Interpolation and regression of rotation matrices

Nicolas Boumal

Université catholique de Louvain, Louvain-la-Neuve, Belgium,
Department of mathematical engineering, ICTEAM Institute,
nicolasboumal@gmail.com

Abstract. The problem of fitting smooth curves to data on the group of rotations is considered. This problem arises when resampling or denoising data that consists in rotation matrices measured at different times. The rotation matrices typically correspond to the orientation of some physical object, such as a camera or a flying or submarine device. We propose to compute sequences of rotations (discretized curves) that strike a tunable balance between data fidelity and smoothness, where smoothness is assessed by means of a proposed notion of velocity and acceleration along discrete curves on the group of rotations. The best such curve is obtained via optimization on a manifold. Leveraging the simplicity of the cost, we present an efficient algorithm based on second-order Riemannian trust-region methods, implemented using the Manopt toolbox.

Keywords: optimization on manifolds; non-parametric regression; denoising of rotations; video stabilization; 3D motion planning.

1 Introduction

Rotations in $\mathbb{R}^n$ are conveniently represented as $n \times n$ orthogonal matrices with determinant $+1$ (as opposed to -1, that is, reflections are not allowed). The set of such rotations forms a Lie group, called the *special orthogonal group* $\mathrm{SO}(n)$. The usual Riemannian structure on $\mathrm{SO}(n)$ is that of a Riemannian submanifold of $\mathbb{R}^{n\times n}$ endowed with the inner product $\langle H_1, H_2\rangle = \mathrm{trace}(H_1^\top H_2)$, see Table 1.

We consider problems for which the data consists in measurements of rotations $p_1, \ldots, p_N \in \mathrm{SO}(n)$ corresponding to measurement times $t_1 \leq \cdots \leq t_N$ and such that the task is to find a curve $\gamma\colon [t_1, t_N] \to \mathrm{SO}(n)$ that is as smooth (as regular) as possible while reasonably fitting the data: $\gamma(t_i) \approx p_i$. In other words: the task it that of regression on the group of rotations. Mathematically, this task can be modeled as that of minimizing the following cost functional [7]:

$$E_{\mathrm{c}}(\gamma) = \frac{1}{2}\sum_{i=1}^{N} w_i \,\mathrm{dist}^2(\gamma(t_i), p_i) + \frac{\lambda}{2}\int_{t_1}^{t_N} \|\dot\gamma(t)\|^2 \,\mathrm{d}t + \frac{\mu}{2}\int_{t_1}^{t_N} \|\ddot\gamma(t)\|^2 \,\mathrm{d}t, \quad (1)$$

where $\dot\gamma(t)$ is the velocity of γ at time t, i.e., $\dot\gamma(t) = \frac{\mathrm{d}}{\mathrm{d}t}\gamma(t)$—which is necessarily a tangent vector at $\gamma(t)$—and $\ddot\gamma(t)$ is the acceleration of γ at time t, i.e., the projection of the second derivative of γ in the ambient space to the tangent

Set:	$\mathrm{SO}(n) = \{A \in \mathbb{R}^{n\times n} \ : \ A^\top A = I_n \text{ and } \det(A) = 1\}$
Tangent spaces:	$\mathrm{T}_A\mathrm{SO}(n) = \{A\Omega \in \mathbb{R}^{n\times n} \ : \ \Omega + \Omega^\top = 0\}$
Inner product:	$\langle A\Omega_1, A\Omega_2\rangle = \mathrm{trace}(\Omega_1^\top \Omega_2)$
Vector norm:	$\|A\Omega\| = \sqrt{\langle A\Omega, A\Omega\rangle}$
Distance:	$\mathrm{dist}(A, B) = \|\log(A^\top B)\|$
Exponential:	$\mathrm{Exp}_A(A\Omega) = A\exp(\Omega)$
Logarithm:	$\mathrm{Log}_A(B) = A\log(A^\top B)$
Projector:	$\mathrm{P}_A(H) = A\,\mathrm{skew}\big(A^\top H\big)$

Table 1. Toolbox for $\mathrm{SO}(n)$. Matrix norms correspond to the Frobenius norm.

space at $\gamma(t)$: $\ddot{\gamma}(t) = \mathrm{P}_{\gamma(t)}(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\gamma(t))$. The parameters $\lambda \geq 0$ and $\mu \geq 0$ enable the user to tune the balance between the conflicting goals of fitting and smoothness. When $\lambda > 0$, $\mu = 0$, the optimal γ is piecewise geodesic. When $\lambda = 0$, $\mu > 0$, the optimal γ is an approximating cubic spline.

