IMA Journal of Numerical Analysis (2015) Page 1 of 13
doi:10.1093/imanum/drnxxx

Fast closest logarithm algorithm in the special orthogonal group

ADRIEN ESCANDE[†]
CNRS-AIST JRL (JOINT ROBOTICS LABORATORY), UMI3218/RL
AIST CENTRAL 2, 1-1-1 UMEZONO, TSUKUBA, IBARAKI 305-8568 JAPAN

[Received on 11 May 2015]

For interpolating between elements of $\mathrm{SO}(n)$, it is attractive to work in $\mathfrak{so}(n)$, passing from one space to the other via the exponential map. However, the logarithm is a multi-valued map and the choice of a particular image affects the quality of the interpolation. In this paper, we propose a fast and accurate algorithm to compute the image that seems the most appropriate for interpolation: given $Q \in \mathrm{SO}(n)$ and $A \in \mathfrak{so}(n)$, our algorithm returns the logarithm of Q which is the closest to A, under minimal conditions on Q. We carefully study the mathematical properties of our problem to establish the algorithm, discuss its implementation and demonstrate its efficiency.

Keywords: Interpolation, special orthogonal group, logarithm

1. Introduction

Given real numbers $t_1 < t_2 < \cdots < t_p$ and a sequence $(Q_1, Q_2, \ldots, Q_p)$ of elements of $\mathrm{SO}(n)$ $(n \geqslant 2)$, the special orthogonal group of dimension n, let us consider the following interpolation problem, which has applications in many engineering fields:

$$\begin{aligned} \text{find} \quad & Q : [t_1, t_p] \longrightarrow \mathrm{SO}(n) \\ \text{such that} \quad & Q(t_i) = Q_i \quad i = 1..p \end{aligned} \tag{1.1}$$

with Q being continuous and derivable up to a given level.

This is a not trivial problem since $\mathrm{SO}(n)$ is a non-Euclidean space. A classical way to solve it is to make use of the exponential map and work in $\mathfrak{so}(n)$, the Lie algebra associated to $\mathrm{SO}(n)$. Indeed, $\mathfrak{so}(n)$ is a vector space that can be seen as a subspace of $\mathrm{M}_n(\mathbb{R})$, the space of $n \times n$ matrix over $\mathbb{R}$. Thus, having found matrices $A_i \in \mathfrak{so}(n)$ such that $e^{A_i} = Q_i$ for $i = 1..p$, we can interpolate component by component with usual algorithms to get an interpolation function $A : [t_1, t_p] \longrightarrow \mathfrak{so}(n)$ satisfying $A(t_i) = A_i$. Then

$$Q(t) = e^{A(t)} \tag{1.2}$$

is a solution to the problem (1.1). The smoothness properties of Q derive from those of A thanks to the exponential function.

The difficulty of this approach comes from the fact that the logarithm map is multi-valued: there are infinitely many matrices A_i (we enumerate them later) such that $e^{A_i} = Q_i$. A reasonable choice is to select the A_i to minimize the variations of the interpolating function. These A_i do not however necessarily

[†]Corresponding author. Email: adrien[dot]escande[at]gmail[dot]com.
This work was partially funded by a grant from the Japan Society for Promotion of Science (JSPS; Grant-in-Aid for JSPS Fellows P13786) and the FP7-ICT-2013-10/611909 KOROIBOT Project.

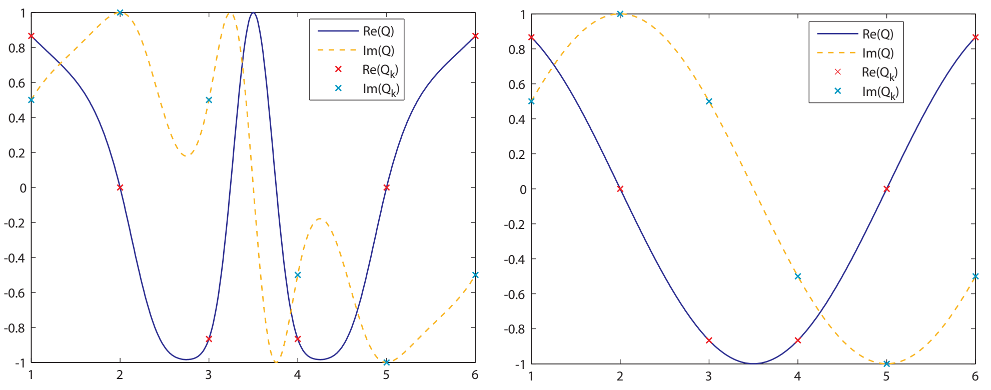


FIG. 1. Two possible interpolations for the key points given in eq. (1.3). The interpolating function $Q(t)$ is depicted by its real (plain blue curve) and imaginary (dashed orange curve) parts. In the left picture, Q is obtained by interpolating between the principal logarithms of $(Q_1,\ldots,Q_6)$. For the right picture, a more adequate choice of logarithms was made. Red and indigo markers show the real and imaginary parts of the key points.

correspond to the principal branch of the logarithm as we illustrate with the simple following example in SO(2), that we identify with the unit circle in the complex plane: let us interpolate the sequence

$$(Q_1,\ldots,Q_6) = \left(\frac{\sqrt{3}}{2}+\frac{1}{2}\mathbf{i}, \mathbf{i}, -\frac{\sqrt{3}}{2}+\frac{1}{2}\mathbf{i}, -\frac{\sqrt{3}}{2}-\frac{1}{2}\mathbf{i}, -\mathbf{i}, \frac{\sqrt{3}}{2}-\frac{1}{2}\mathbf{i}\right) \tag{1.3}$$

for $t_i = 1..6$, where $\mathbf{i}$ is the imaginary unit.
The principal logarithms are $(\mathbf{i}\pi/6, \mathbf{i}\pi/2, 5\mathbf{i}\pi/6, -5\mathbf{i}\pi/6, -\mathbf{i}\pi/2, -\mathbf{i}\pi/6)$, but the expected interpolation is obtained with (for example) $(\mathbf{i}\pi/6, \mathbf{i}\pi/2, 5\mathbf{i}\pi/6, 7\mathbf{i}\pi/6, 3\mathbf{i}\pi/2, 11\mathbf{i}\pi/6)$, as show in Fig. 1.

