APPROXIMATION OF SEQUENCES OF SYMMETRIC MATRICES WITH THE SYMMETRIC RANK-ONE ALGORITHM AND APPLICATIONS

SYLVAIN ARGUILLÈRE, (SYLVAIN.ARGUILLERE@UPMC.FR)
SORBONNE UNIVERSITÉS, UPMC UNIV PARIS 06, CNRS UMR 7598,
LABORATOIRE JACQUES-LOUIS LIONS, F-75005, PARIS, FRANCE

Abstract. The symmetric rank-one update method is well known in optimization for its applications in quasi-Newton algorithms. In particular, Conn, Gould, and Toint proved in 1991 that the matrix sequence resulting from this method approximates the Hessian of the minimized function under a suitable linear-independence assumption. Extending their idea, we prove that symmetric rank-one updates can be used to approximate any sequence of symmetric invertible matrices, which has applications to more general problems, such as the computation of constrained geodesics in shape analysis imaging problems. We also provide numerical simulations for the method and some of these applications.

Introduction. Let d be an integer and $f:\mathbb{R}^d\to\mathbb{R}$ a $\mathcal{C}^2$ function. We consider the problem of minimizing f over $\mathbb{R}^d$. A well-known efficient algorithm to numerically solve this minimization problem is Newton's method: starting at some point x_0, it considers the sequence

$$x_{k+1}=x_k-h_kH(f)_{x_k}^{-1}\nabla f(x_k),$$

with ∇f the gradient of f, $H(f)$ its Hessian, and $h_k>0$ some appropriate step length.

However, very often the Hessian of f is too computationally difficult to compute, leading to the introduction of so-called quasi-Newton methods. The methods define a sequence

$$x_{k+1}=x_k-h_kB_k^{-1}\nabla f(x_k),$$

where (B_k) is a sequence of symmetric matrices such that

$$B_{k+1}(x_{k+1}-x_k)=\nabla f(x_{k+1})-\nabla f(x_k). \tag{0.1}$$

Indeed, since

$$\begin{aligned}\nabla f(x_{k+1})-\nabla f(x_k)&=\left(\int_0^1 H(f)_{x_k+t(x_{k+1}-x_k)}dt\right)(x_{k+1}-x_k)\\&\simeq H(f)_{x_k}(x_{k+1}-x_k),\end{aligned}$$

we get

$$B_{k+1}(x_{k+1}-x_k)\simeq H(f)_{x_k}(x_{k+1}-x_k).$$

It is reasonable to expect B_k to be close to $H(f)_{x_k}$ in the direction $s_k=x_{k+1}-x_k$ (see [4, 5, 7]).

There are many ways to build a matrix sequence (B_k) satisfying (0.1). However, it was proved in [3] and [9] that some of these methods let B_k approximate $H(f)_{x_k}$ in all directions instead of just one, i.e.,

$$\|B_k-H(f)_{x_k}\|\underset{k\to\infty}{\to}0$$

which implies

$$\|B_k - H(f)_{x_*}\| \underset{k\to\infty}{\to} 0,$$

with the additional assumption of uniform linear independence of the sequence

$$s_k = x_{k+1} - x_k,$$

a notion that will be recalled later. In [3] for example, this is proved for the update of B_k by

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k) = A_k s_k, \quad r_k = y_k - B_k s_k, \quad B_{k+1} = B_k + \frac{r_k r_k^T}{r_k^T s_k}, \tag{0.2}$$

with

$$A_k = \int_0^1 H(f)_{x_k + t(x_{k+1} - x_k)} dt.$$

In this paper, our aim is to generalize the approach in [3] by defining the above symmetric rank-one algorithm for any sequence of symmetric matrices (A_k) and vectors (s_k), and to derive a convergence result, opening a wider range of applications.

For instance, if a sequence A_k converges to an invertible matrix A_*, then we can use the above algorithm to approximate the inverse A_*^{-1} of the limit A_*. Indeed, let $(e_0, \ldots, e_{d-1})$ be the canonical vector basis of $\mathbb{R}^d$. We define the sequence (s_k) in $\mathbb{R}^d$ by

$$s_k := A_k e_{k \bmod d}, \quad y_k = A_k^{-1} s_k = e_{k \bmod d}. \tag{0.3}$$

This sequence is uniformly linearly independent, hence the sequence B_k defined by (0.2) will converge to A_*^{-1}. The rate of convergence depends on the dimension d and on the rate of convergence of A_k, but B_k is much easier to compute than A_k^{-1}.

One of the motivating applications is the computation of geodesics constrained to embedded submanifolds of Riemannian spaces. Indeed, to obtain a geodesic between two fixed points of a submanifold, we need to find a converging sequence of maps $t \mapsto \lambda_k(t)$ given implicitly by an equation of the form

$$A_k(t)\lambda_k(t) = c_k(t),$$

where $A_k(t)$ is a convergent sequence of symmetric, positive definite matrices of high dimension (see [1]). The λ_k are Lagrange multipliers induced by the equations of the submanifold. It may be very computationally demanding to solve such a linear system for every time t and every step k. Instead, we can take

$$\lambda_k(t) = B_k(t)c_k(t),$$

with $B_k(t)$ obtained by applying the symmetric rank-one algorithm described in the previous paragraph. This is particularly useful in Shape Spaces, where the studied manifolds have a very high dimension and a very complex metric.[1]

[1]The present article was actually motivated by such a problem appearing in shape analysis, investigated in [1].

This paper is structured as follows. We give the general framework in Section 1, then state the main result after recalling two equivalent definitions of the uniform linear independence of a sequence of vectors in Section 2. Section 3 is dedicated to intermediate results that will, along with notions developed in Section 4, lead to the proof of our theorem. Section 5 presents numerical simulations for approximating random matrices and their inverse. Finally, in Section 6, we describe the shape deformation problem for which the algorithm was introduced and apply the symmetric rank-one update method to some simple examples, comparing it to the classical method described in [1].

1. Notations and symmetric rank-one algorithm. Consider a sequence $(A_k)_{k\in\mathbb{N}}$ of real square symmetric $d \times d$ matrices. Assume that this sequence converges to some matrix A_*, i.e.,

$$\|A_k - A_*\| \underset{k\to\infty}{\to} 0,$$

where $\|\cdot\|$ is the operator norm on the space $M_d(\mathbb{R})$ of $d\times d$ matrices induced by the canonical Euclidean norm $|\cdot|$ on $\mathbb{R}^d$. Then define

$$\eta_{k,l} = \sup_{k\le i\le l} \|A_i - A_k\|, \quad \text{and} \quad \eta_{k,*} = \sup_{i\ge k} \|A_i - A_k\|$$

for all $k \le l \in \mathbb{N}$. Note that

$$\forall k \le l \in \mathbb{N}, \quad \eta_{k,l} \le \eta_{k,*} \qquad \text{and} \qquad \eta_{k,*} \to 0 \quad \text{as} \quad k \to \infty.$$

Now let $(s_k)_{k\in\mathbb{N}}$ be a sequence of vectors in $\mathbb{R}^d$.

The objective is to find a somewhat simple sequence $(B_k)_{k\in\mathbb{N}}$ of symmetric matrices such that $B_k \to A_*$, using only s_k and $y_k = A_k s_k$.

We use the symmetric rank-one update method from [3]. Start with $B_0 = I_d$, the $d \times d$ identity matrix. Define for $k \in \mathbb{N}$

$$y_k = A_k s_k, \quad r_k = (A_k - B_k)s_k = y_k - B_k s_k,$$

and compute

$$B_{k+1} = B_k + \frac{r_k r_k^T}{r_k^T s_k}.$$

Remark: When $r_k^T s_k = 0$, one just skips the update.