Regression may serve at least two purposes: that of denoising the data (the measurements p_i may be noisy) and that of resampling the data (a user may need to know the rotation/orientation of some object at a time t for which no measurement is available). In the extreme case where $\gamma(t_i) = p_i$ is enforced exactly (interpolation case), it is this latter purpose that is most important.

In particular, interpolation on $\mathrm{SO}(n)$ may be useful for 3D motion planning (of a camera in 3D computer graphics for example) where the position and orientation of an object is set by a user who requires the system to smoothly transition from one configuration to the next at prescribed times. Likewise, regression on $\mathrm{SO}(n)$ may be useful to stabilize video frames acquired by an unsteady camera [6]. In that context, it is useful to estimate the orientation of the camera at each frame, using either image processing or a built-in gyroscope. In the latter case, it may furthermore be so that the frames are acquired at different times than the orientation measurements, which requires a resampling that can be achieved with the proposed method. The present contribution could be applied directly to the camera stabilization application considered in [6], with the nice benefit of additional smoothness owing to a second-order regularity term.

Computationally, the problem formulation using (1) may not be the most practical as it involves an optimization problem in infinite dimension on a manifold. To address this concern, discretization methods were proposed in [4]. The idea is to search for a sequence of rotations $\gamma = (\gamma_1, \ldots, \gamma_{N_\mathrm{d}}) \in \mathrm{SO}(n)^{N_\mathrm{d}}$ associated to time labels $t_1 = \tau_1 < \tau_2 < \cdots < \tau_{N_\mathrm{d}} = t_N$ such that the discrete sequence γ appears as smooth as possible (we need to define that) and fits the data as well as possible.

For simplicity of exposition, we assume throughout that the discretization times τ_i are uniformly spaced over the interval $[t_1, t_N]$ with spacing $\Delta\tau = \tau_{i+1} - \tau_i$. This restriction is purely for ease of notation and is easy to relax.

There are three steps to take in order to discretize the cost function (1). First, the data points p_i corresponding to time labels t_i need to be matched

to discretization points γ_i corresponding to discretization times τ_i. To this end, define s_i, an index between 1 and N_d such that τ_{s_i} is as close as possible to t_i (typically, one would include the t_i's as a subset of the τ_i's for perfect matching). Second, the notions of velocity $\dot{\gamma}(t)$ and acceleration $\ddot{\gamma}(t)$, both vectors in the tangent space $\mathrm{T}_{\gamma(t)}\mathrm{SO}(n)$, need to be discretized. For curves in $\mathbb{R}^n$, the velocity may be approximated using finite differences as such:

$$\dot{\gamma}(\tau_i) \approx \frac{\gamma(\tau_{i+1}) - \gamma(\tau_i)}{\Delta\tau}. \tag{2}$$

The *geometric finite differences* method proposed in [4] builds on the observation that the difference appearing in the numerator is not defined for manifolds in general but may be interpreted: it is a vector starting at $\gamma_i = \gamma(\tau_i)$, pointing toward γ_{i+1} and with length equal to the distance separating these two points. The logarithmic map on a manifold (the inverse of the exponential map, which generates geodesics) embodies this same notion, hinting toward the following definition of velocity along a discrete curve on $\mathrm{SO}(n)$:

$$\dot{\gamma}_i \triangleq \gamma_i \frac{\log(\gamma_i^\top \gamma_{i+1})}{\Delta\tau}. \tag{3}$$

Likewise, the acceleration at time τ_i may be approximated for curves in $\mathbb{R}^n$:

$$\ddot{\gamma}(\tau_i) \approx \frac{(\gamma(\tau_{i+1}) - \gamma(\tau_i)) + (\gamma(\tau_{i-1}) - \gamma(\tau_i))}{\Delta\tau^2}. \tag{4}$$

Using the same trick, the discrete acceleration on a manifold is defined as:

$$\ddot{\gamma}_i \triangleq \gamma_i \frac{\log(\gamma_i^\top \gamma_{i+1}) + \log(\gamma_i^\top \gamma_{i-1})}{\Delta\tau^2}. \tag{5}$$

