Motion Interpolation in $\mathbf{SIM}(3)$

Christine Allen–Blanchette, Spyridon Leonardos, and Jean Gallier

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
e-mail: jean@cis.upenn.edu

December 15, 2014

Abstract. In this paper, we explore motion interpolation in the group $\mathbf{SIM}(3)$, the group of affine transformations which are the composition of a rotation, a translation, and a uniform rescaling. The group $\mathbf{SIM}(n)$ is a Lie group with Lie algebra $\mathfrak{sim}(n)$, and we give a formula for the exponential map $\exp\colon \mathfrak{sim}(n) \to \mathbf{SIM}(n)$. We prove the surjectivity of the exponential map for any $n \geq 1$, and for $n = 3$, we give an explicit formula and show how to compute logarithms. We use these algorithms for computing logarithms and exponentials to perform motion interpolation in $\mathbf{SIM}(3)$. Given a sequence $A_0, A_1, \ldots, A_n$ of transformations in $\mathbf{SIM}(3)$, we compute a sequence of logs $X_0, X_1, \ldots, X_n$ in $\mathfrak{sim}(3) \approx \mathbb{R}^7$, then fit a cubic spline $c(t)$ that interpolates the X_i, and then compute the curve $e^{c(t)}$ in $\mathbf{SIM}(3)$. However, the fact that the logarithm is multivalued causes problems. Whenever a rotation "crosses through π," the principal logarithm (the one associated with an angle in $[0, \pi]$) is not the correct choice and the motion goes "the long way." To correct this problem, we choose the next log so that the length of the interpolating arc from A_i to A_{i+1} is minimized. Unfortunately, we now obtain sequences of cubic splines with discontinuous junctions. To repair this problem, we introduce a class of sequences of cubic splines with discontinuous junctions but with continuity of the first and second derivatives at junction points. Since the exponential map removes the discontinuities (because for two distinct logs X and Y of $A \in \mathbf{SIM}(3)$, we have $e^X = e^Y$), we obtain C^2-continuous motions in $\mathbf{SIM}(3)$. We give several examples illustrating our implementation of the above methods.

1 Introduction

In this paper, we investigate methods for interpolating various deformations of a body (rigid or not). The paradigm that we use to define a deformation is the one used in elasticity theory. According to this method, the motion and deformation of a body (rigid or not) can be described by a *curve* in a *group G of transformations* of a space E (say $\mathbb{R}^n$, $n = 2, 3$). Given an *initial shape* $\mathcal{B} \in E$, a *deformation* of $\mathcal{B}$ is a (smooth enough) curve

$$\mathcal{D}\colon [0, T] \to G.$$

The element $\mathcal{D}(t)$ of the group G specifies how $\mathcal{B}$ is moved and deformed at time t, and the (moved and) deformed body $\mathcal{B}_t$ at time t is given by

$$\mathcal{B}_t = \mathcal{D}(t)(\mathcal{B}).$$

If $G = \mathbf{SO}(3)$, then we are modeling *rotations of a rigid body* (in $\mathbb{R}^3$). If $G = \mathbf{SE}(3)$, then we are modeling the *motion of a rigid body* (in $\mathbb{R}^3$). This means that the rigid body $\mathcal{B}$ rotates and translates in space. In this paper, we consider the slightly more general group $G = \mathbf{SIM}(3)$, which means that we are modeling a *simple deformation* of a (nonrigid) body (in $\mathbb{R}^3$). In addition to rotating and translating, the body $\mathcal{B}$ can grow and shrink in a uniform fashion (by a homothety).

The group $\mathbf{SIM}(n)$ consists of all affine maps ρ of $\mathbb{R}^n$ defined such that

$$\rho(x) = \alpha R x + u$$

for all $x \in \mathbb{R}^n$, where $R \in \mathbf{SO}(n)$ is a rotation matrix (an orthogonal matrix of determinant $+1$), u is some vector in $\mathbb{R}^n$ (the translation part of ρ), and $\alpha \in \mathbb{R}$ with $\alpha > 0$ (the scale factor). Such a map can be represented by the $(n+1) \times (n+1)$ matrix

$$\begin{pmatrix} \alpha R & u \\ 0 & 1 \end{pmatrix}$$

in the sense that

$$\begin{pmatrix} \rho(x) \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha R & u \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix}$$

iff

$$\rho(x) = \alpha R x + u.$$

The group $\mathbf{SIM}(n)$ is called the group of *direct affine similitudes* of $\mathbb{R}^n$.