2. Main Result. For every k, we have

$$B_{k+1}s_k = B_k s_k + r_k = B_k s_k + y_k - B_k s_k = y_k,$$

so

$$A_k s_k = B_{k+1} s_k.$$

The main idea is that if A_k, A_{k+1}, …, A_{k+m} are not too far from each other (which is the case for k large enough), we expect $B_{k+m}s_{k+i}$ to be close to $A_{k+m}s_{k+i}$ for $i \le m$. Then, if we can extract from every finite subsequence $(s_k, \dots, s_{k+m})$ a vector basis of $\mathbb{R}^d$, we will obtain the desired convergence.

For a more precise statement, we next define the notion of uniform linear independence. The most intuitive and geometric definition is the following.

Definition 1. *Take a sequence $s = (s_k)_{k\in\mathbb{N}}$ of vectors in $\mathbb{R}^d$, $d \in \mathbb{N} \setminus \{0\}$, and let $m \geq d$ be an integer. Then s is said to be m-uniformly linearly independent if there exists some constant $\alpha > 0$, such that for all $k \in \mathbb{N}$, there are d integers $k+1 \leq k_1 < \cdots < k_d \leq k+m$ such that*

$$|\det(s_{k_1}, \dots, s_{k_d})| \geq \alpha |s_{k_1}| \dots |s_{k_d}|.$$

In other words, from every finite segment of (s_k) of length m, we can extract a linear basis $s_{k_1}, \dots, s_{k_d}$ that will, once normalized, form a parallelepiped that does not become flat as k goes to infinity.

Remark: A sufficient condition for $s = (s_k)_{k\in\mathbb{N}}$ to be m-uniformly linearly independent is the following. If the sequence of subsets $(\{s_{k+1}, \dots, s_{k+m}\})_{k\in\mathbb{N}}$ converges to a subset $\{s_{*,1}, \dots, s_{*,m'}\}$, with $m' \leq m$ a positive integer, that generates $\mathbb{R}^d$, then, for some integer k_0 large enough, $(s_{k_0+k})_{k\in\mathbb{N}}$ is m-uniformly linearly independent. This is an obvious consequence of the continuity of the determinant.

Another definition was given in [3] after [8] as follows.

Definition 2. *A sequence $s = (s_k)_{k\in\mathbb{N}}$ of vectors in $\mathbb{R}^d$, d a positive integer, is said to be (m, β)-uniformly linearly independent, where $d \leq m \in \mathbb{N}$ and $\beta > 0$, if for all $k \in \mathbb{N}$, there are d integers $k+1 \leq k_1 < \cdots < k_d \leq k+m$ such that*

$$\left| \lambda \left(\frac{s_{k_1}}{|s_{k_1}|}, \dots, \frac{s_{k_d}}{|s_{k_d}|} \right) \right| \geq \beta,$$

where $\lambda(M)$ is the singular value of the square matrix M of smallest magnitude.

Remark: A sequence $s = (s_k)$ in $\mathbb{R}^d$ is (m, β)-uniformly linearly independent for some $m \geq d$ and $\beta > 0$ if and only it is m-uniformly linearly independent in the sense of Definition 1. Indeed, let $v_1, \dots, v_d \in \mathbb{R}^d$, and denote $V = \left(\frac{v_1}{|v_1|}, \dots, \frac{v_d}{|v_d|} \right)$. If $|\lambda(V)| \geq \beta > 0$, then $\det(V) \geq \beta^d$, which proves the first part of the equivalence. On the other hand, we know that the eigenvalue of V with largest modulus has modulus less than $\sqrt{d} \max_{i=1,\dots,d} \frac{|s_{k_i}|}{|s_{k_i}|} = \sqrt{d}$. Now, assume that $\det(V) \geq \alpha > 0$. Then $|\lambda(V)| \geq \frac{\alpha}{d^{\frac{d-1}{2}}}$, ensuring the second part of the equivalence.

Theorem 1. *Let (A_k), (s_k), (y_k), (r_k) and (B_k) be defined as in Section 1, with (A_k) having a limit A_*. Assume that there exists a constant $c > 0$ such that for every integer k,*

$$|r_k^T s_k| \geq c |r_k| |s_k|.$$

Then, for every $\beta > 0$ such that (s_k) is (m, β)-uniformly linearly independent in the sense of Definition 2, we have for every $k \in \mathbb{N}$ the quantitative estimates

$$\|B_{k+m} - A_*\| \leq \left(1 + \left(\frac{2+c}{c} \right)^{m+1} \right) \frac{\sqrt{d}}{\beta} \eta_{k,*}. \tag{2.1}$$

The next two sections are dedicated to the proof of this theorem.

Remark: The assumption $|r_k^T s_k| \geq c |r_k| |s_k|$ is necessary, as showcased by the following counter-example.

Fix a constant sequence of matrices $A_k = A$ and the uniformly linear independent sequence $s_k = e_{k \mod d}$, with $e_0, \dots, e_{d-1}$ the canonical basis of $\mathbb{R}^d$, and $k \mod d$ the remainder of the Euclidean division of k by d. Assume that the first column of A has a 1 in every entry. Then $r_{ld} = 0$ for every $l \in \mathbb{N}$, hence the update will be skipped every d steps, and the first column of B_k will stay equal to

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

for every k. In particular, B_k does not converge toward A.

So, even though the update happens relatively often, it does not happen often enough to get the desired result.

3. First estimates. In this section, we give upper bounds on

$$\left| (B_{k+m} - A_k) \frac{s_k}{|s_k|} \right|,$$

and deduce estimates on

$$\left| \frac{(B_{k+m} - A_*)x}{|x|} \right|$$

for a particular set of $x \in \mathbb{R}^d$.

PROPOSITION 1. *Let $(A_k)_{k\in\mathbb{N}}$ be a sequence of real symmetric matrices in $M_d(\mathbb{R})$, $d \in \mathbb{N}$, and (s_k) be a sequence in $\mathbb{R}^d$. Define y_k, B_k and r_k as above. Assume that there exists a constant $0 < c \leq 1$ and for every $k \in \mathbb{N}$,*

$$|r_k^T s_k| \geq c |r_k| |s_k|.$$

Then, for every $l \geq k+1$,

$$|(A_k - B_l)s_k| \leq \left(\frac{2+c}{c} \right)^{l-k-1} \eta_{k,l-1} |s_k|.$$

Proof: We prove this inequality by induction on l, with $k \in \mathbb{N}$ fixed. For $l = k+1$, we know that $B_{k+1}s_k = A_k s_k = y_k$, hence

$$|(A_k - B_{k+1})s_k| = 0.$$

We will use the notation

$$IH(l) := \left(\frac{2+c}{c} \right)^{l-k-1} \eta_{k,l-1} |s_k|,$$

where IH stands for Induction Hypothesis. Now, assume the result to be true for some $l \geq k+1$, i.e.,

$$|(A_k - B_l)s_k| \leq \left(\frac{2+c}{c} \right)^{l-k-1} \eta_{k,l-1} |s_k| = IH(l). \tag{3.1}$$

Let us prove that

$$|(A_k - B_{l+1})s_k| \le \left(\frac{2+c}{c}\right)^{l-k} \eta_{k,l}|s_k| = IH(l+1).$$

Note that

$$\begin{aligned}
|(A_k - B_{l+1})s_k| &= |A_k s_k - (B_l + \frac{r_l r_l^T}{r_l^T s_l})s_k| \\
&= |A_k s_k - B_l s_k - \frac{r_l r_l^T s_k}{r_l^T s_l}| \\
&\le |(A_k - B_l)s_k| + \frac{|r_l||r_l^T s_k|}{c|r_l||s_l|} \\
&\le IH(l) + \frac{|r_l^T s_k|}{c|s_l|}.
\end{aligned} \tag{3.2}$$