This formula exhibits several desirable properties. In particular, $\ddot{\gamma}_i$ is zero if and only if there exists a geodesic $\gamma(t)$ such that $\gamma(\tau_j) = \gamma_j$, $j = i-1, i, i+1$. Finally, the third step to discretize (1) is to replace the integrals over $[t_1, t_N]$ with an appropriately weighted sum of the numbers $\|\dot{\gamma}_i\|^2$ and $\|\ddot{\gamma}_i\|^2$. Combining these steps yields the cost function from [4] for discrete regression on $\mathrm{SO}(n)$:

$$\begin{aligned} E_\mathrm{d}(\gamma) = \frac{1}{2}\sum_{i=1}^{N} w_i \,\|\log(\gamma_{s_i}^\top p_i)\|^2 + \frac{\lambda}{2}\sum_{i=1}^{N_\mathrm{d}-1} \Delta\tau \left\| \frac{\log(\gamma_i^\top \gamma_{i+1})}{\Delta\tau} \right\|^2 \\ + \frac{\mu}{2}\sum_{i=2}^{N_\mathrm{d}-1} \Delta\tau \left\| \frac{\log(\gamma_i^\top \gamma_{i+1}) + \log(\gamma_i^\top \gamma_{i-1})}{\Delta\tau^2} \right\|^2. \end{aligned} \tag{6}$$

In [4], the cost function (6) is minimized over $\mathrm{SO}(n)^{N_\mathrm{d}}$ using a Riemannian conjugate gradient method [2]. It turns out that this first-order algorithm can be slow, in particular for large values of μ. This hints that second-order methods might be of use. Unfortunately, computing the Hessian of E_d is intricate because of the matrix logarithm.

In this communication, we propose to simplify the cost function (6) to make it both simpler to compute and simpler to differentiate. This paves the way toward second-order optimization methods to solve the regression problem on $\mathrm{SO}(n)$. In the next section, the new cost function is proposed and differentiated. In Section 3, an algorithm to minimize the cost is described. Section 4 presents some numerical experiments.

2 A simpler cost function

Because we aim at second-order optimization methods and because it is not convenient to differentiate the matrix logarithm twice, in simplifying the cost function (6) we will get rid of all matrix logarithms.

The proposed cost function. A first obvious observation is that, for two close rotation matrices $A, B \in \mathrm{SO}(n)$, the geodesic distance and the chordal distance (that is, the distance in the embedding space $\mathbb{R}^{n\times n}$) are approximately equal:

$$\mathrm{dist}(A, B) = \left\|\log(A^\top B)\right\| \approx \|B - A\|.$$

Indeed, let Ω be skew-symmetric with $\|\Omega\| = 1$ and let $B = A\exp(t\Omega)$. Then,

$$\begin{aligned}\mathrm{dist}^2(A, B) &= \left\|\log(A^\top B)\right\|^2 = t^2,\\ \|B - A\|^2 &= \|\exp(t\Omega) - I_n\|^2 = \left\|t\Omega + \frac{1}{2}t^2\Omega^2 + \mathcal{O}(t^4)\right\|^2\\ &= t^2 + t^3\left\langle\Omega, \Omega^2\right\rangle + \mathcal{O}(t^4) = t^2 + \mathcal{O}(t^4).\end{aligned}$$

We used the Taylor expansion of the matrix exponential $\exp(\Omega) = \sum_{k=0}^{\infty}\Omega^k/k!$ and the fact that Ω^2 is symmetric, so that the inner product $\left\langle\Omega, \Omega^2\right\rangle$ vanishes.

This first observation is sufficient to take care of the two first terms in (6). For the third term, let us try to approximate the logarithmic map directly. The map $B \mapsto A\log(A^\top B)$ associates to $B \in \mathrm{SO}(n)$ a tangent vector at $A \in \mathrm{SO}(n)$ that points from A to B. To produce a similar vector, a simple way is to project the vector $B - A$ from the ambient space $\mathbb{R}^{n\times n}$ to the tangent space at A (an idea that generalizes nicely [3]):

$$\mathrm{P}_A(B - A) = A\,\mathrm{skew}\big(A^\top(B - A)\big) = A\,\mathrm{skew}\big(A^\top B - I_n\big) = A\,\mathrm{skew}\big(A^\top B\big).$$

This suggests to approximate the matrix logarithm by the operator $\mathrm{skew}(M) = (M - M^\top)/2$. Indeed, with the same matrices A and B such that $A^\top B = \exp(t\Omega)$,

$$\begin{aligned}\log(A^\top B) &= t\Omega,\\ \mathrm{skew}\big(A^\top B\big) &= \mathrm{skew}\left(I + t\Omega + \frac{1}{2}t^2\Omega^2 + \mathcal{O}(t^3)\right) = t\Omega + \mathcal{O}(t^3).\end{aligned}$$

Hence, for close enough matrices A and B, the skew operator is a good approximation of the matrix logarithm. Furthermore, it has the nice advantage of returning vectors in the correct tangent space.