The choice of the appropriate logarithm is rather obvious in this example (and more generally in SO(2)): for Q_1, any choice is valid, in particular the principal logarithm. Then given a choice A_i for Q_i, we chose among the possible A_{i+1} the one which is the closest to A_i, so that the distance between two consecutive A_i is at most π.

In this paper, we want to extend this approach to SO(n). To do so, we focus on the following problem: for $Q \in \mathrm{SO}(n)$ and $A \in \mathfrak{so}(n)$, compute $X = \log_A(Q)$ defined as the (if unique) element of $\mathfrak{so}(n)$ the closest to A which is a logarithm of Q:

$$\begin{array}{cc} \underset{X\in\mathfrak{so}(n)}{\min.} & \frac{1}{2}\|X-A\|_F^2 \\ s.t. & e^X = Q \end{array} \tag{1.4}$$

where $\|.\|_F$ denotes the Frobenius norm.

The problem of computing the adequate logarithm was already addressed in Shingel (2009), in which the author proposes an iterative Newton-like algorithm to approximate the closest logarithm with quadratic convergence. Each iteration requires to build a so-called reduced commutator matrix (see. Bloch & Iserles (2005)) of dimension $m = n(n-1)/2$, compute a function of it using Padé approximation, then solve a $m \times m$ linear system. Each iteration has thus a complexity of $O(n^6)$, so that the algorithm can only be applied for small values of n. Furthermore, the algorithm converges only when Q_{k+1} is close enough of Q_k. This proved to be a serious limitation when we wanted to use random Q_k.

We propose here an algorithm in $O(n^3)$ valid almost everywhere on $\mathrm{SO}(n)$. This algorithm is based on two steps. We first give an expression of all the logarithms of an element of $\mathrm{SO}(n)$. Except in particular cases, there are countable many possibilities, so that we can rewrite problem (1.4) as an optimization problem on $\mathbb{Z}^r$ (for some r defined later), known as the Closest Lattice Vector Problem (CVP) (see Agrell *et al.* (2002); Hanrot *et al.* (2011)). We then show that, despite this problem being NP-hard in the general case, we can compute an exact solution very easily, thanks to a particularity of our case.

The paper is organized as follows: in section 2, we give the definitions and theorems necessary to establish our algorithm. The algorithm itself is written in section 3 along with implementation considerations, while the proofs of the theorems are delayed to section 4. We then present some experimental results and validations (section 5) and conclude the paper (section 6). Some of the longer proofs are given in the appendices.

In the rest of the paper, we adopt the following notations: $\mathrm{M}_n(\mathbb{K})$ denotes the set of n-by-n matrices over the field $\mathbb{K}$, and 0_n and I_n are its zero and identity elements. $\langle .,. \rangle$ is the Frobenius inner product and, with $\mathrm{vec}(M)$ the vectorization of the matrix M, i.e. its representation as a vector of its elements, $\langle M_1, M_2 \rangle = \mathrm{vec}(M_1)^T \mathrm{vec}(M_2)$. M^* is the conjugate transpose of M. $\lfloor . \rfloor$ denotes the floor function, and $\mathrm{round}(.)$ the rounding to the nearest integer.

2. Main results

Problem (1.4) is a constrained least-square program over $\mathfrak{so}(n)$. We first transform it to an unconstrained program by finding a general expression for the solutions of $e^X = Q$ for $Q \in \mathrm{SO}(n)$ and $X \in \mathfrak{so}(n)$.

The real Schur decomposition of Q is $Q = UDU^T$ with $U \in \mathrm{O}(n)$ and D a block diagonal matrix (see Horn & Johnson (2013, Corollary 2.5.11(c))):

$$D = \begin{bmatrix} D_1 & & & \\ & \ddots & & \\ & & D_r & \\ & & & I_{\text{odd}} \end{bmatrix}, \quad D_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \in \mathrm{SO}(2) \tag{2.1}$$

with I_{odd} an empty matrix if n is even or $I_{\text{odd}} = I_1$ if n is odd, and r = $\lfloor n/2 \rfloor$. U can be chosen so that $0 \leqslant \theta_i \leqslant \pi$ and the D_i appear by decreasing order of θ_i. The first matrices D_i groups the -1 eigenvalues (if any) which always occur by pair and correspond to $\theta_i = \pi$. Likewise, the $+1$ eigenvalues are grouped 2-by-2 in the last D_i with $\theta_i = 0$, leaving a single eigenvalue 1 alone if n is odd.

Let us then define the following matrices:

$$F_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \bar{F}_i = \begin{bmatrix} 0_{2i-2} & & \\ & F_0 & \\ & & 0_{n-2i} \end{bmatrix}, \quad \text{and } X_i = U\bar{F}_i U^T, \quad i = 1..r \tag{2.2}$$

so that $D_i = e^{\theta_i F_0}$, $D = e^{\sum \theta_i \bar{F}_i}$ (since the $\bar{F}_i$ commute), and $Q = Ue^{\sum \theta_i \bar{F}_i}U^T = e^{U(\sum \theta_i \bar{F}_i)U^T} = e^{\sum \theta_i X_i}$. Denoting u_i the i-th column of U, note that X_i is simply obtained by $X_i = u_{2i}u_{2i-1}^T - u_{2i-1}u_{2i}^T$.

Throughout this paper we consider the following hypothesis for Q:

$$\mathscr{H}: \quad \theta_i \neq \theta_j \text{ for all } 1 \leqslant i \neq j \leqslant r \quad \text{and} \quad \theta_r \neq 0 \text{ if } n \text{ is odd}$$

The second part of the hypothesis implies that Q has no more than 2 eigenvalues equal to 1.
$\mathscr{H}$ is satisfied almost everywhere on $\mathrm{SO}(n)$.

Our first main result is the next theorem:

THEOREM 2.1 Let $Q \in \mathrm{SO}(n)$ satisfying $\mathscr{H}$ and U, X_i and r defined as above. Then all the real skew-symmetric solutions to $e^X = Q$ are given by

$$X = \sum_{i=1}^{r} (\theta_i + 2k_i\pi)X_i, \quad (k_1, \ldots, k_r) \in \mathbb{Z}^r \tag{2.3}$$

With this result, problem (1.4) becomes the following unconstrained program, where $X_0 = \sum \theta_i X_i$:

$$\min_{k\in\mathbb{Z}^r} \frac{1}{2} \left\| X_0 + 2\pi \sum_{i=1}^{r} k_i X_i - A \right\|_F^2 \tag{2.4}$$

which can be solved easily thanks to a property of the X_i, giving us our second main results:

THEOREM 2.2 Let $Q \in \mathrm{SO}(n)$ satisfying $\mathscr{H}$, $A \in \mathfrak{so}(n)$ and $X_0 \in \mathfrak{so}(n)$ such that $e^{X_0} = Q$. A solution to problem (1.4) is

$$X = X_0 + 2\pi \sum_{i=1}^{r} k_i^* X_i \quad \text{with } k_i^* = \mathrm{round}\left(\frac{\langle X_i, A - X_0\rangle}{4\pi}\right) \tag{2.5}$$

This solution is unique if for $i = 1..r$, $\frac{1}{2}\langle X_i, A - X_0\rangle \not\equiv \pi \pmod{2\pi}$.