Let us find a bound for $\frac{|r_l^T s_k|}{c|s_l|}$, the second term of the right-hand side. First we have

$$\begin{aligned}
|r_l^T s_k| &= |y_l^T s_k - s_l^T B_l s_k| \\
&\le |y_l^T s_k - s_l^T y_k| + |s_l^T (y_k - B_l s_k)| \\
&= |y_l^T s_k - s_l^T y_k| + |s_l^T (A_k - B_l)s_k| \\
&\le |y_l^T s_k - s_l^T y_k| + |s_l| IH(l).
\end{aligned}$$

However, since A_l is symmetric and $y_l = A_l s_l$,

$$|y_l^T s_k - s_l^T y_k| = |s_l^T (A_l - A_k)s_k| \le \eta_{k,l}|s_l||s_k|,$$

from which we deduce

$$|r_l^T s_k| \le \eta_{k,l}|s_l||s_k| + IH(l)|s_l|.$$

Using (3.2), we get

$$\begin{aligned}
|(A_k - B_{l+1})s_k| &\le IH(l) + \frac{|r_l^T s_k|}{c|s_l|} \\
&\le IH(l) + \frac{1}{c}\eta_{k,l}|s_k| + \frac{1}{c} IH(l) \\
&= (1 + \frac{1}{c}) IH(l) + \frac{1}{c}\eta_{k,l}|s_k| \\
&= \frac{1+c}{c}\left(\frac{2+c}{c}\right)^{l-k-1} \eta_{k,l-1}|s_k| + \frac{1}{c}\eta_{k,l}|s_k| \\
&\le \left(\frac{2+c}{c}\right)^{l-k} \eta_{k,l}|s_k| = IH(l+1),
\end{aligned}$$

where the last inequality comes from the simple fact that $\eta_{k,l-1} \le \eta_{k,l}$. □

This proposition shows that if $A_k,\ A_{k+1}, \ldots,\ A_l$ are close to one another (i.e., if $\eta_{k,l}$ is small), then $B_l s_k$ stays quantifiably close to $A_k s_k$.

Now, note that $\|A_* - A_k\| \leq \eta_{k,*}$, and $\eta_{k,*}$ decreases to 0 as k goes to infinity. Keeping the same assumptions, we obtain the following result.

COROLLARY 1. *Take $m, k \in \mathbb{N}$, and let $x \in \mathbb{R}^d$ be in the span of $s_k, \ldots, s_{k+m}$. If*

$$\frac{x}{|x|} = \sum_{i=0}^{m} \lambda_i \frac{s_{k+i}}{|s_{k+i}|}, \quad \lambda_0, \ldots, \lambda_m \in \mathbb{R},$$

then

$$\frac{|B_{k+m+1}x - A_*x|}{|x|} \leq \eta_{k,*} \left(1 + \left(\frac{2+c}{c}\right)^m\right) \sum_{i=0}^{m} |\lambda_i|.$$

Proof: First, it follows from Proposition 1 that

$$\begin{aligned}
\frac{|B_{k+m+1}x - A_*x|}{|x|} &\leq \sum_{i=0}^{m} \frac{|\lambda_i|}{|s_{k+i}|} |B_{k+m+1}s_{k+i} - A_*s_{k+i}| \\
&\leq \sum_{i=0}^{m} \frac{|\lambda_i|}{|s_{k+i}|} \Big(|B_{k+m+1}s_{k+i} - A_{k+i}s_{k+i}| + |A_*s_{k+i} - A_{k+i}s_{k+i}|\Big) \\
&\leq \sum_{i=0}^{m} |\lambda_i| \left(\left(\frac{2+c}{c}\right)^i \eta_{k,k+m} + \eta_{k,*}\right).
\end{aligned}$$

Letting

$$C(m) = \left(1 + \left(\frac{2+c}{c}\right)^m\right)$$

and using $\eta_{k,k+m} \leq \eta_{k,*}$, we get

$$\frac{|B_{k+m+1}x - A_*x|}{|x|} \leq \eta_{k,*} C(m) \sum_{i=0}^{m} |\lambda_i|.$$

The result follows. □

In particular, if we can let k go to infinity while keeping $\sum_{i=0}^{m} |\lambda_i|$ bounded, then we obtain $B_{k+m}x \to A_*x$. Thus, if we can do it for all $x \in \mathbb{R}^d$, we will have proved that $B_k \to A_*$.

Thus, if we prove that *every* normalized vector $x \in \mathbb{R}^d$ is a uniformly bounded linear combination of $s_k, \ldots, s_{k+m}$ as k goes to infinity, we have proved our theorem. In the next section of this paper, we will define a third notion of uniform linear independence of a sequence directly related with this property and prove that it is equivalent to the previous definitions.

4. Uniform m-span of a sequence and applications.

In order to investigate the subspace on which $B_k \to A_*$, we need a notion that is more general than uniform linear independence, that of a uniform span of a sequence of vectors.

DEFINITION 3. *Let $s = (s_k)_{k \geq 0}$ be a sequence in $\mathbb{R}^d$, and let $m \in \mathbb{N}$. We say that a vector x in $\mathbb{R}^d$ is uniformly in the $m-$span of s if for some fixed $\gamma_x > 0$,*

$$\forall k \in \mathbb{N}, \quad \exists \lambda_0, \ldots, \lambda_m \in \mathbb{R} \quad \frac{x}{|x|} = \sum_{i=0}^{m} \lambda_i \frac{s_{k+i}}{|s_{k+i}|} \quad \text{and} \quad \sum_{i=0}^{m} |\lambda_i| \leq \gamma_x. \tag{4.1}$$

We denote by $US_m(s)$ the set of all such vectors.

We have the following trivial result.

LEMMA 1. *$US_m(s)$ is a vector sub-space of $\mathbb{R}^n$. Moreover, there exists a constant $\gamma > 0$ such that Property (4.1) holds for all $x \in US_m(s)$ with $\gamma_x = \gamma$, i.e.,*

$$\begin{gathered}\exists \gamma > 0, \quad \forall k \in \mathbb{N},\ x \in US_m(s), \quad \exists \lambda_0, \ldots, \lambda_m \in \mathbb{R}, \\ \frac{x}{|x|} = \sum_{i=0}^{m} \lambda_i \frac{s_{k+i}}{|s_{k+i}|} \quad \textit{and} \quad \sum_{i=0}^{m} |\lambda_i| \leq \gamma. \end{gathered} \tag{4.2}$$

To prove the existence of γ in (4.2), it suffices to consider an orthonormal basis $(x_i)_i$ of $US_m(s)$, associated with some constants $(\gamma_{x_i})_{1 \leq i \leq d}$, in Property (4.1). Then we can just take $\gamma = \gamma_{x_1} + \cdots + \gamma_{x_d}$.

Remark: There holds $US_m(s) \subset \bigcap_{k=0}^{\infty} \operatorname{span}(s_k, \ldots, s_{k+m})$.

Example: Define the sequence $s = (s_k)$ by

$$s_k = \begin{cases} e_{k \bmod d} & \text{when} \quad k \neq d-1 \bmod d, \\ e_0 + \dfrac{1}{k} e_{d-1} & \text{when} \quad k = d-1 \bmod d. \end{cases}$$

Then

$$US_m(s) = \begin{cases} \{0\} \quad \text{if} \quad 0 \leq m \leq d-1 \\ \operatorname{span}(e_0, \ldots, e_{d-2}) \text{ otherwise.} \end{cases}$$

Using this definition, a simple application of Corollary 1 gives the following result.