The main reason for considering the Lie group $\mathbf{SIM}(3)$ is that, as far as we know, it is the largest group of affine transformations for which the exponential map $\exp\colon \mathfrak{sim}(3) \to \mathbf{SIM}(3)$ is surjective and easily computable. In fact, we give an explicit formula for the exponential. Furthermore, we also show that the (multivalued) logarithm function $\log\colon \mathbf{SIM}(3 \to \mathfrak{sim}(3)$ is easily computable.

The *interpolation problem* is the following: given a sequence $g_0, \ldots, g_m$ of deformations $g_i \in \mathbf{SIM}(3)$, with $g_0 = \mathrm{id}$, find a (reasonably smooth) curve $c\colon [0, m] \to \mathbf{SIM}(3)$ such that

$$c(i) = g_i, \quad i = 0, \ldots, m.$$

Unfortunately, the naive solution which consists in performing an interpolation

$$(1-t)g_i + tg_{i+1} \quad (0 \le t \le 1)$$

between g_i and g_{i+1} does not work, because $(1-t)g_i + tg_{i+1}$ *does not belong* to $\mathbf{SIM}(3)$ (in general).

A way around this difficulty is to interpolate in the linear space $\mathfrak{sim}(3)$ (the Lie algebra of $\mathbf{SIM}(3)$) and then use the exponential map to go back to $\mathbf{SIM}(3)$. The Lie algebra $\mathfrak{sim}(n)$ of $\mathbf{SIM}(n)$ consists of all $(n+1) \times (n+1)$ matrices of the form

$$\begin{pmatrix} \lambda I_n + \Omega & u \\ 0 & 0 \end{pmatrix} \quad \Omega \in \mathfrak{so}(n),\ u \in \mathbb{R}^n,\ \lambda \in \mathbb{R},$$

where $\mathfrak{so}(n)$ consists of the vector space of all $n \times n$ skew symmetric matrices. Fortunately, the exponential map $\exp\colon \mathfrak{sim}(3) \to \mathbf{SIM}(3)$ is *surjective* (this holds not just for $n = 3$ but also for all $n \ge 1$; see Theorem 3.2). This means that we have a *logarithm function* (actually, a multivalued function) $\log\colon \mathbf{SIM}(3) \to \mathfrak{sim}(3)$, such that

$$e^{\log A} = A, \quad A \in \mathbf{SIM}(3).$$

We can use the maps $\log\colon \mathbf{SIM}(3) \to \mathfrak{sim}(3)$ and $\exp\colon \mathfrak{sim}(3) \to \mathbf{SIM}(3)$ to interpolate in $\mathbf{SIM}(3)$ as follows: Given the sequence of "snapshots"

$$g_0, g_1, \ldots, g_m, \quad \text{in } \mathbf{SIM}(3)$$

1. Compute logs

$$X_0 = \log g_0,\ X_1 = \log g_1, \ldots,\ X_m = \log g_m, \quad \text{in } \mathfrak{sim}(3);$$

2. Find an interpolating curve $X\colon [0, m] \to \mathfrak{sim}(3)$, in $\mathfrak{sim}(3)$;

3. Exponentiate, to get the curve

$$c(t) = e^{X(t)}, \quad \text{in } \mathbf{SIM}(3).$$

Since $\mathfrak{sim}(3)$ is a *vector space* (with an inner product), interpolating in $\mathfrak{sim}(3)$ can be done easily using *spline curves*. Two problems remain:

1. Computing the logarithm of a matrix in $\mathbf{SIM}(3)$.

2. Computing the exponential of a matrix in $\mathfrak{sim}(3)$.

In this paper, we give a formula for the exponential map $\exp\colon \mathfrak{sim}(n) \to \mathbf{SIM}(n)$, and we prove its surjectivity for any $n \geq 1$. For $n = 3$, we give an explicit formula and show how to compute logarithms. We use these algorithms for computing logarithms and exponentials to perform motion interpolation in $\mathbf{SIM}(3)$. Given a sequence $A_0, A_1, \ldots, A_n$ of transformations in $\mathbf{SIM}(3)$, we compute a sequence of logs $X_0, X_1, \ldots, X_n$ in $\mathfrak{sim}(3) \approx \mathbb{R}^7$, then fit a cubic spline $c(t)$ that interpolates the X_i, and then compute the curve $e^{c(t)}$ in $\mathbf{SIM}(3)$. However, the fact that the logarithm is multivalued causes problems. Whenever a rotation "crosses through π," the principal logarithm (the one associated with an angle in $[0, \pi]$) is not the correct choice and the motion goes "the long way."