We conclude this section with the remark that $log_0(Q)$ is the principal logarithm of Q:

THEOREM 2.3 Take $A = 0$. If $\mathscr{H}$ holds and Q has no -1 eigenvalue, the solution to problem (1.4) is the principal logarithm of Q.

3. Implementation

To solve problem (1.4), we need to compute first X_0 and the X_i, then apply eq. (2.5). Matrices X_i also appear in Gallier & Xu (2003) where they are obtained by solving $m = n(n-1)/2$ systems of size r with the same Vandermonde matrix. This matrix quickly becomes very poorly conditioned as n increases, so that the computation of the X_i is completely wrong when n is more than a few units. On the contrary, performing a real Schur decomposition of Q and computing X_0 and the X_i from it is both a fast and numerically stable solution. The final algorithm can be sum up as follows:

Algorithm 3.1 $\log_A(Q)$

```
(i) compute the real Schur decomposition $Q = UDU^T$ with $D$ as in eq. (2.1)
(ii) let $X_i = u_{2i}u_{2i-1}^T - u_{2i-1}u_{2i}^T$ for $i = 1..r$
(iii) compute the angle $\theta_i$ corresponding to $D_i$ for $i = 1..r$
(iv) let $X_0 = \sum \theta_i X_i$
(v) let $k_i = \mathrm{round}\left(\frac{\langle X_i, A-X_0\rangle}{4\pi}\right)$ for $i = 1..r$
(vi) return $X = X_0 + 2\pi \sum k_i X_i$
```

In practice, there are few chances that a built-in Schur decomposition routine gives D with the required structure, as the Schur decomposition is only unique up to a permutation of the diagonal blocks and the signs of the off-diagonal elements (the sinuses) is arbitrary. In particular, the -1 and $+1$ eigenvalues might be mixed together or with the 2×2 blocks. There will be the need to apply proper permutations and reflexions. Numerically, it is necessary to decide when a diagonal value $D_{i,i}$ is considered as a -1 or $+1$ eigenvalue. Instead of comparing directly the value to ± 1 with a given precision, it is more precise to look at the neighbor values, i.e. to test if $|D_{i+1,i}| \leqslant \varepsilon$ and $|D_{i,i+1}| \leqslant \varepsilon$, with ε a user-specified threshold (we typically take 5×10^{-14} for 64-bit floats). This is because, numerically, $\cos\theta = 1$ as soon as $|\theta| < \sqrt{2\varepsilon_m}$, with ε_m the machine precision, while $\sin\theta$ has the same precision as θ.

The proper implementation of step (iii) is of course to use a built-in *atan2* function, *i.e.* taking $\theta_i = \operatorname{atan2}(D_{2i-1,2i}, D_{2i-1,2i-1})$. The Frobenius inner product of step (v) should be implemented as $\operatorname{vec}(X_i)^T \operatorname{vec}(A - X_0)$: the vectorization is costless as matrices are internally represented by 1-dimensional arrays, so the computation is simply a dot product.

The algorithm should check the properties of Q during its run: if Q is not in $\mathrm{SO}(n)$, an error must be returned; if $\mathscr{H}$ does not hold, a warning can be printed. The former can be checked cheaply after the Schur decomposition in step (i), from the form of D and ensuring that each D_i is in $\mathrm{SO}(2)$. The latter is to be checked after step (iii).

If $A = 0$, the algorithm can be stopped at step (iv) and return X_0, which is then the principal logarithm of Q if the hypothesis of Theorem 2.3 holds. In this case, the implementation of the algorithm in Matlab is 5 to 20 times faster than the built-in *logm* function. This is not surprising since *logm* does not take into account the fact that Q is an element of $\mathrm{SO}(n)$.

Theoretically, the most costly step of the algorithm is the Schur decomposition whose complexity is $O(n^3)$ (see Golub & Van Loan (1996, p. 359)). Steps (ii) and (v) are r repetitions of an operation in $O(n^2)$, so in $O(n^3)$ as well. The accumulations in steps (iv) and (vi) are also in $O(n^3)$. Lastly, step (iii) is in $O(n)$. The algorithm has thus a complexity of $O(n^3)$.

4. Proofs of the main results

4.1 *Commuting matrices*

The proof of Theorem 2.1 relies heavily on matrices commuting with D and $F = \sum(\theta_i + 2k_i\pi)\bar{F}_i$. We restrict to the case where $0 \leqslant \theta_i \leqslant \pi$ and $\mathscr{H}$ is satisfied to give the characteristics of such matrices.

With $\mathscr{M}$ denoting the set $\left\{ \begin{bmatrix} m_1 & -m_2 \\ m_2 & m_1 \end{bmatrix}, (m_1, m_2) \in \mathbb{C}^2 \right\}$, we have the two following theorems:

THEOREM 4.1 Let D be defined as in eq. (2.1) with $0 \leqslant \theta_i \leqslant \pi$ and $\mathscr{H}$ satisfied. $M \in \mathrm{M}_n(\mathbb{C})$ commutes with D if and only if (*iff*)

$$M = \begin{bmatrix} M_1 & & & \\ & \ddots & & \\ & & M_r & \\ & & & \alpha I_{\mathrm{odd}} \end{bmatrix}, \quad \text{with } \alpha \in \mathbb{C} \text{ and } \begin{cases} M_i \in \mathrm{M}_2(\mathbb{C}) & \text{if } \theta_i = 0 \text{ or } \theta_i = \pi \\ M_i \in \mathscr{M} & \text{otherwise} \end{cases} \tag{4.1}$$

THEOREM 4.2 Let F be defined as above, with $0 \leqslant \theta_i \leqslant \pi$ and $\mathscr{H}$ satisfied. $M \in \mathrm{M}_n(\mathbb{C})$ commutes with F *iff*

$$M = \begin{bmatrix} M_1 & & & \\ & \ddots & & \\ & & M_r & \\ & & & \alpha I_{\mathrm{odd}} \end{bmatrix}, \quad \text{with } \alpha \in \mathbb{C} \text{ and } \begin{cases} M_i \in \mathrm{M}_2(\mathbb{C}) & \text{if } \theta_i = 0 \text{ and } k_i = 0 \\ M_i \in \mathscr{M} & \text{otherwise} \end{cases} \tag{4.2}$$

The proof of both theorems is given in Appendix A.