Piecing these two observations together yields the following cost function:

$$E(\gamma) = \frac{1}{2}\sum_{i=1}^{N} w_i \left\|\gamma_{s_i} - p_i\right\|^2 + \frac{\lambda}{\Delta\tau}\frac{1}{2}\sum_{i=1}^{N_\mathrm{d}-1} \left\|\gamma_i - \gamma_{i+1}\right\|^2 + \frac{\mu}{\Delta\tau^3}\frac{1}{2}\sum_{i=2}^{N_\mathrm{d}-1} \left\|\operatorname{skew}\big(\gamma_i^\top(\gamma_{i+1} + \gamma_{i-1})\big)\right\|^2. \quad (7)$$

This is the cost function used in the present paper. Notice that, as the discretization becomes finer with $N_\mathrm{d} \to \infty$, $\Delta\tau \to 0$, this cost function converges (in some natural sense) to the continuous formulation (1).

Interestingly, one obtains the same simplified cost function by exploiting the fact that the finite differences (2) and (4) do make sense in the embedding space $\mathbb{R}^{n\times n}$. Then, $\dot\gamma_i$ is defined as the (not tangent) vector $\dot\gamma_i = (\gamma_{i+1} - \gamma_i)/\Delta\tau$ and $\ddot\gamma_i$ is defined as the (tangent) vector $\ddot\gamma_i = \mathrm{P}_{\gamma_i}(\gamma_{i+1} - 2\gamma_i + \gamma_{i-1})/\Delta\tau^2$.

Differentiating E. Consider the two functions f and g below:

$$f\colon \mathrm{SO}(n)^2 \to \mathbb{R}, \qquad f(A,B) = \frac{1}{2}\left\|A - B\right\|^2,$$
$$g\colon \mathrm{SO}(n)^3 \to \mathbb{R}, \qquad g(A,B,C) = \frac{1}{2}\left\|\operatorname{skew}\big(A^\top(B+C)\big)\right\|^2.$$

The cost E (7) is a linear combination of f and g with the γ_i's as input. Hence, it suffices to provide formulas for the gradient and Hessian of f and g to obtain the gradient and Hessian of E.

Let $\nabla f(A,B)$ denote the Euclidean gradient of f, that is, the gradient of f seen as a function on $(\mathbb{R}^{n\times n})^2$. Similarly, let $\nabla^2 f(A,B)$ denote the Euclidean Hessian of f. Obviously,

$$\nabla f(A,B) = (A - B, B - A), \qquad \nabla^2 f(A,B)[\dot A, \dot B] = (\dot A - \dot B, \dot B - \dot A). \quad (8)$$

These Euclidean quantities may be transformed into their Riemannian counterparts by exploiting the fact that $\mathrm{SO}(n)$ is a Riemannian submanifold of $\mathbb{R}^{n\times n}$. Indeed, it holds for a function $h\colon \mathrm{SO}(n) \to \mathbb{R}$ that the Riemannian gradient $\operatorname{grad} h(A)$ is obtained from the Euclidean gradient $\nabla h(A)$ via orthogonal projection on the tangent space at A [2, § 3.6.1], that is,

$$\operatorname{grad} h(A) = A\operatorname{skew}\big(A^\top \nabla h(A)\big). \quad (9)$$

Similarly, the Hessian of h at A along a tangent vector $A\Omega$—Ω is skew-symmetric—is obtained by projecting the directional derivative of the Riemannian gradient back to the tangent space [2, § 5.3.3]:

$$\operatorname{Hess} h(A)[A\Omega] = A\operatorname{skew}\big(A^\top \nabla^2 h(A)[A\Omega] - \Omega\operatorname{sym}\big(A^\top \nabla h(A)\big)\big). \quad (10)$$

Applying identities (9) and (10) to (8) (entry-wise) yields explicit formulas for the Riemannian gradient and Hessian of f. Similarly for the function g, the

Euclidean gradient and Hessian are:

$$\nabla g(A,B,C) = \big(-(B+C)R, AR, AR\big),$$
$$\nabla^2 g(A,B,C)[\dot{A},\dot{B},\dot{C}] = \Big(-(\dot{B}+\dot{C})R - (B+C)\dot{R}, \dot{A}R + A\dot{R}, \dot{A}R + A\dot{R}\Big),$$

with $R = \mathrm{skew}\big(A^\top(B+C)\big)$ and $\dot{R} = \mathrm{skew}\Big(\dot{A}^\top(B+C) + A^\top(\dot{B}+\dot{C})\Big)$. Again, the identities (9) and (10) provide an explicit means of transforming ∇g and $\nabla^2 g$ into their Riemannian counterparts.