To the best of our knowledge, this problem has not been investigated by anybody else. To correct this problem, we choose the next log so that the length of the interpolating arc from A_i to A_{i+1} is minimized. Unfortunately, we now obtain sequences of cubic splines with discontinuous junctions. To repair this problem, we introduce a class of sequences of cubic splines with discontinuous junctions but with continuity of the first and second derivatives at junction points. Since the exponential map removes the discontinuities (because for two distinct logs X and Y of $A \in \mathbf{SIM}(3)$, we have $e^X = e^Y$), we obtain C^2-continuous motions in $\mathbf{SIM}(3)$. We give several examples illustrating our implementation of the above methods.

2 A Formula for the Exponential in $\mathbf{SIM}(3)$

We begin with some preliminary propositions.

Proposition 2.1. *Given any $(n+1) \times (n+1)$ matrix B of the form*

$$B = \begin{pmatrix} \Gamma & u \\ 0 & 0 \end{pmatrix},$$

where Γ is any real $n \times n$ matrix and $u \in \mathbb{R}^n$, we have

$$e^B = \begin{pmatrix} e^\Gamma & Vu \\ 0 & 1 \end{pmatrix},$$

with

$$V = I_n + \sum_{k \geq 1} \frac{\Gamma^k}{(k+1)!}.$$

Furthermore,

$$V = \int_0^1 e^{t\Gamma} dt.$$

Proof. Using induction on $n \geq 1$, it is easy to prove that

$$B^n = \begin{pmatrix} \Gamma^n & \Gamma^{n-1} W \\ 0 & 0 \end{pmatrix}.$$

This proves the first part of the proposition.

Since the power series for $e^{\Gamma t}$ converges uniformly on $[0,1]$, we have

$$\begin{aligned}\int_0^1 e^{t\Gamma}dt &= \int_0^1 \left(I + \sum_{k=1}^{\infty} \frac{(t\Gamma)^k}{k!}\right) dt \\ &= \left[tI + \sum_{k=1}^{\infty} t^{k+1}\frac{\Gamma^k}{(k+1)!}\right]_{t=0}^{t=1} \\ &= I + \sum_{k=1}^{\infty} \frac{\Gamma^k}{(k+1)!} = V.\end{aligned}$$

This proves the second part of the proposition. □

We now specialize Proposition 2.1 to the case where Γ is of the form $\Gamma = \lambda I + \Omega$.

Proposition 2.2. *Given any $(n+1)\times(n+1)$ matrix B of the form*

$$B = \begin{pmatrix} \lambda I + \Omega & u \\ 0 & 0 \end{pmatrix},$$

where Ω is a $n \times n$ matrix, $\lambda \in \mathbb{R}$, and $u \in \mathbb{R}^n$, we have

$$e^B = \begin{pmatrix} e^{\lambda}e^{\Omega} & Vu \\ 0 & 1 \end{pmatrix},$$

with

$$V = I_n + \sum_{k\geq 1} \frac{(\lambda I + \Omega)^k}{(k+1)!} = \int_0^1 e^{\lambda t I} e^{t\Omega} dt.$$

Proof. The diagonal matrix λI commutes with any matrix, so

$$(\lambda I)\Omega = \Omega(\lambda I),$$

and by a well-known property of the matrix exponential, we deduce that

$$e^{\lambda I + \Omega} = e^{\lambda I}e^{\Omega} = (e^{\lambda} I)e^{\Omega} = e^{\lambda}e^{\Omega}.$$

Similarly,

$$e^{t(\lambda I + \Omega)} = e^{\lambda t I}e^{t\Omega}.$$

Then, Proposition 2.2 follows immediately from Proposition 2.1 with $\Gamma = \lambda I + \Omega$. □

If Ω is a skew symmetric matrix, then e^{Ω} is a rotation matrix, so we have shown that for any matrix $B \in \mathfrak{sim}(n)$ given by

$$B = \begin{pmatrix} \lambda I + \Omega & u \\ 0 & 0 \end{pmatrix},$$

we have $e^B \in \mathbf{SIM}(n)$, with

$$e^B = \begin{pmatrix} e^{\lambda} e^{\Omega} & Vu \\ 0 & 1 \end{pmatrix},$$

and

$$V = I_n + \sum_{k \geq 1} \frac{(\lambda I + \Omega)^k}{(k+1)!} = \int_0^1 e^{\lambda t I} e^{t\Omega} dt.$$

Since the exponential map $\exp\colon \mathfrak{so}(n) \to \mathbf{SO}(n)$ is surjective, the surjectivity of the exponential map $\exp\colon \mathfrak{sim}(n) \to \mathbf{SIM}(n)$ depends on the invertibility of V. We prove that this exponential map is indeed surjective in Section 3.