It is also useful to study under which similarity a skew-symmetric matrix remains skew-symmetric:

LEMMA 4.1 Let A be a non-zero skew-symmetric matrix, and U a non-singular matrix. UAU^{-1} is skew-symmetric *iff* U^TU commutes with A

Proof. $\left(UAU^{-1}\right)^T = -UAU^{-1} \iff U^{-T}A^TU^T = -UAU^{-1} \iff U^{-T}AU^T = UAU^{-1} \iff AU^TU = U^TUA$ □

LEMMA 4.2 Let $A \in \mathfrak{so}(2), A \neq 0$. For U non-singular matrix, UAU^{-1} is skew-symmetric *iff* $U \in \mathscr{M}$

Proof. From the previous lemma, U^TU commutes with A so $U^TU \in \mathscr{M}$. Since U^TU is symmetric, this implies that $U^TU = aI_2$ for some a. We thus have $U = bV$ with $b^2 = a$ and V a unitary matrix, therefore $U \in \mathscr{M}$. Conversely, for any non-singular matrix U in $\mathscr{M}$, UAU^{-1} is skew-symmetric. □

4.2 *Enumeration of solutions for $e^X = Q$*

In this section, we prove Theorem 2.1. The starting point is a theorem from Gantmacher (1959, p. 241):

THEOREM 4.3 For a non-singular complex matrix Q with Jordan canonical form ZJZ^{-1} and $J = \mathrm{diag}(J_1, \dots, J_r)$, the general solution to the equation $e^X = Q$ is

$$X = ZM \begin{bmatrix} L_1 & & \\ & \ddots & \\ & & L_r \end{bmatrix} M^{-1}Z^{-1}, \quad \text{with } L_i = log(J_i) + 2\mathbf{i}k_i\pi I_{r_i},\ k_i \in \mathbb{Z} \tag{4.3}$$

where r_i is the size of i-th Jordan block J_i, $log(J_i)$ is a logarithm of J_i (typically the principal logarithm when defined) and M is a non-singular (complex) matrix which commutes with J.

Culver (1966) examines the conditions for the form (4.3) to be real and possibly unique. In this section, we adapt the solution (4.3) to the case of $Q \in \mathrm{SO}(n)$ satisfying $\mathscr{H}$ and $X \in \mathfrak{so}(n)$.

For $Q \in \mathrm{SO}(n)$, the Jordan canonical form is simply the eigenvalue decomposition. Introducing $P_0 = \sqrt{2}/2 \begin{bmatrix} \mathbf{i} & -1 \\ 1 & -\mathbf{i} \end{bmatrix}$ and $P = \mathrm{diag}(P_0, \cdots, P_0, I_{\mathrm{odd}})$, which are both unitary matrices, this decomposition can be obtained from the real Schur decomposition as $Q = UPCP^*U^T$ with

$$C = P^*DP = \begin{bmatrix} C_1 & & & \\ & \ddots & & \\ & & C_r & \\ & & & I_{\mathrm{odd}} \end{bmatrix}, \quad C_i = \begin{bmatrix} \cos\theta_i + \mathbf{i}\sin\theta_i & 0 \\ 0 & \cos\theta_i - \mathbf{i}\sin\theta_i \end{bmatrix} \tag{4.4}$$

Let E_{odd} be the empty matrix if n is even and $\left[2\mathbf{i}k_{\mathrm{odd}}\pi\right]$ otherwise, $k_{\mathrm{odd}} \in \mathbb{Z}$. By applying Theorem 4.3 to eq. (4.4) we get the form of the solution $X = UPMEM^{-1}P^*U^T$ where M is a non-singular matrix commuting with C and

$$E = \begin{bmatrix} E_1 & & & \\ & \ddots & & \\ & & E_r & \\ & & & E_{\mathrm{odd}} \end{bmatrix}, \quad E_i = \mathbf{i}\begin{bmatrix} \theta_i + 2k_i\pi & 0 \\ 0 & -\theta_i + 2k_i'\pi \end{bmatrix}, \quad (k_i, k_i') \in \mathbb{Z}^2 \tag{4.5}$$

X cannot be real skew symmetric unless its eigenvalues are either 0 or purely imaginary and occurring by conjugate pairs. When $\mathscr{H}$ holds, this implies that $k_i' = -k_i$ and $E_{\text{odd}} = 0$. Then we define

$$F_i = P_0 E_i P_0^* = (\theta_i + 2k_i\pi)F_0 \quad \text{and} \quad F = \begin{bmatrix} F_1 & & & \\ & \ddots & & \\ & & F_r & \\ & & & 0_{\text{odd}} \end{bmatrix} = \sum(\theta_i + 2k_i\pi)\bar{F}_i \tag{4.6}$$

and the solution writes $X = UPMP^*FPM^{-1}P^*U^T$, for M non-singular matrix commuting with C. Noting that $MC = CM \iff MP^*DP = P^*DPM \iff PMP^*D = DPMP^*$, X can finally be expressed as $UMFM^{-1}U^T$ with M a non-singular matrix commuting with D. Such a matrix M is obtained with theorem 4.1.

The F_i are in $\mathfrak{so}(2)$ and thus F is in $\mathfrak{so}(n)$. Yet this last form of X may not be skew-symmetric, depending on the choice of M. According to Lemma 4.1, it is skew-symmetric *iff* $(UM)^T(UM) = M^TM$ commutes with F. M^TM has thus the form given by Theorem 4.2.

We can now state an intermediate result (which can actually be proven even if $\mathscr{H}$ does not hold, with a careful handling of k_i, k_i', k_j, k_j' when $\theta_i = \theta_j$):

THEOREM 4.4 Let $Q \in \mathrm{SO}(n)$ satisfying $\mathscr{H}$ and UDU^T be its real Schur decomposition. The real skew-symmetric solutions to the equation $e^X = Q$ are all the matrices $UMFM^{-1}U^T$ where F is given by eq. (4.6) with $(k_1, \dots, k_r) \in \mathbb{Z}^r$ and M is a non-singular matrix commuting with D and such that M^TM commutes with F.