PROPOSITION 2. *Let (A_k), (s_k), (y_k), (r_k) and (B_k) be defined as in Section 1, assuming that (A_k) has a limit A_* and that $|r_k^T s_k| \geq c|r_k||s_k|$ for some fixed constant $c > 0$.*
Then, for every $m \in \mathbb{N}$

$$\sup_{x \in US_m(\gamma)} \frac{|B_{k+m+1}x - A_* x|}{|x|} \leq C(m)\gamma\eta_{k,*}, \tag{4.3}$$

where γ is taken from (4.2) and

$$C(m) = \left(1 + \left(\frac{2+c}{c}\right)^m\right).$$

Finally the main result is a consequence of this proposition and of the following lemma.

LEMMA 2. *Let $s = (s_k)_{k \geq 0}$ be a sequence in $\mathbb{R}^d$, and let $m \in \mathbb{N}$. Then s is (m, β)-uniformly linearly independent if and only if $US_m(s) = \mathbb{R}^d$. Moreover, we can take $\gamma = \frac{\sqrt{d}}{\beta}$ in (4.2).* **Proof:** Let $v_1, \ldots, v_d$ be linearly independent elements of $\mathbb{R}^d$ and define the invertible matrix

$$V = \left(\frac{v_1}{|v_1|}, \ldots, \frac{v_d}{|v_d|}\right).$$

Let $\Lambda = (\lambda_1, \dots, \lambda_d)^T \in \mathbb{R}^d$, be such that $x = V\Lambda$ has $|x| = 1$. Then

$$\sum_{i=1}^{d} |\lambda_i| \leq \sqrt{d}\Lambda = \sqrt{d}|V^{-1}x| \leq \frac{\sqrt{d}}{\lambda(V)}.$$

This proves that if a sequence $s = (s_k)$ in $\mathbb{R}^d$ is (m, β)-uniformly linearly independent, then $US_m(s) = \mathbb{R}^d$ and we can take $\gamma_m(s) = \frac{\sqrt{d}}{\beta}$.

On the other hand, take a unit vector $x \in \mathbb{R}^d$ such that

$$V^{-1T}V^{-1}x = \frac{1}{\lambda(V)^2}x.$$

Then, denoting $(\lambda_1, \dots, \lambda_d)^T = \Lambda = V^{-1}x$,

$$\frac{1}{\lambda(V)} = \lambda(V)\ \frac{1}{|\lambda(V)|^2} = \lambda(V)|V^{-1T}V^{-1}x| = \lambda(V)|V^{-1T}\Lambda| \leq |\Lambda| \leq \sum_{i=1}^{d} |\lambda_i|,$$

which proves the converse. $\square$

Our main result is proved.

5. Numerical simulations. In this section, after running numerical simulations of our algorithm on random symmetric matrices, we check that the sequence of inverses of a sequence of matrices can indeed be approximated.

All simulations were done using Matlab on a standard desktop computer.

5.1. Approximation of a sequence of matrices. Here we test the algorithm on random symmetric matrices with coefficients generated by a normalized Gaussian law.

Let $d \in \mathbb{N}\backslash\{0\}$ and define a square symmetric matrix $A_* = \frac{1}{2}(M + M^T)$, where the entries of the $d \times d$ matrix M were chosen at random using the normalized Gaussian law. Fix $0 < \lambda < 1$, and define the sequence $(A_k)_{k\in\mathbb{N}}$ of symmetric matrices obtained by perturbing the matrix A_* as follows

$$A_k = A_* + \frac{\lambda^k}{2}(M_k + M_k^T),$$

where M_k is a matrix with random coefficients taken uniformly in $[-1, 1]$. This gives $\|M_k\| \leq d$. Obviously, $A_k \to A_*$ linearly as $k \to \infty$. More precisely, we have

$$\|A_k - A_*\| \leq d\lambda^k,$$

so $\eta_{k,*} \leq d\lambda^k$.

Remark: While the Gaussian law is better suited to generate random real numbers, we wanted to have clear bounds on the norm of the perturbations $\frac{\lambda^k}{2}(M_k + M_k^T)$. This is why we only took coefficients with absolute value less than one for each M_k.

We define the sequence of unit vectors (s_k), $k \in \mathbb{N}$, $(d, 1)$-uniformly linearly independent, by the formula

$$s_k = e_{k \bmod d}, \quad k \in \mathbb{N},$$

where $(e_0, \dots, e_{d-1})$ is the canonical basis of $\mathbb{R}^d$. Then we apply the symmetric rank-one update to obtain a sequence of symmetric matrices $(B_k)_{k\in\mathbb{N}}$, starting with $B_0 = I_d$. If we assume that there exists $c > 0$ such that $|r_k^T s_k| \geq c|r_k||s_k|$ for every $k \in \mathbb{N}$ then we can apply Theorem 1 with $m = d$, $\beta = 1$, and obtain

$$\|B_k - A_*\| \leq \varepsilon(c, d, k, \lambda)$$

where

$$\varepsilon(d, k, \lambda, c) = \left(1 + \left(\frac{2+c}{c}\right)^{d+1}\right) d^{3/2} \lambda^{k-d}.$$

Remark: In the algorithm, the term $r_k^T s_k = e_{k \bmod d}^T (A_k - B_k) e_{k \bmod d}$ is just the $k \bmod d$-th diagonal term of $(A_k - B_k)$. It is difficult to give an *a priori* estimate on the term c in Theorem 1. For example, if the diagonal terms of the A_k are equal to one, since $B_0 = I_d$, $r_k^T s_k = 0$ for every k and B_k will never be updated.

Table 5.1 computes the best (i.e., the smallest) upper bounds $\varepsilon(c, d, k, \lambda)$ possible for $d = 10$ for different values of k and λ. They are obtained by taking $c = 1$. This will let us compare the rates of convergence of our simulations with the best possible estimates obtainable by Theorem 1. A zero corresponds to a numerical value smaller than the machine epsilon 2.2e-16.

$\varepsilon(1, 10, k, \lambda)$	k=10	k=20	k=50	k=100
λ =0.9	5.6×10^6	2.0×10^6	8.3×10^4	4.4×10^2
λ =0.5	5.6×10^6	5.5×10^3	5.1×10^{-6}	0
λ =0.1	6.5×10^4	5.6×10^{-4}	0	0

Table 5.1: Values for $\varepsilon(c, d, k, \lambda)$

We computed the final distance $\delta_k = \|B_k - A_*\|$ between B_k and A_* for $d = 10$, various values of λ, and various numbers of steps k. We repeated the simulation 1000 times for each value of λ and k, each time with new random values for both A_* and every M_k, $k \in \mathbb{N}$. Table 5.2 gives the mean value and the maximum value of δ_k over these 1000 simulations for each number of steps and each λ.

	k=10		k=20		k=50		k=100	
	mean(δ_k)	max(δ_k)	mean(δ_k)	max(δ_k)	mean(δ_k)	max(δ_k)	mean(δ_k)	max(δ_k)
λ =0.9	1.4×10	4.7×10^3	7.5×10^0	3.8×10^3	1.8×10^{-1}	8.0×10^0	1×10^{-3}	1×10^{-1}
λ =0.5	2.7×10^0	1.7×10^3	6.4×10^{-4}	3×10^{-2}	7.4×10^{-13}	$5.3\ \times10^{-11}$	0	0
λ =0.1	6.8×10^{-2}	7.0×10^0	8.3×10^{-12}	2.1×10^{-9}	0	0	0	0

Table 5.2: Simulation results for $\delta = \|B_k - A_*\|$

Comparing the two tables, we see that the numerical convergence rate is even better than the best possible one given by Theorem 1. These simulations strongly support the theoretical results.

5.2. Approximation of a sequence of inverses. As mentioned in the introduction, our algorithm lets us compute the inverse A_*^{-1} of the limit A_* provided A_* is invertible.