Returning to the formula for the exponential in $\mathbf{SIM}(n)$, note that if we have an explicit formula for e^{Ω}, then we may have a chance to compute the integral

$$\int_0^1 e^{\lambda t I} e^{t\Omega} dt,$$

and obtain an explicit formula for e^B. For $n = 3$, thanks to the Rodrigues formula, we can carry out this plan.

For any matrix $\Omega \in \mathfrak{so}(3)$ of the form

$$\Omega = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix},$$

if we let $\theta = \sqrt{a^2 + b^2 + c^2}$, then Rodrigues formula states that

$$\begin{aligned} e^{\Omega} &= I_3 + \frac{\sin\theta}{\theta}\Omega + \frac{(1 - \cos\theta)}{\theta^2}\Omega^2, \quad \text{if } \theta \neq 0 \\ e^0 &= I_3, \quad \text{if } \theta = 0. \end{aligned}$$

Then, we have the following theorem.

Theorem 2.3. *for any matrix $B \in \mathfrak{sim}(3)$ given by*

$$B = \begin{pmatrix} \lambda I + \Omega & u \\ 0 & 0 \end{pmatrix},$$

we have

$$e^B = \begin{pmatrix} e^{\lambda} e^{\Omega} & Vu \\ 0 & 1 \end{pmatrix},$$

and if

$$\Omega = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix},$$

and $\theta = \sqrt{a^2 + b^2 + c^2}$, *then*

$$e^{\Omega} = I_3 + \frac{\sin\theta}{\theta}\Omega + \frac{(1-\cos\theta)}{\theta^2}\Omega^2, \quad \text{if } \theta \neq 0$$
$$e^0 = I_3, \quad \text{if } \theta = 0,$$

and V *is determined as follows:*

1. *If* $\theta = 0$ *and* $\lambda = 0$, *then*
$$V = I_3.$$

2. *If* $\theta = 0$ *and* $\lambda \neq 0$, *then*
$$V = \frac{(e^{\lambda} - 1)}{\lambda} I_3.$$

3. *If* $\theta \neq 0$ *and* $\lambda = 0$, *then*
$$V = I_3 + \frac{(1-\cos\theta)}{\theta^2}\Omega + \frac{(\theta - \sin\theta)}{\theta^3}\Omega^2.$$

4. *If* $\theta \neq 0$ *and* $\lambda \neq 0$, *then*
$$V = \frac{(e^{\lambda} - 1)}{\lambda} I_3 + \frac{(\theta(1 - e^{\lambda}\cos\theta) + e^{\lambda}\lambda\sin\theta)}{\theta(\lambda^2 + \theta^2)}\Omega + \left(\frac{(e^{\lambda} - 1)}{\lambda\theta^2} - \frac{e^{\lambda}\sin\theta}{\theta(\lambda^2 + \theta^2)} - \frac{\lambda(e^{\lambda}\cos\theta - 1)}{\theta^2(\lambda^2 + \theta^2)}\right)\Omega^2.$$

With the convention that $\frac{e^{\lambda}-1}{\lambda} = 1$ if $\lambda = 0$, the third clause for V is subsumed by the fourth clause.

Proof. If $\lambda = \theta = 0$, then $\Gamma = \Omega = 0$, so $V = I$. If $\theta = 0$ and $\lambda \neq 0$, we need to compute

$$\int_0^1 e^{\lambda t I} e^{t\Omega} dt.$$

But in this case, $\Omega = 0$, so

$$V = \int_0^1 e^{\lambda t I_3} dt = \frac{1}{\lambda}\left[e^{\lambda t I_3}\right]_{t=0}^{t=1} = \frac{1}{\lambda}(e^{\lambda} - 1)I_3.$$

If $\theta \neq 0$ and $\lambda \neq 0$, using the Rodrigues formula, we have

$$\begin{aligned} V &= \int_0^1 e^{\lambda t}\left(I_3 + \frac{\sin\theta t}{\theta t}\Omega t + \frac{(1-\cos\theta t)}{\theta^2 t^2}\Omega^2 t^2\right)dt \\ &= \int_0^1 e^{\lambda t}dt\, I_3 + \int_0^1 \frac{e^{\lambda t}\sin\theta t}{\theta}dt\,\Omega + \int_0^1 \frac{e^{\lambda t}}{\theta^2}dt\,\Omega^2 - \int_0^1 \frac{e^{\lambda t}\cos\theta t}{\theta^2}dt\,\Omega^2. \end{aligned}$$