A matrix M as described by this theorem has the form given by Theorem 4.1 with each M_i a non-singular element of $\mathscr{M}$ and $\alpha \neq 0$. Indeed, if $\theta_1 = \pi$, Theorem 4.1 implies than M_1 can be any matrix, but the requirement that M^TM commutes with F enforces that $M_1^TM_1$ commutes with F_1, yielding (Lemma 4.2) that $M_1 \in \mathscr{M}$. Same goes for M_r if $\theta_r = 0$. In the other cases, and for M_2 to M_{r-1}, Theorem 4.1 implies that $M_i \in \mathscr{M}$, and $M_i^TM_i$ automatically commutes with F_i.

According to Theorem 4.2, such a matrix M commutes with F. Then $X = UMFM^{-1}U^T = UFU^T = U\sum(\theta_i + 2k_i\pi)\bar{F}_iU^T = \sum(\theta_i + 2k_i\pi)X_i$ which proves Theorem 2.1.

For Q respecting $\mathscr{H}$, there is a countable infinity of solutions. To the contrary, if Q does not respect $\mathscr{H}$, the number of solutions is a non-countable infinity: only the diagonal blocks of M corresponding to unique θ_i commute with the associated F_i so that their choice does not matter. The choice of the others does impact the solution.

4.3 *Formulation as a CVP and solution*

In this section, we prove Theorem 2.2. We rely on the orthogonality of the X_i:

THEOREM 4.5 $\langle X_i, X_j \rangle = 2\delta_{i,j}$ ($\delta_{i,j}$ the Kronecker delta).

Proof. If $i = j$, $\langle X_i, X_j \rangle = \|X_i\|_F^2 = \|\bar{F}_i\|_F^2 = \|F_0\|_F^2 = 2$, using the fact that U is orthogonal to pass from X_i to $\bar{F}_i$ (see Horn & Johnson (2013, p.342)).
If $i \neq j$, we use the fact that on $\mathfrak{so}(n)$ $\langle A, B\rangle = -\operatorname{trace}(AB)$ and $X_iX_j = U\bar{F}_i\bar{F}_jU^T = 0$. □

Let us denote $A' = A - X_0$, $a' = \operatorname{vec}(A')$, and $N = \begin{bmatrix}\operatorname{vec}(X_1) & \cdots & \operatorname{vec}(X_r)\end{bmatrix}$. Since for any matrix M, $\|M\|_F^2 = \|\operatorname{vec}(M)\|_2^2$ and by linearity of the vec operator, we have $\|X_0 + 2\pi\sum k_iX_i - A\|_F^2 = \|\operatorname{vec}(X_0 + 2\pi\sum k_iX_i - A)\|_2^2 = \|2\pi\sum k_i \operatorname{vec}(X_i) - \operatorname{vec}(A')\|_2^2 = \|2\pi Nk - a'\|_2^2$.

Under hypothesis $\mathscr{H}$, problem (1.4) is thus ultimately reformulated as

$$\min_{k\in\mathbb{Z}^r} \left\|2\pi Nk - a'\right\|_2^2 \tag{4.7}$$

This latter formulation is known as the Closest Lattice Vector Problem (CVP) (see Agrell *et al.* (2002); Hanrot *et al.* (2011) for comprehensive surveys of the subject) and is NP-hard in the general case. In our particular case however, we have $N^T N = 2I_r$ as a consequence of Theorem 4.5, so that the solution of the problem is extremely simple to compute, using the following theorem:

THEOREM 4.6 Let $f(x) = x^T x - q^T x$ for some $q \in \mathbb{R}^r$. Let x_0 be the minimizer of f over $\mathbb{R}^r$. A global minimizer x^* of f over $\mathbb{Z}^r$ is obtained by rounding each component of x_0 to the nearest integer: $x^* = \text{round}(x_0)$.
If no component of x_0 has a decimal part equal to 0.5, the solution is unique.

Proof. See appendix B. □

Taking $q = \frac{1}{2\pi}N^T a'$, solving eq. (4.7) is equivalent to minimizing the quantity $k^T k - q^T k$ whose minimizer over $\mathbb{R}^r$ is $k_0 = \frac{1}{4\pi}N^T a'$ and thus, whose minimizer over $\mathbb{Z}^r$ is, according to Theorem 4.6,

$$k^* = \text{round}\left(\frac{1}{4\pi}N^T a'\right) = \left[\text{round}\left(\tfrac{1}{4\pi}\langle X_1, A - X_0\rangle\right) \quad \cdots \quad \text{round}\left(\tfrac{1}{4\pi}\langle X_r, A - X_0\rangle\right)\right]^T \tag{4.8}$$

Theorem 2.2 is a direct consequence of the above results. The uniqueness condition is obtained by writing that the decimal part of $\frac{1}{4\pi}\langle X_1, A - X_0\rangle$ must not be 0.5.

Note that X_0 can be any specific skew-symmetric solution to $e^X = Q$, not necessarily $\sum \theta_i X_i$ since shifting X_0 by $\sum 2k_i^0\pi$ for a given k^0 will yield a solution with $k^* - k^0$ instead of k^*.

If Q has no -1 eigenvalue, then $0 \leqslant \theta_i < \pi$ and $X_0 = \sum \theta_i X_i$ is the principal logarithm of Q. For $A = 0$, $\frac{1}{4\pi}\langle X_i, A - X_0\rangle = -\dfrac{1}{2pi}\theta_i$ due to the linearity of vec and Theorem 4.5, and $-\dfrac{1}{2} < -\dfrac{1}{2pi}\theta_i \leqslant 0$ so that $k_i^* = 0$. Thus $\log_0(Q)$ is the principal logarithm of Q, proving Theorem 2.3.

5. Experimental results

The above algorithm, as well as Shingel's one (see Shingel (2009, eq. (9))) were implemented and tested in Matlab on an Intel Core 2 Duo CPU at 2.53GHz, with 4Go of RAM. For both, we spent time to optimize the speed and make the implementation robust. For Shingel's algorithm, we used (4,4)-Padé approximations for evaluating the matrix function it relies on.