Indeed, consider the following sequences for the symmetric rank-one algorithm

$$s_k := A_k e_{k \bmod d}\,, \quad y_k = A_k^{-1} s_k = e_{k \bmod d}. \tag{5.1}$$

In other words, s_k is the $k \bmod d$-th column of A_k. Then the sequence $(s_k)_{k\in\mathbb{N}}$ is (d,β)-linearly independent for some $\beta > 0$ starting at some k_0 large enough since, as k goes to infinity, the finite set $\{s_k, \ldots, s_{k+d-1}\} = \{A_k e_{k \bmod d}, \ldots, A_{k+d-1} e_{k+d-1 \bmod d}\}$ will converge to the generating set $\{A_* e_0, \ldots, A_* e_{d-1}\}$, and the sequence $(s_{k_0+k})_{k\in\mathbb{N}}$ is therefore d-uniformly linearly independent for some k_0 big enough. The smallest singular value of the matrix

$$\left(\frac{s_k}{|s_k|}, \ldots, \frac{s_{k+d-1}}{|s_{k+d-1}|}\right) \tag{5.2}$$

will converge to that of

$$\left(\frac{A_* e_0}{|A_* e_0|}, \ldots, \frac{A_* e_{d-1}}{|A_* e_{d-1}|}\right),$$

which depends only on A_*, which gives an insight on the correct value of β. The value of c in Theorem 1, however, cannot be guessed here either.

Take the sequence $(B_k)_{k\in\mathbb{N}}$ obtained by applying the symmetric rank-one update method, with starting point $B_0 = I_d$ and using the sequence $(s_k)_{k\in\mathbb{N}}$ defined by (5.2). Assuming that there exists $c > 0$ such that $|r_k^T s_k| \geq c|r_k||s_k|$ for every $k \in \mathbb{N}$, this sequence converges to A_*^{-1} by Theorem 1. The rate of convergence depends on the dimension d and the rate of convergence of A_k. Note that B_k is much easier to compute than A_k^{-1}. Indeed, the complexity for the computation of the inverse of a $d \times d$ matrix is greater than the $O(d^2)$ complexity required in each symmetric rank-one update. This can be useful when solving approximately converging sequences of linear equations, as we will show in the next section.

In our numerical simulation, we computed the distance δ'_k between B_k and A_*^{-1} for different values of k and λ. We used the same sequence (A_k) with random coefficients as in the previous section, with (A_k) converging linearly to a random (but invertible) symmetric matrix A_* with rate $0 < \lambda < 1$. For each number of steps and each value of λ considered, we repeated this simulation 1000 times for different A_* and different random matrices A_k. Table 5.3 gives the mean value and maximum value of δ'_k over these 1000 simulations.

	k=10		k=20		k=50		k=100	
	mean(δ'_k)	max(δ'_k)	mean(δ'_k)	max(δ'_k)	mean(δ'_k)	max(δ'_k)	mean(δ'_k)	max(δ'_k)
λ =0.9	6.5×10	2.4×10^4	2.0×10	4.5×10^3	3.5×10	2.4×10^4	2.2×10^{-1}	9.1×10
λ =0.5	3.3×10	1.3×10^4	2.5×10^0	1.8×10^3	2.8×10^{-8}	2.8×10^{-5}	2.8×10^{-9}	2.6×10^{-6}
λ =0.1	1.4×10	6.7×10^3	4.8×10^{-9}	1.9×10^{-6}	3.7×10^{-11}	2.8×10^{-8}	2.2×10^{-12}	1.4×10^{-10}

Table 5.3: Simulation results for symmetric rank-one update on inverses

We see that $\delta'_k = \|B_k - A_*^{-1}\|$ does decrease to zero, but with a slower rate than that of the approximation of A_* itself given in Table 5.2. Moreover, the maximal value is significantly larger than the mean value for this distance. A reasonable explanation for both discrepancies is that A_* can be ill-conditioned when generated in such a random way. This can cause them to be almost singular, which would have two consequences. First, the rate of convergence of $(A_k^{-1})_{k\in\mathbb{N}}$ to A_*^{-1} is slower than that of $(A_k^{-1})_{k\in\mathbb{N}}$ to A_*. Therefore, the $\eta_{k,*}$ from Theorem 1 is larger than in the case described in Section 5.1. Second, the sequence $s_k = A_k e_{k \bmod d}$ is "less" uniformly linearly independent (that is, the constant β from Definition 2 is smaller).

To test this hypothesis, we tried the simulation again but this time we forced A_* to have a good conditioning. This will make the sequence of matrices A_k uniformly well-conditioned. This kind of sequence can appear in certain numerical simulations of PDEs, in cases where the A_k are discretized versions of a positive-definite self-adjoint operator.

For the simulation, we took A^* so that its singular values all belong to $[0.5, 1.5]$. For this, we used $A_* = O^T DO$, where D is a diagonal matrix of size 10×10 with diagonal coefficients randomly generated using the uniform law on $[0.5, 1.5]$, and O was obtained by orthonormalizing the columns of a random matrix Z, whose coefficients were generated using a Gaussian law. Leaving the rest of the process unchanged, we performed 1000 simulations for the same values of λ and k as those from Table 5.3 and obtained Table 5.4.

	k=10		k=20		k=50		k=100	
	mean(δ'_k)	max(δ'_k)	mean(δ'_k)	max(δ'_k)	mean(δ'_k)	max(δ'_k)	mean(δ'_k)	max(δ'_k)
λ =0.9	2.3×10	2.3×10^3	1.1×10	2.2×10^3	2.5×10^{-1}	2.5×10	2.3×10^{-3}	1.2×10^0
λ =0.5	2.8×10^0	2.2×10^2	1.2×10^{-3}	1.6×10^{-1}	1.3×10^{-12}	4.8×10^{-10}	1.5×10^{-15}	8.7×10^{-12}
λ =0.1	3.6×10^{-1}	3.8×10	7.4×10^{-12}	7×10^{-9}	1.1×10^{-14}	4.8×10^{-12}	8.2×10^{-15}	1.4×10^{-12}

Table 5.4: Simulation results for inverses of matrices with singular values in $[0.5, 1.5]$

As expected, the numbers on Table 4 show that the sequence $(B_k)_{k\in\mathbb{N}}$ converges to A_*^{-1} much faster than in the case of a purely random A_*. In fact, the convergence is almost as good as the one shown by Table 5.2 in the previous section. This confirms that the method is more effective with sequences of matrices that are well scaled.

We also did an extra simulation in the case of a purely random A_*: since the sequence $(s_k)_{k\in\mathbb{N}}$ in (5.1) has no reason to be particularly good (i.e., uniformly linearly independent with a nice constant), we applied our algorithm this time by taking a new sequence for $(y_k)_{k\in\mathbb{N}}$, letting each y_k be a random vector with coefficients taken along a normal Gaussian law at each step. We still set $s_k = A_k y_k$. This is done in the hopes that, on average, the sequence s_k could be "more" uniformly linearly independent, that is, the term β in Definition 2 could be higher. We computed the mean values for $\delta'_k = \|B_k - A_*^{-1}\|$ over 1000 repetition of this simulation. They are given in Table 5.5.

	k=10	k=20	k=50	k=100
λ =0.9	8.3×10	5.6×10	4.7×10	1.2×10
λ =0.5	8.5×10	2.9×10	1.0×10^{-4}	$1.2\ \times10^{-7}$
λ =0.1	5.4×10	2.1×10^{-4}	3.7×10^{-7}	1.0×10^{-9}

Table 5.5: Simulation results for inverses of matrices y_k *randomly generated*

This experiment shows that taking a random sequence of vectors $(y_k)_{k\in\mathbb{N}}$ is not as effective as taking the y_k periodically equal to the canonical basis of $\mathbb{R}^d$.

