now we present the numerical results obtained by MATLAB, for the given initial point in the table 1 of [7].

x_0	N It.	N Test	Opt. point	Opt. value	$\|\text{grad}\, f(x_k)\|$
(0.45, 0.51)	14	42	(0.7071043, 0.7071093)	1.169449	7.088473e-006
(0.4, 0.6)	16	48	(0.7071049, 0.7071087)	1.169449	5.397685e-006
(0.1, 0.9)	18	56	(0.7071050, 0.7071086)	1.169449	5.068903e-006
(0.2, 0.3)	16	48	(0.7071049, 0.7071087)	1.169449	5.397685e-006
(0.7, 0.6)	14	42	(0.7071098, 0.7071037)	1.169449	8.706039e-006

Table 1: Numerical results for Experiment 1

6.2 Experiment 2

In this experiment we minimize the function

$$\begin{array}{rccl} \overline{f}: & (0,1)^2 & \to & \mathbb{R} \\ & x & \mapsto & \overline{f}(x) = \log\left(1 - \log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)\right) \end{array}$$

restricted to the unitary disc $S^1 = \{(x_1, x_2) : x_1^2 + x_2^2 = 1\}$, i.e. we minimize the function

$$\begin{array}{rccl} f: & S^1 - Z & \to & \mathbb{R} \\ & x & \mapsto & f(x) = \log\left(1 - \log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)\right). \end{array}$$

where $Z = \{(0,1),(0,-1),(1,0),(-1,0)\}$.
In the same way that in the experiment 1, the gradient of f and the retraction will be:

$$\text{grad}\, f(x) = P_x \nabla \overline{f}(x) = [\nabla \overline{f}(x)]^T - xx^T[\nabla \overline{f}(x)]^T$$

$$R_x\xi = \frac{x+\xi}{\|x+\xi\|}.$$

The implementation for this experiment is the following.

Implementation of the algorithm for Experiment 2

Similarly to the first application, only we must make the following changes.
Line 6 must be deleted.
Line 20 replaced by:

```
v=X(1)*(1-X(1))*X(2)*(1-X(2));
u=-log(v);
du1=(-X(2)*(1-X(2))*(1-2*X(1))*log10(exp(1)))/v;
du2=(-X(1)*(1-X(1))*(1-2*X(2))*log10(exp(1)))/v;
D1f=(du1/(1+u))*log10(exp(1))
D2f=(du2/(1+u))*log10(exp(1))
factual=log10(1+u)
gradf=[D1f D2f]'-X*X'*[D1f D2f]'
```

Lines 24 and 25 must be replace by:

```
factual=log10(1+u)
eta=-gradf % search direction
```

Line 27 replace by :

```
fnuevo=log10(1-log10(R(1)*(1-R(1))*R(2)*(1-R(2))))
```

Line 37 replace by:

```
fn=log10(1-log10(r(1)*(1-r(1))*r(2)*(1-r(2))))
dif=factual-fn
```

Below we present the numerical results for Experiment 2.

x_0	N It.	N Test	Opt. point	Opt. value	$\|\text{grad}\, f(x_k)\|$
(0.45, 0.51)	12	12	(0.7070938, 0.7071197)	0.6179478	9.018082e-006
(0.4, 0.6)	14	14	(0.7070963, 0.7071173)	0.6179478	7.312436e-006
(0.1, 0.9)	15	15	(0.7070988, 0.7071148)	0.6179478	5.588769e-006
(0.2, 0.3)	14	14	(0.7070963, 0.7071173)	0.6179478	7.312436e-006
(0.7, 0.6)	13	13	(0.7071149, 0.7070987)	0.6179478	5.630557e-006

Table 2: Numerical results for the experiment 2

6.3 Experiment 3

In this experiment we minimize the function

$$\begin{array}{rccl} \overline{f}: & (0,1)^n & \to & \mathbb{R} \\ & x & \mapsto & \overline{f}(x) = \arctan\left(-\log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)\right) \end{array}$$

restricted to the unitary disc $S^1 = \{(x_1, x_2) : x_1^2 + x_2^2 = 1\}$, i.e., we minimize the function

$$\begin{array}{rccl} f: & S^1 - Z & \to & \mathbb{R} \\ & x & \mapsto & f(x) = \arctan\left(-\log\left(x_1\left(1-x_1\right)x_2\left(1-x_2\right)\right)\right) \end{array}$$

where $Z = \{(0,1),(0,-1),(1,0),(-1,0)\}$.

The gradient and the retraction are found as in the experiment 1 and 2.

Implementation of the algorithm for Experiment 3

We must be the following changes with respect to the implementation of first application.
Line 6 must be deleted.
Line 20 replace by:

```
  v=X(1)*(1-X(1))*X(2)*(1-X(2));
u= -log(v);
du1=(-X(2)*(1-X(2))*(1-2*X(1))*log10(exp(1)))/v;
du2=(-X(1)*(1-X(1))*(1-2*X(2))*log10(exp(1)))/v;
D1f=du1/(1+u^2)
D2f=du2/(1+u^2)
factual=atan(u)
gradf=[D1f D2f]'-X*X'*[D1f D2f]'
```

Lines 24 and 25 must be replace by:

```
factual=atan(u)
eta= -gradf % search direction
```