5.1 *Validity*

We first assert the validity of our algorithm with the following test: for a given matrix size n, we randomly take $r = \lfloor n/2 \rfloor$ values $0 \leqslant \theta_i < 100$ and form a matrix F as in eq. (4.6). We then generate a random orthogonal matrix U using the method of Stewart (1980) and compute the matrices $A = UFU^T$ and $Q = e^A$. Next, we generate a random matrix $R \in \mathrm{M}_n(\mathbb{R})$ from which we get the skew-symmetric matrix $B = R - R^T$. We take a random α, $0 \leqslant \alpha < \frac{\sqrt{2}\pi}{\|B\|_F}$ and form the matrix $A' = A + \alpha B$. We finally compute $L = log_{A'}(Q)$ with our algorithm and compare it with A. We should get $L = A$, what we justify now: taking the θ_i randomly ensures that $\mathscr{H}$ holds (even on finite-precision computer, the probability to get twice the same value is extremely low with a good pseudo-random generator). Furthermore, according to Lemma 5.1 below, the distance (in Frobenius norm) between two distinct solutions of

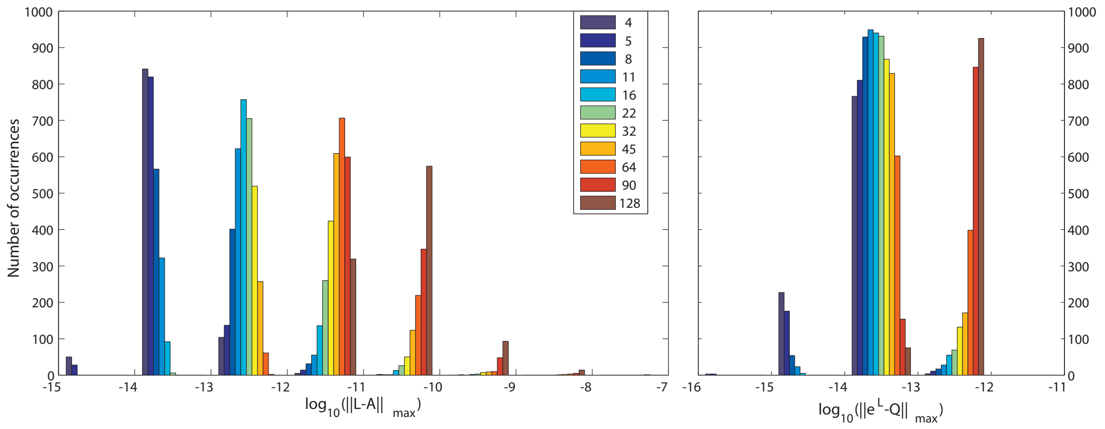


FIG. 2. Accuracy in the computation of $\log_{A'}(Q)$ for $n \in \{4,5,8,11,16,22,32,45,64,90,128\}$. With the error e being $\|L-A\|_{\max}$ (left) or $\left\|e^L-Q\right\|_{\max}$ (right), each bar of the histograms displays the number of times we got $10^i \leqslant e < 10^{i+1}$ over 1000 tests. $\log_{10}$ is the logarithm in base 10.

$e^X = Q$ is at least $2\sqrt{2}\pi$. Therefore, any matrix in the open ball of center A and radius $\sqrt{2}\pi$ is closer to A than to any other solution. With our upper bound on α, we ensure that the closest solution to A' is A: $\|A'-A\|_F = \alpha \|B\|_F < \sqrt{2}\pi$.

LEMMA 5.1 Let X' and X'' be two distinct solutions of $e^X = Q$. If $\mathscr{H}$ holds, $\|X'-X''\|_F \geqslant 2\sqrt{2}\pi$.

Proof. If $\mathscr{H}$ holds, all the solutions are given by eq. (2.3). Thus $X' = \sum(\theta_i + 2k'_i\pi)X_i$ and $X'' = \sum(\theta_i + 2k''_i\pi)X_i$ for some k' and k'' in $\mathbb{Z}$. We note $k = k' - k''$ and have $\|X'-X''\|_F = 2\pi\left\|\sum k_i X_i\right\|_F = 2\pi\left\|\sum k_i \bar{F}_i\right\|_F = 2\pi\sqrt{2\sum k_i^2} = 2\sqrt{2}\pi\|k\|_2$. The minimum norm for $k \neq 0$ is 1. □

We performed the test for different values of n from 4 to 128. For each n we repeated the test 1000 times. We asserted that we found the good solution by computing $\|L-A\|_{\max}$ and $\left\|e^L-Q\right\|_{\max}$. We chose the max norm ($\|M\|_{\max} = \max_{i,j}\left|m_{i,j}\right|$) instead of the Frobenius norm here, because the former, unlike the latter, is independent of the matrix size n. We recall that for $M \in \mathrm{M}_n(\mathbb{R})$, $\|M\|_{\max} \leqslant \|M\|_F \leqslant n\|M\|_{\max}$ (see Horn & Johnson (2013, Problem 5.6.P23)). The results are depicted by the histograms in Fig. 2. They show that the algorithm always converges to the correct solution and is very accurate: for the values of n considered, the error $\|L-A\|_{\max}$ is never more than 10^{-8} and mostly between 10^{-14} and 10^{-10}. It degrades with the size of the matrices. Even when L is not extremely close to A, it is numerically a very good estimate of $\log Q$ in that the error between e^L and Q is lower than 10^{-12}.

Our algorithm is therefore correct and accurate. The accuracy is not a surprise, since we rely on a Schur decomposition which is known to be numerically very stable.

5.2 *Computation time*

We also checked the performances of our algorithm, and compared them with those of classical built-in functions in Matlab (namely the matrix logarithm *logm* and the Schur decomposition *schur*) and our implementation of Shingel's algorithm. To do so, we took, for each matrix size n, the average computation time over 100 runs with matrices obtained randomly. Each algorithm was tested with the