For large values of d, as k goes to infinity, this method gives us an approximation of the whole sequence $(A_k^{-1})_{k\in\mathbb{N}}$ and is faster than computing the inverse of A_k at every step. Indeed, the complexity of one rank-one update is only in $O(d^2)$.

Remark: This method does not allow for better computations of the inverse of a badly scaled matrix A by setting $A_k = A$ for every k. A quick Matlab simulation showed that the command $inv(A)$ gives more precise results.

6. An application: optimal deformations of constrained shapes. The main problem of shape deformation analysis is to find an optimal deformation from one shape to another. From the numerical point of view, a shape is usually a collection of distinct points $q^T = (x_1^T, \dots, x_n^T)$ where n is a positive integer and each x_i is an element of $\mathbb{R}^d$. These points are usually a discretization of the boundary of a certain domain in $\mathbb{R}^d$. The space of such shapes is called the space $Lmk_d(n)$ of n landmarks in $\mathbb{R}^d$, i.e.,

$$Lmk_d(n) = \{q = (x_1^T, \dots, x_n^T)^T \in \mathbb{R}^{nd}, i \neq j \Rightarrow x_i \neq x_j\}.$$

A deformation of an initial shape q_0, with $q_0^T = (x_{1,0}^T, \dots, x_{n,0}^T)$, is a curve $t \mapsto q(t)$ with $q(0) = q_0$, of Sobolev class $W^{1,2}$, that is, an absolutely continuous curve with square-integrable speed.

6.1. Large deformation diffeomorphic metric mapping.. The so-called LDDMM (Large Deformation Diffeomorphic Metric Mapping) setting for studying deformations of landmarks is used in computational anatomy [2, 6, 11].

We start by considering a Reproducing Kernel Hilbert Space V of smooth vector fields on $\mathbb{R}^d$, that is, a subspace V of $\mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R}^d)$ equipped with a Hilbert product $\langle \cdot, \cdot \rangle_V$ such that the inclusion $V \hookrightarrow \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R}^d)$ is continuous. For such a space, there exists a matrix-valued *reproducing kernel* $K : \mathbb{R}^d \times \mathbb{R}^d \to M_d(\mathbb{R}^d)$ such that, for any $x, v \in \mathbb{R}^d$, and every $X \in V$,

$$\langle K(\cdot, x)v, X \rangle_V = v^T X(x).$$

Such a space V is completely determined by its reproducing kernel K. The kernel we use in this example is given by

$$K(x, y) = e^{-\frac{|x-y|^2}{2\sigma}} I_d,$$

for some $\sigma > 0$.

Then, one considers deformations $t \mapsto q(t) \in Lmk_d(n)$ of the form

$$q(t)^T = (\varphi^X(t) \cdot q_0)^T = (\varphi^X(t)(x_{1,0})^T, \dots, \varphi^X(t)(x_{n,0})^T),$$

where $(\varphi^X(t))_{t \in [0,1]}$ is a family of diffeomorphisms of $\mathbb{R}^d$, flow of a time-dependent vector field $t \mapsto X(t, \cdot) \in V$ on $\mathbb{R}^d$ such that $t \mapsto \|X(t, \cdot)\|_V$ is square-integrable. The optimal deformation $t \in [0, 1] \mapsto \varphi^X(t) \cdot q_0$ from a starting shape q_0 to a target shape q_1 is the one such that the total energy $\frac{1}{2} \int_0^1 \langle X(t), X(t) \rangle_V \, dt$ is minimal. The reason of using such a setting is that it actually provides an optimal deformation of the full space $\mathbb{R}^d$ thanks to the flow φ^X.

In particular, as $q(0)$ is the discretization of the boundary of a certain domain U of $\mathbb{R}^d$, $q(t)$ will then be a discretization of the boundary of the deformed domain $U(t) = \varphi(t)(U)$.

Since it is extremely hard to determine this minimum, one usually considers the penalized functional

$$\hat{J}(X) = \frac{1}{2} \int_0^1 \langle X(t), X(t) \rangle_V \, dt + g(\varphi^X(1) \cdot q_0),$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is a smooth data attachment term, minimal at q_1. A classical result [12], consequence of the solution to the spline interpolation problem for vector fields in V, is that minimizing $\hat{J}$ is equivalent to minimizing the functional

$$J(u) = \frac{1}{2} \int_0^1 \sum_{i,j=1}^{n} e^{\frac{|x_i(t) - x_j(t)|^2}{2\sigma}} u_i(t)^T u_j(t) dt + g(q(1)),$$

where $u_i \in L^2(0,1;\mathbb{R}^d)$ for every $i = 1,\dots,n$, and $q(t) = (x_1(t),\dots,x_n(t))$ satisfies the control system

$$q(0) = q_0, \quad \dot{x}_i(t) = \sum_{j=1}^{n} e^{\frac{|x_i(t)-x_j(t)|^2}{2\sigma}} u_i(t) \quad \text{a.e. } t \in [0,1], \quad i = 1,\dots,n.$$

We can retrieve the corresponding flow φ^X by integrating the vector field

$$X(t,x) = \sum_{j=1}^{n} e^{\frac{|x-x_j(t)|^2}{2\sigma}} u_i(t).$$

We can write this differential equation as $\dot{q}(t) = K^{\sigma}_{q(t)} u(t)$, where K^{σ}_{q} is the block matrix of size $nd \times nd$ consisting of blocks of size $d \times d$, with the (i,j)-th block is equal to $e^{\frac{|x_i(t)-x_j(t)|^2}{2\sigma}} I_d$, and $u = (u_1^T,\dots,u_n^T)^T \in (\mathbb{R}^d)^n$. This is a symmetric, positive-definite matrix for every q in $Lmk_d(n)$, and we also have

$$\sum_{i,j=1}^{n} e^{\frac{|x_i-x_j|^2}{2\sigma}} u_i^T u_j = u^T K^{\sigma}_{q} u.$$

In this form, this an optimal control problem in finite dimension. It has an optimal solution which satisfies certain Hamiltonian equations, and can be solved numerically (see[1, 11, 12]).

6.2. Shapes with constraints.. The shape deformation problem which motivated the symmetric rank-one update described in this paper is an extension of the one described in the previous section, aimed at studying several shapes simultaneously. Let n^1, n^2 be two positive integers. Assume that we have two different starting shapes q_0^1 and q_0^2, belonging to different landmark spaces $Lmk_d(n^1)$ and $Lmk_d(n^2)$, each one being a discretization of a different domain U^1 and U^2 with $U^1 \cap U^2 = \emptyset$.

Usually, one would just consider the total shape to be the union of those two shapes, and deform it using a single diffeomorphism. However, the objects we want to model should be considered as two independent shapes, as in the case of images of different parts of the brain.

This is why, instead, we would like to model a deformation of q_0^1 and q_0^2 such that they evolve independently from one another (each one being deformed by a different diffeomorphism), but are immersed in a deformable background (deformed by a third diffeomorphism), whose boundary coincides with the union of the boundaries of U^1 and U^2: one obtains a discretization q^3 of this boundary by concatenation $(q_0^3)^T = ((q_0^1)^T, (q_0^2)^T) \in Lmk_d(n^3)$, where $n^3 = n^1 + n^2$. The total shape $q_0^T = ((q_0^1)^T, (q_0^2)^T, q_0^3)^T)$ belongs to the space $Lmk_d(n^1) \times Lmk_d(n^2) \times Lmk_d(n^3)$.