3 Trust-regions to minimize the cost

In order to minimize the cost function E (7), we use the *Riemannian trust-regions method* (RTR) [1]. We work with Manopt [5] (`http://www.manopt.org`), a Matlab toolbox for optimization on manifolds. The implementation of the RTR algorithm in Manopt is an adaptation of the original GenRTR code [1].

Using the Manopt toolbox, a number of steps required for the usage of optimization algorithms on manifolds are simplified. In particular, the geometry of the manifold $\mathrm{SO}(n)^{N_\mathrm{d}}$ is already included in the toolbox, so that we do not need to provide functions for the metric, the exponential and logarithmic maps etc. Conveniently, the identities (9) and (10) are built in the toolbox too, so that it is sufficient to provide code for the Euclidean gradient and Hessian. Diagnostics tools are included to help the user verify that the code is correct.

The RTR method requires code for the cost, its gradient and its Hessian, which we worked out in the previous section. It also requires an initial guess (an initial iterate), which we describe below. Reasons for choosing GenRTR include the strong theoretical guarantees (global convergence toward critical points and quadratic convergence speed in the neighborhood of critical points) as well as its excellent applicative track record.

All the default parameter values for GenRTR are used, except for the following two. First, the maximum trust-region radius $\bar{\Delta}$ is set to $\pi\sqrt{nN_\mathrm{d}}$ (and the initial trust-region radius is set to $\Delta_0 = \bar{\Delta}/8$, as usual). This scales as the maximum distance between two points on the manifold $\mathrm{SO}(n)^{N_\mathrm{d}}$. Second, the limit on the number of Hessian evaluations is set to three times the dimension of the search space, that is, $3\frac{n(n-1)}{2}N_\mathrm{d}$. In theory, to solve a linear system involving the Hessian using a conjugate gradient algorithm, it should be sufficient to authorize up to as many Hessian evaluations as the dimension of the manifold. In practice though, more iterations are sometimes required, possibly due to the finite precision of the computations. We authorize many Hessian evaluations in order to exhibit the role of second-order information in solving the present problem.

An initial guess. The search space $\mathrm{SO}(n)^{N_\mathrm{d}}$ is not convex, and there is thus no guarantee that GenRTR will find a global minimizer of E. In order to increase the chances of finding a local minimizer of good quality, the initial guess should

be chosen as close as possible to the global minimizer (which is of course unknown). The proposed heuristic is to choose the initial rotations γ_i according to a piecewise-geodesic interpolation of the data points p_i.

A refinement procedure. For positive values of the second-order regularity parameter, $\mu > 0$, the various optimization algorithms we tried exhibit relatively slow convergence while far away from a critical point. In order to work with a large number N_d of discretization points, it is helpful to first compute a solution with small N_d; this solution can then be refined by sampling the piecewise-geodesic curve linking the obtained points γ_i. Doing so provides an excellent initial guess for the problem with a larger value of N_d. This procedure can be iterated to produce solutions with large N_d at reasonable cost.

4 Numerical experiments

We perform two experiments to illustrate the following points: (i) the new cost function is appropriate to produce smooth curves on $\mathrm{SO}(n)$, and (ii) trust-region methods are far more efficient than the conjugate gradient algorithm proposed in [4] for the purpose of regression on $\mathrm{SO}(n)$. These points are illustrated in Figures 1 and 2 respectively, with the corresponding experimental setups described in the captions. In both cases, there are $N = 4$ data points (the same as in [4]) with time labels $[t_1, t_2, t_3, t_4] = [0, 1/3, 2/3, 1]$ and identical weights $w_i = 1$.

A possible explanation of the success of second-order methods for the problem at hand is the large condition number of the Hessian at the solution. For $\mu = 10^{-3}, 10^{-2}, 10^{-1}$ resp., the condition number is $6.1 \cdot 10^5, 7.1 \cdot 10^6, 7.9 \cdot 10^7$. Interestingly, it can also be seen from Figure 2 that the same trust-region method without knowledge of the Hessian, where the Hessian is approximated by finite differences of the gradient (RTR-FD), performs well too, so that all the credit cannot be given to knowledge of the Hessian alone.