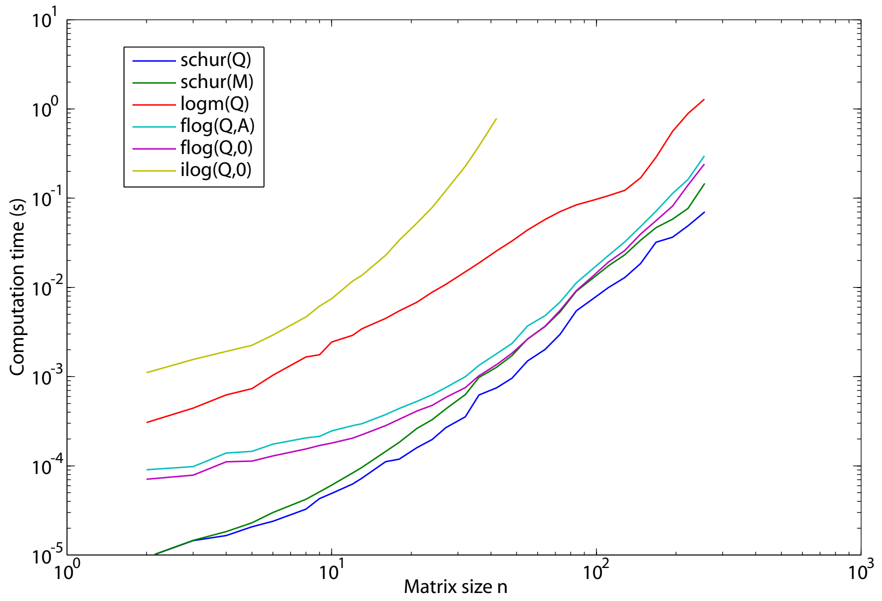


FIG. 3. Computation times for different algorithms and different matrix sizes. *schur* and *logm* are the built-in Matlab functions, *flog* is the algorithm of this paper, *ilog* is the algorithm in Shingel (2009). M, Q and A are matrices such that $M \in \mathrm{M}_n(\mathbb{R})$, $Q \in \mathrm{SO}(n)$ and $A \in \mathfrak{so}(n)$.

same set of random matrices (when applicable). As soon as n was more than a few units, it became too difficult to generate random pairs (Q,A) such that the computation of $\log_A(Q)$ with Shingel's algorithm would converge. Therefore, we restricted for this algorithm to the computation of $\log_0(Q)$, and generated random matrices Q until we got 100 working for a given size. While testing, it appeared that the Schur decomposition of a (special) orthogonal matrix Q was faster to obtain than the Schur decomposition of an ordinary one M, so we included the timings of both cases for better comparison.

The results of this test are depicted in Fig. 3 for the following cases:

(a) real Schur decomposition of $Q \in \mathrm{SO}(n)$,
(b) real Schur decomposition of $M \in \mathrm{M}_n(\mathbb{R})$,
(c) matrix logarithm of Q with Matlab's *logm*,
(d) $\log_A(Q)$ with our algorithm, $A \in \mathfrak{so}(n)$,
(e) $\log_0(Q)$ with our algorithm, stopping at step (iv) (short version),
(f) $\log_0(Q)$ with Shingel's algorithm.

Our algorithm (case (d)) is, as expected, much faster than Shingel's (case (f)): 16.8 times faster for $n = 6$, 286 times faster for $n = 36$, which is coherent with the complexities in n^3 and n^6. It is faster than *logm* by a factor $3 - 15$, while the short version (case (e)) is faster by a factor $5 - 20$.
Case (d) is slower than case (a) by a factor $7 - 9$ for the lowest dimensions ($n \leqslant 6$) but the ratio quickly drop to $2 - 3$ for higher dimensions: various overheads, in particular the cost of permuting the Schur decomposition to get D as in eq. (2.1), get negligible when n increases. At higher dimensions, the decomposition and the computation of the X_i (steps (i) and (ii)) account for $70\% - 75\%$ of the whole algorithm. Consequently, case (d) is only about 20% slower than case (e).

6. Conclusion

We proposed an algorithm that computes the real skew-symmetric logarithm of a special orthogonal matrix Q which is the closest to a given skew-symmetric matrix. To do so, we carefully characterized all the possible values for $\log(Q)$ and showed that under the hypothesis that Q has not twice the same imaginary eigenvalue and at most twice the same real eigenvalue, the possibilities are countable. We then rewrote the problem as an integer program that we showed to be trivially solvable. The obtained algorithm builds on a Schur decomposition and is both fast and accurate.

While the hypothesis $\mathscr{H}$ might seem restrictive at first, it is not in practice, since it is satisfied almost everywhere on $\mathrm{SO}(n)$, making the algorithm widely applicable. If $\mathscr{H}$ does not hold for a matrix Q_0, eq. (2.3) does not enumerate all the solution of $e^X = Q_0$ so that the algorithm will return a solution for problem (1.4) which is likely sub-optimal. This solution is the limit of the solution obtained for $Q_0 + \delta Q \in \mathrm{SO}(n)$ when δQ goes to 0_n[1].

The extension to all of $\mathrm{SO}(n)$ appears to be much more complex. However, it remains to precisely study the characteristics of matrices that both commutes with the matrices D and let the solutions to the problem $e^X = Q$ real skew-symmetric. Maybe some properties could be found that would simplify this extension.

While the motivation for this work is to interpolate on $\mathrm{SO}(n)$, minimizing the variations of the output, the paper in itself concentrates on solving problem (1.4). To our best knowledge, choosing $A_{i+1} = \log_{A_i}(Q_{i+1})$ does not necessarily yield the minimum varying interpolation between the Q_i, but is a good heuristic choice to avoid too many variations. We do not tackle here the problem of finding the minimum varying interpolation between the Q_i. However, we believe that the study we did here, especially on the enumeration of the logarithms, can be a good basis for addressing the problem.

REFERENCES

AGRELL, E., ERIKSSON, T., VARDY, A., VARDY, E. & ZEGER, K. (2002) Closest point search in lattices. *IEEE Transactions on Information Theory*, **48**, 2201–2214.

BLOCH, A. M. & ISERLES, A. (2005) Commutators of skew-symmetric matrices. *International Journal of Bifurcation and Chaos*, **15**, 793–801.

CULVER, W. J. (1966) On the existence and uniqueness of the real logarithm of a matrix. *Proceedings of the American Mathematical Society*, **17**, 1146–1151.

GALLIER, J. & XU, D. (2003) Computing exponentials of skew-symmetric matrices and logarithm of orthogonal matrices. *International Journal of Robotics and Automation*, **18**, 10–20.

GANTMACHER, F. R. (1959) *The theory of matrices*, vol. 1. AMS Chelsea Publishing.

GOLUB, G. & VAN LOAN, C. (1996) *Matrix computations*, 3rd edn. John Hopkins University Press.

HANROT, G., PUJOL, X. & STEHLÉ, D. (2011) Algorithms for the shortest and closest lattice vector problems. *Coding and Cryptology*. Lecture Notes in Computer Science, vol. 6639. Springer Berlin Heidelberg, pp. 159–190.