The main reason for considering constrained shape deformation is to model such a system [1]. Indeed, to model a deformation $q(t)^T = (q^1(t)^T, q^2(t)^T, q^3(t)^T)$ of the total shape, we can use the control system $q(0) = q_0, \quad \dot{q}^1(t) = K^{\sigma^1}_{q^1(t)} u^1(t), \quad \dot{q}^2(t) = K^{\sigma^2}_{q^2(t)} u^2(t), \quad \dot{q}^3(t) = K^{\sigma^3}_{q^3(t)} u^3(t), \quad$ a.e. $t \in [0,1]$, where $u^i \in L^2(0,1;\mathbb{R}^{dn^i}), i = 1,2,3$. This control system can be written $\dot{q}(t) = K_{q(t)} u(t)$, where

$$K_q = \begin{pmatrix} K^{\sigma^1}_{q^1} & 0 & 0 \\ 0 & K^{\sigma^2}_{q^2} & 0 \\ 0 & 0 & K^{\sigma^3}_{q^3} \end{pmatrix}.$$

Note that the matrix K_q is symmetric and positive definite for every $q \in Lmk_d(n^1) \times Lmk_d(n^2) \times Lmk_d(n^3)$. The functional we want to minimize is therefore given by

$$J(u) = \frac{1}{2}\int_0^1 u(t)^T K_{q(t)} u(t) dt + g(q(1)).$$

K_q is a symmetric positive definite matrix. However, one also needs to preserve the condition $q^3(t)^T = ((q^1(t))^T, (q^2(t))^T)$ for every $t \in [0,1]$ (that is, the boundary of the deformed background coincides with the boundaries of the deformed domains). This means that we should impose on our control system the constraints $(\dot{q}^3)^T = ((\dot{q}^1)^T, (\dot{q}^2)^T)$, i.e.,

$$K^{\sigma^3}_{q^3(t)} u^3(t) = ((K^{\sigma^1}_{q^1(t)} u^1(t))^T, (K^{\sigma^2}_{q^2(t)} u^2(t))^T)^T \quad \text{a.e. } t \in [0,1].$$

These linear constraints can be as written $CK_{q(t)}u(t) = 0$, with

$$C = \begin{pmatrix} I_{n^3} & -I_{n^3} \end{pmatrix}.$$

In conclusion, we wish to find a minimum of $J(u) = \frac{1}{2}\int_0^1 u(t)^T K_{q(t)} u(t) dt + g(q(1))$ over all square-integrable functions $u : [0,1] \to (\mathbb{R}^d)^{n^1+n^2+n^3}$ such that $q(0) = q_0$ and, for almost every t in $[0,1]$, $\dot{q}(t) = K_{q(t)}u(t)$ and $CK_{q(t)}u(t) = 0$.

Now, define for $q \in Lmk_d(n^1) \times Lmk_d(n^2) \times Lmk_d(n^1+n^2)$ the symmetric positive definite $n^3 \times n^3$ matrix $A_q = CK_qC^T$.

Then, according to an appropriate version of the Pontryagin Maximum Principle ([1, 10]), if u is optimal for our constrained minimization problem, and q is the curve such that $q(0) = q_0$ and $\dot{q} = K_q u$, then there exists an absolutely continuous curve $p : [0,1] \in (\mathbb{R}^d)^{n^1+n^2+n^3}$, called the *momentum* of the trajectory, with square-integrable speed such that $p(1) + \nabla g_{q(1)} = 0$, and, for almost every t in $[0,1]$,

$$\begin{cases} u(t) = \left(p(t) - C^T A^{-1}_{q(t)} CK_{q(t)} p(t)\right), \\ \dot{q}(t) = K_{q(t)} \left(p(t) - C^T A^{-1}_{q(t)} CK_{q(t)} p(t)\right), \\ \dot{p}(t) = -\frac{1}{2}\left(p(t) - C^T A^{-1}_{q(t)} CK_{q(t)} p(t)\right)^T \nabla_q K_{q(t)} \left(p(t) - C^T A^{-1}_{q(t)} CK_{q(t)} p(t)\right). \end{cases} \tag{6.1}$$

Here, we used the notation $a^T \nabla_q K_q b$ for the gradient of the map $q \mapsto a^T K_q b$ at q, for $a, b \in (\mathbb{R}^d)^{n^1+n^2+n^3}$ fixed. Since this is a differential equation with smooth coefficient, it has a unique solution for fixed (q_0, p_0)

Remark: The system (6.1) actually consists of the Hamiltonian geodesic equations on the submanifold defined by $C = 0$ for the Riemannian metric whose cometric tensor is K_q.

Moreover, in this case, $t \mapsto u(t)^T K_{q(t)} u(t)$ is constant. Hence, the minimization of J reduces to the minimization of

$$\tilde{J}(p_0) = \frac{1}{2}\left(p_0 - C^T A^{-1}_{q_0} CK_{q_0} p_0\right)^T K_{q_0} \left(p_0 - C^T A^{-1}_{q_0} CK_{q_0} p_0\right) + g(q(1))$$

with respect to the initial momentum $p_0 = p(0)$. Note that this reduction is fundamental in our approach.

The computation the gradient of $\tilde{J}$ requires solving an adjoint equation with coefficients depending on the derivatives of the right-hand side of (6.1). This is described in more detail in [1].

All operations involved in the computation of this gradient have complexity in $O((n^1+n^2)^2)$ at each time step, with the distinct exception of the computation of the inverse of A_q (or, at least, the resolution of linear equations of the form $A_q a = b$ which appear both in (6.1) and several times in the associated adjoint equation), whose complexity is higher.

6.3. Implementation of the symmetric rank-one update.. One of the most time-consuming aspects of this method is the computation, at each time step, of the inverse of A_q. We can speed things up by applying a symmetric rank-one update as follows.

Let $e_0, \dots, e_{d(n^1+n^2)-1}$ be the canonical vector basis of $\mathbb{R}^{dn^1+dn^2}$. For any $k \in \mathbb{N}$, define

$$y_k = e_{k \bmod d(n^1+n^2)}.$$

We start with the initial momentum $p_0 = 0$ and let $B_0(t) = A_{q_0}^{-1}$ for $t \in [0,1]$. The computation of $A_{q_0}^{-1}$ is necessary to compute the different gradients anyway, so this does not add any extra time, and gives a better initial conditioning of the matrix. Then, assuming we have constructed an initial momentum p_k and a family of matrices $B_k(t),\ t \in [0,1]$, we use (6.1) to compute a trajectory $x_k(t)$, replacing A_q^{-1} by $B_k(t)$. Finally, at each time t, we define

$$\begin{aligned} s_k(t) &= A_{q_k(t)} y_k, \\ r_k(t) &= B_k(t) s_q(t) - y_k, \\ B_{k+1}(t) &= B_k(t) + \frac{r_k(t) r_k^T(t)}{r_k^T(t) s_k(t)}. \end{aligned}$$

We can then compute the gradient of $\tilde{J}$ with an adjoint equation, where any directional derivative

$$\partial_q (A_{q_k(t)} a)^{-1}(b) = -A_{q_k(t)}^{-1} \partial_q (A_{q_k(t)})(b) A_{q_k(t)}^{-1} a, \quad a \in \mathbb{R}^{n^3}, \quad b \in (\mathbb{R}^d)^{n^1+n^2+n^3}$$

is replaced with $-B_k(t)\partial_q A_{q_k(t)}(v) B_k(t) a$. This lets us perform the minimization of $\tilde{J}$ using gradient descent or a regular quasi-Newton algorithm.

As long as the algorithm gives a converging sequence of initial momenta p_k, the trajectories $q_k(t)$ will also converge to a trajectory $q_*(t)$, making each $A_{q_k(t)}$, with $t \in [0,1]$ fixed, a converging sequence, with invertible limit $A_*(t)$. Therefore, each $B_k(t)$, for $t \in [0,1]$ fixed, converges to $A_*(t)$ as $k \to \infty$. In other words, as $k \to \infty$, we are indeed computing the true gradient of $\tilde{J}$.