Acknowledgment. This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. NB is an F.R.S.-FNRS Research Fellow.

References

1. Absil, P.A., Baker, C., Gallivan, K.: Trust-region methods on Riemannian manifolds. Foundations of Computational Mathematics 7(3), 303–330 (2007)
2. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton, NJ (2008)
3. Absil, P.A., Malick, J.: Projection-like retractions on matrix manifolds. SIAM Journal on Optimization 22(1), 135–158 (2012)
4. Boumal, N., Absil, P.A.: A discrete regression method on manifolds and its application to data on SO(n). In: Proceedings of the 18th IFAC World Congress (Milan). vol. 18, pp. 2284–2289 (2011)

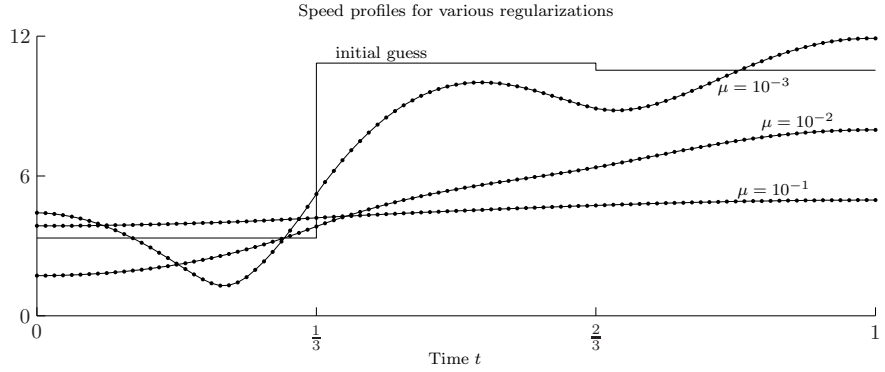

Fig. 1. Minimizing E (7) produces smooth-looking discrete regression curves on $\mathrm{SO}(n)$. This plot represents the speed profiles $\|\dot{\gamma}_i\|$ (3) of the obtained regression curves with $N_\mathrm{d} = 97$ discretization points for $\lambda = 0$ and various values of μ. The larger μ is, the more we insist on smoothness. The smaller μ is, the more we insist on passing close to the $N = 4$ data points p_i. The initial guess is piecewise-geodesic, hence its piecewise-constant speed profile.

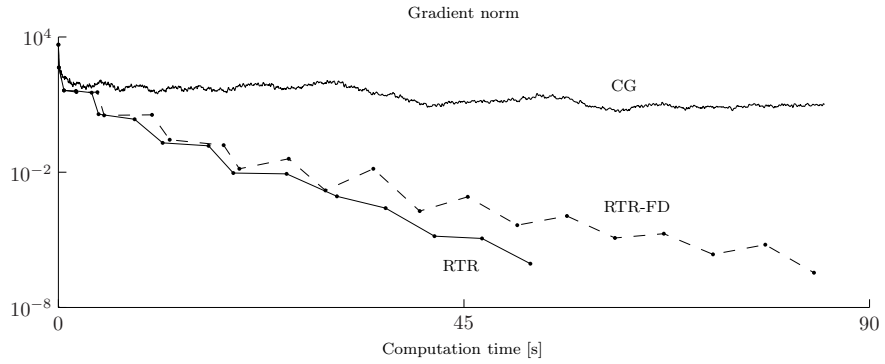

Fig. 2. The Riemannian trust-region method with exact Hessian information (RTR) widely outperforms the conjugate gradient method (CG) used in [4] for minimizing the cost E (7). The plot shows the convergence profile of both RTR and CG with $N_\mathrm{d} = 97$ discretization points and regularization parameters $\lambda = 0$ and $\mu = 10^{-2}$. The RTR-FD curve shows the convergence profile of the RTR method when the Hessian is approximated using finite differences of the gradient of E (an algorithm for which there is currently no local convergence theory).

5. Boumal, N., Mishra, B.: The Manopt toolbox. `http://www.manopt.org` (2013), version 1.0.1
6. Jia, C., Evans, B.: 3D rotational video stabilization using manifold optimization. `http://users.ece.utexas.edu/~bevans/papers/2013/stabilization/index.html` (2013)
7. Samir, C., Absil, P.A., Srivastava, A., Klassen, E.: A gradient-descent method for curve fitting on Riemannian manifolds. Foundations of Computational Mathematics 12(1), 49–73 (2012)