HORN, R. A. & JOHNSON, C. R. (2013) *Matrix Analysis*, 2nd edn. Cambridge University Press.

SHINGEL, T. (2009) Interpolation in special orthogonal groups. *IMA Journal of Numerical Analysis*, **29**, 731–745.

STEWART, G. W. (1980) The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, **17**, 403–409.

[1]Even when imposing $0 \leqslant \theta_i \leqslant \pi$ and the θ_i in decreasing order, the Schur decomposition $Q_0 = U_0 D_0 U_0^T$ is not unique (but D_0 is). However U_0 can be chosen so that the limit of the Schur decomposition of $Q_0 + \delta Q$ is $U_0 D_0 U_0^T$. Whatever the choice of U_0 our algorithm returns the same solution, hence this property.

Appendix A. Proof of Theorems 4.1 and 4.2

In all this section, D is such that $0 \leqslant \theta_i \leqslant \pi$ and $\mathscr{H}$ holds. We first detail the proof of Theorem 4.1.

If n is odd, D has the form $\begin{bmatrix} D' & \\ & 1 \end{bmatrix}$ and M is partitioned conformally as $\begin{bmatrix} M' & m_1 \\ m_2^T & \alpha \end{bmatrix}$. $MD = DM$ implies that $M'D' = D'M'$, $D'm_1 = m_1$, $D'^T m_2 = m_2$ and α can have any value. Because of $\mathscr{H}$, 1 is not an eigenvalue of D' so $m_1 = 0$ and $m_2 = 0$. We can thus restrict to studying M such that $MD = DM$ for n even.

We partition M in 2-by-2 blocks $M_{i,j}$. Solving $MD = DM$ is equivalent to solving r^2 2-by-2 systems $M_{i,j}D_j = D_iM_{i,j}$. For one of this system, we write $M_{i,j} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix}$, $c_k = \cos\theta_k$ and $s_k = \sin\theta_k$, $k = i, j$. We then get the system:

$$\begin{bmatrix} c_i - c_j & s_i & s_j & 0 \\ -s_i & c_i - c_j & 0 & s_j \\ -s_j & 0 & c_i - c_j & s_i \\ 0 & -s_j & -s_i & c_i - c_j \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} = 0 \tag{A.1}$$

Noting $S_{i,j}$ its matrix, we have $\det(S_{i,j}) = 4\,(\cos\theta_1 - \cos\theta_2)^2$. If $i \neq j$, $\theta_i \neq \theta_j$, thus $\det(S_{i,j}) \neq 0$, the only solution is zero and $M_{i,j} = 0$. Otherwise, for $i = j$, we have two cases:

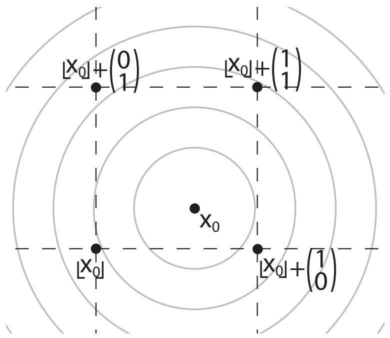
- $\theta_i = 0$ or $\theta_i = \pi$, so that $S_{i,i} = 0$, any vector is solution and $M_{i,i} \in \mathrm{M}_2(\mathbb{C})$
- $\theta_i \neq 0$, then solving the above systems yield $m_1 = m_4$ and $m_2 = -m_3$, i.e. $M_{i,i} \in \mathscr{M}$.

M is thus block diagonal and has the form of eq. (4.1), which proves Theorem 4.1.

The proof of Theorem 4.2 follows the same reasoning, remarking that for $i \neq j$, $\theta_i + 2k_i\pi \neq \pm(\theta_j + 2k_j\pi)$ whatever $(k_i, k_j) \in \mathbb{Z}^2$.

Appendix B. Proof of Theorem 4.6

We want to show that the minimizer x^* over $\mathbb{Z}^r$ of $f(x) = x^Tx - q^Tx$ is obtained by rounding each



$\lfloor x_0 \rfloor + \begin{pmatrix}0\\1\end{pmatrix}$
$\lfloor x_0 \rfloor + \begin{pmatrix}1\\1\end{pmatrix}$
x_0
$\lfloor x_0 \rfloor$
$\lfloor x_0 \rfloor + \begin{pmatrix}1\\0\end{pmatrix}$

FIG. 4. Example of optimization over $\mathbb{Z}^2$. x^0 is the minimum of the function depicted by the grey circles and we have $x^* = \lfloor x_0 \rfloor + [1\ 0]^T$.

component of the minimizer x_0 of f over $\mathbb{R}^r$.
Whatever $q \in \mathbb{R}^r$, f is a strictly convex function whose minimizer over $\mathbb{R}^r$ satisfies $2x_0 = q$.

Let $\lfloor x_0 \rfloor$ be the componentwise floor of x_0. We note $d = x_0 - \lfloor x_0 \rfloor$ and make the variable change $x = \lfloor x_0 \rfloor + z = x_0 - d + z$. Note that d is in $[0,1)^r$. We then have

$$f(x) = (x_0 - d + z)^2 - q^t (x_0 - d + z) \tag{A.2}$$

$$= x_0^2 + \underbrace{(2x_0 - q)^T}_{0} (z - d) + (z - d)^2 - q^T x_0 \tag{A.3}$$

$$= (z - d)^2 + x_0^T (x_0 - q) \tag{A.4}$$

so that minimizing f over $\mathbb{Z}^r$ is equivalent to

$$\min_{z \in \mathbb{Z}^r} \|z - d\|^2 \tag{A.5}$$

Since $\|z - d\|^2 = \sum (z_i - d_i)^2$, which is a sum of positive quantities, it is easy to see that the minimum is reached when each z_i is the nearest integer to d_i, i.e. 0 if $d_i < 0.5$, 1 if $d_i > 0.5$ and 0 or 1 if $d_i = 0.5$. Whatever the tie-breaking rule, the minimizer z^* is obtained by rounding d componentwise and thus $x^* = \lfloor x_0 \rfloor + z^*$ is obtained by rounding x_0 componentwise.

Since $z \in \{0,1\}^r$, a geometric interpretation of this proof is that x_0 belongs to the hypercube $\lfloor x_0 \rfloor + [0,1]^r$ and that x^* is the vertex of this hypercube that is the closest to x_0 (see Fig 4).