6.4. Numerical simulations. We consider the shapes q_0^1 and q_0^2 as an equidistant discretization of $n^1 = n^2 = n$ points on circles of radius 1 in $\mathbb{R}^2$, centered at $(-1,0)$ (resp. $(2,0)$), and we try to match them with circles of radius 1 (resp. 0.9) centered at $(-0.75,0)$ (resp. $(1.5,0)$). See Figure 6.1.

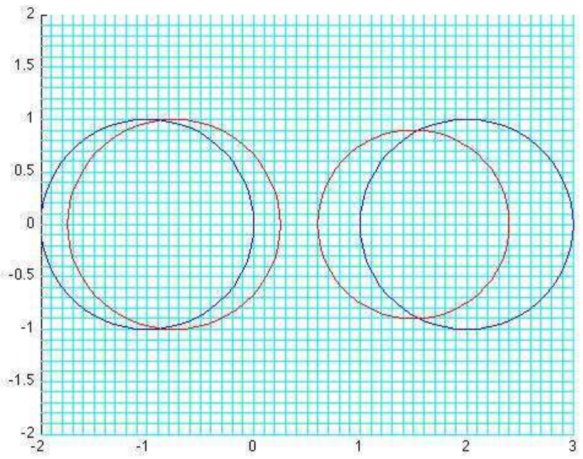

Figure 6.1: Multiple shape experiment: initial shapes (two outer circles) and target shapes (two inner circles).

This is actually a difficult matching to perform without the context of constrained shape deformation, because deforming two separate shapes into targets that are so close to each other is very difficult using a single diffeomorphism. Constrained multiple shapes provide an appropriate framework for finding such a matching.

The following values are taken for the constants: $\sigma^1 = 0.5$, $\sigma^2 = 0.3$ and $\sigma^3 = 0.1$. We used 10 time steps, with step length $\Delta t = 0.1$.

The first thing that we compared is the time necessary to accomplish one gradient step using the adjoint equations to (6.1) as described in [1]. Then we measured the time required to complete the algorithm with the same stopping condition on the gradient algorithm. Finally, we compare how much the constraints are satisfied in the final deformation obtained by the gradient algorithm. This is done by computing the total value of the lack of satisfaction of the constraints

$$\gamma(u(\cdot)) = \sqrt{\int_0^1 |CK_{q(t)}u(t)|^2 dt}.$$

We obtain Table 6.1. It gives a comparison between the regular method of computing the exact solutions of any linear system $A_q a = b$ appearing in the adjoint equations, and the one using the symmetric rank-one update method described in the previous paragraph.

Number of points	n=30		n=60	
Method used	Regular method	Rank-one update	Regular method	Rank-One update
Time (one step)	3.3	1	9.8	2.6
Total time	63	23	69	23
$\gamma(u(\cdot))$	1.0×10^{-15}	6.6×10^{-1}	1.0×10^{-15}	1.1×10^{0}

Table 6.1: Comparison of the time necessary to perform a gradient descent algorithm and satisfaction of constraints

As expected, the symmetric rank-one algorithm is much faster. However, since the regular method computes the exact solutions for satisfying the constraints, those are satisfied with great precision. This is not the case when using the rank-one update method where we obtain $\gamma(u(\cdot)) = 1.1$ in the case $n = 60$.

Remark: this value is actually quite small, since we are computing an Euclidean norm in $\mathbb{R}^{n^1+n^2} = \mathbb{R}^{120}$.

Figure 6.2 is a picture of the final deformation obtained by the rank-one update (since we are in the framework of LDDMM, the deformations are induced by diffeomorphisms, showcased by their action on a grid).

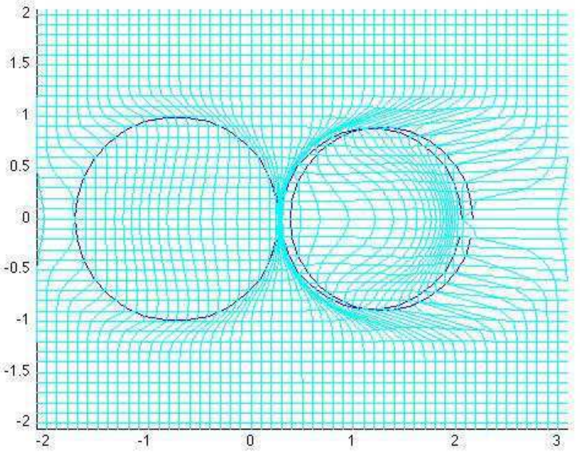

Figure 6.2: Multiple shape experiment: matching circles with the rank-one update method.

Four circles are required to fully represent the total shape: one for each of the two shapes, and two for the background. The constraints are satisfied when the two shapes coincide with the background, so that only two circles appear.

In Figure 6.2, the constraints forced the background and the shape to coincide on the left-hand side. On the right-hand side, however, the background and the disk did not move together completely and so the constraints are only approximately satisfied. Note that while visible, the difference is rather small.

We could force the constraints to be better satisfied by increasing the number of gradient steps. In fact, we obtained numbers as small as 1e-4 for the value of $\gamma(u(\cdot))$ by taking a great number of steps (around 2000 or so). However, in this case, it is faster to use the regular method.

7. Conclusion. The symmetric rank-one update used in quasi-Newton methods for minimizing real functions can be generalized to a more abstract framework. The algorithm this paper obtained can be used to approximate sequences of symmetric matrices. This opens several possible applications, one of which is the computation of the sequence of inverses of a sequence of invertible matrices. This can be applied to compute constrained optimal controls for which the computation of the inverse of the constraints is too computationally demanding. This is particularly useful when tackling problems of shape deformations, where the number of constraints can be quite large. We are hopeful that other applications of the symmetric rank-one update method can be given.

However, some limitations remain. In particular, the rate of convergence may be low, since, in higher dimensions, the term $\left(\frac{1+c}{c}\right)^{m+1}$ in the main theorem will be quite large, because m is at least equal to the number of columns in the matrices of the sequence and is therefore large in high dimensions. Moreover, just as for the classical quasi-Newton algorithms, there is no clear way to find a lower bound for c, even in the more simple cases.

REFERENCES

[1] S. Arguillère, E. Trélat, A. Trouvé, and L. Younès. Shape deformation analysis from the optimal control viewpoint. *To appear in Jounal de mathématiques pures et appliquées*, http://arxiv.org/abs/1401.0661.

[2] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *Int. J. Comput. Vis.*, 61(2):139–157, 2005.

[3] A. R. Conn, N. I. Gould, and P. L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50(2):177–195, 1991.

[4] J. E. Dennis, Jr. and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19(1):46–89, 1977.

[5] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. Corrected reprint of the 1983 original.

[6] P. Dupuis, U. Grenander, and M. I. Miller. Variational problems on flows of diffeomorphisms for image matching. *Quart. Appl. Math.*, 56(3):587–600, 1998.

[7] P. E. Gill, W. R. Murray, and M. H. Wright. *Practical optimization*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[8] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*, volume 30 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original.

[9] G. Schuller. On the order of convergence of certain quasi-Newton methods. *Numer. Math.*, 23:181–192, 1974.

[10] E. Trélat. *Contrôle optimal*. Mathématiques Concrètes. [Concrete Mathematics]. Vuibert, Paris, 2005. Théorie & applications. [Theory and applications].

[11] A. Trouvé. Diffeomorphism groups and pattern matching in image analysis. *International Journal of Computational Vision*, 37(1):17, 2005.

[12] L. Younes. *Shapes and diffeomorphisms*, volume 171 of *Applied Mathematical Sciences*. Springer-Verlag, Berlin, 2010.