Line 27 replace by:

```
fnuevo=atan(-log10(R(1)*(1-R(1))*R(2)*(1-R(2))))
```

Line 37 replace by:

```
fn=atan(-log10(r(1)*(1-r(1))*r(2)*(1-r(2))))
dif=factual-fn
```

The numerical results are the following:

x_0	N It.	N Test	Opt. point	Opt. value	$\|\text{grad}\, f(x_k)\|$
(0.45, 0.51)	15	15	(0.7070973, 0.7071163)	1.263311	5.779599e-006
(0.4, 0.6)	17	17	(0.7070964, 0.7071172)	1.263311	6.322268e-006
(0.1, 0.9)	20	20	(0.7070957, 0.7071178)	1.263311	6.723781e-006
(0.2, 0.3)	17	17	(0.7070964, 0.7071172)	1.263311	6.322268e-006
(0.7, 0.6)	15	15	(0.7071185, 0.7070951)	1.263311	7.135816e-006

Table 3: Numerical results for Experiment 3

6.4 Experiment 4

In this experiment we minimize the function

$$\begin{array}{rccl} \overline{f}: & \mathbb{R}^2 & \to & \mathbb{R} \\ & x & \mapsto & \overline{f}(x) = x_1^2 + (\frac{1}{2})x_2^2 \end{array}$$

restricted to the unitary disc $S^1 = \{(x_1, x_2) : x_1^2 + x_2^2 = 1\}$, i.e., we minimize the function

$$\begin{array}{rccl} f: & S^1 & \to & \mathbb{R} \\ & x & \mapsto & f(x) = x_1^2 + (\frac{1}{2})x_2^2 \end{array}$$

The optimal points of the problem are $(0, 1)$ y $(0, -1)$. The gradient will be found as in the previous experiments and the retraction will be the same, since also we are minimizing on the unitary sphere.

Implementation of the algorithm for Experiment 4

With respect to the implementation of the first application, see Subsection 5.1, we only must make the following changes:
Line 6 must be deleted.
Line 20 replace by:

```
gradf=[2*X(1) X(2)]'-X*X'*[2*X(1) X(2)]'
```

Lines 24 and 25 must be replaced by:

```
factual=(X(1))^2+(1/2)*(X(2))^2
eta=-gradf % search direction
```

Line 27 replace by:

```
fnuevo=(R(1))^2+(1/2)*(R(2))^2
```

Line 37 replace by:

```
fn=(r(1))^2+(1/2)*(r(2))^2
dif=factual-fn
```

The numerical results are the following:

x_0	N It.	N Test	Opt. point	Opt. value	$\|\text{grad}\, f(x_k)\|$
(3, 5)	3	3	(7.077251e-009, 1)	5.000018e-001	7.077251e-009
(0.7; 2.5)	2	2	(6.828649e-006, 1)	5.001800e-001	6.828649e-006
(4, 9.3)	3	3	(7.368479e-012, 1)	5.000000e-001	7.368479e-012
(1.7; 0.6)	5	5	(4.256051e-010, 1)	5.000003e-001	4.256051e-010
(0.7, 0.6)	4	4	(6.155604e-012, 1)	5.000000e-001	6.155604e-012

Table 4: Numerical results for Experiment 4

In the above table x_0 is an arbitrary point; N It. denotes the number of iterations performed to reach the optimal point; N Test denotes the number of Armijo test that performed the algorithm; Opt. point denotes the approximate optimal point, Opt. value denotes the approximate optimal value and $\|\text{grad} f(x_k)\|$ denotes the gradient norm.

7 Acknowledgment

The research of the second author was supported by the National Program of Innovation for the Competitiveness and Productivity-PNICP, ECIP-1-P-041-14, Fincyt-Perú.

References

[1] ABSIL P.-A., MAHONY R. and SEPULCHRE R. *Optimization Algorithms on Matrix Manifolds.* Princeton Univerity Press, Princeton, New Jersey, 2008.

[2] BERTSEKAS D.P. *Nonlinear Programing.* Massachusetts institute of tecnology, second edition, 1999.

[3] BOOTHBY W.M. *An Introduction to Differentiable Manifolds and Riemannian Geometry.* Academic Press, London, 1986.

[4] do CARMO M.P. *Riemannian geometry.* Mathematics: Theory & Applications. Birkhuser Boston Inc., Boston, MA, 1992. Translated from the second Portuguese edition by Francis Flaherty.

[5] GABAY D. *Minimizing a differentiable function over a differentiable manifold. Journal of Optimization Theory and Aplication,* 1982, vol 37, pp. 177-219.

[6] LUENBERGER D.G. *The gradient projection method along geodesics.* Management Science, 1972, Vol 18 n. 1, pp. 620-631.

[7] PAPA QUIROZ E.A., QUISPE E.M. and OLIVEIRA P.R. *Steepest descent method with a generalized Armijo search for quasiconvex functions on riemannian manifolds.* Journal of Mathematical Analysis and Applications, Vol 341, pp. 467-477, 2008.

[8] UDRISTE C. *Convex functions and optimization methods on riemannian manifolds*, volume 297 of Mathematics and its Applications. Kluwer Academic Publishers Group, Dordrecht, 1994.

[9] YANG Y. *Globally convergent optimization algorithms on Riemannian manifolds: Uniform framework for unconstrained and constrained optimization.* J. Optim. Theory Appl., 132(2):245-265, 2007.