Thus, we need to compute the integrals

$$\int_0^1 e^{\lambda t}\cos\theta t\, dt \quad \text{and} \quad \int_0^1 e^{\lambda t}\sin\theta t\, dt.$$

Observe that

$$e^{\lambda t + i\theta t} = e^{\lambda t}\cos(\theta t) + ie^{\lambda t}\sin(\theta t),$$

so we simply have to compute the primitive of $e^{\lambda t + i\theta t}$ and take its real and imaginary parts. This is easy: we have

$$\int_0^1 e^{\lambda t + i\theta t}dt = \frac{e^{\lambda+i\theta}-1}{\lambda+i\theta},$$

and from this, we get

$$\begin{aligned} \int_0^1 e^{\lambda t}\sin\theta t\, dt &= \frac{1}{\lambda^2+\theta^2}(\theta(1-e^{\lambda}\cos\theta) + e^{\lambda}\lambda\sin\theta) \\ \int_0^1 e^{\lambda t}\cos\theta t\, dt &= \frac{1}{\lambda^2+\theta^2}(\lambda(e^{\lambda}\cos\theta - 1) + e^{\lambda}\theta\sin\theta). \end{aligned}$$

Then, we have

$$\begin{aligned} V &= \int_0^1 e^{\lambda t}\left(I_3 + \frac{\sin\theta t}{\theta t}\Omega t + \frac{(1-\cos\theta t)}{\theta^2 t^2}\Omega^2 t^2\right)dt \\ &= \int_0^1 e^{\lambda t}dt\, I_3 + \int_0^1 \frac{e^{\lambda t}\sin\theta t}{\theta}dt\,\Omega + \int_0^1 \frac{e^{\lambda t}}{\theta^2}dt\,\Omega^2 - \int_0^1 \frac{e^{\lambda t}\cos\theta t}{\theta^2}dt\,\Omega^2 \\ &= \frac{(e^{\lambda}-1)}{\lambda}I_3 + \frac{(\theta(1-e^{\lambda}\cos\theta) + e^{\lambda}\lambda\sin\theta)}{\theta(\lambda^2+\theta^2)}\Omega \\ &\quad + \left(\frac{(e^{\lambda}-1)}{\lambda\theta^2} - \frac{e^{\lambda}\sin\theta}{\theta(\lambda^2+\theta^2)} - \frac{\lambda(e^{\lambda}\cos\theta - 1)}{\theta^2(\lambda^2+\theta^2)}\right)\Omega^2, \end{aligned}$$

as claimed. □

The next step is to figure out when the matrix V is invertible.

3 Surjectivity of the Exponential for SIM(n)

To determine when V is invertible, it is sufficient to find the eigenvalues of V. Using the Schur decomposition, the matrix Γ can be written as

$$\Gamma = UTU^*,$$

for some unitary matrix U and some upper triangular matrix T. By induction, it follows that $\Gamma^k = UT^kU^*$, and thus

$$V = I + \sum_{k=1}^{\infty} \frac{\Gamma^k}{(k+1)!} = U\Big(I + \sum_{k=1}^{\infty} \frac{T^k}{(k+1)!}\Big)U^*.$$

The diagonal entries of T are the eigenvalues of Γ, and it is easy to see that the diagonal entries of the matrix between U and U^* are of the form

$$1 + \sum_{k=1}^{\infty} \frac{z^k}{(k+1)!},$$

where z is an eigenvalue of Γ. If $z = 0$, then 1 is an eigenvalue of V. Otherwise, since

$$\frac{e^z - 1}{z} = 1 + \sum_{k=1}^{\infty} \frac{z^k}{(k+1)!},$$

we see that the eigenvalues of V are of the form $(e^z - 1)/z$, with $z \neq 0$ an eigenvalue of Γ. An eigenvalue of the form $(e^z - 1)/z$ (with $z \neq 0$) is zero iff

$$e^z = 1,$$

which holds iff $z = ik2\pi$, with $k \in \mathbb{Z} - \{0\}$ (since we are assuming $z \neq 0$).

In our situation,

$$\Gamma = \lambda I + \Omega,$$

with Ω an $n \times n$ skew symmetric matrix. It is well known that the eigenvalues of a (real) skew symmetric matrix are either 0 or $\pm i\theta_j$ with $\theta_j \neq 0$ for $j = 1, \ldots, m$ $(2m \leq n)$. Therefore, the eigenvalues of Γ are λ and $\lambda \pm i\theta_j$, for $j = 1, \ldots, m$. Consequently, V has 0 as an eigenvalue iff $\lambda = 0$ and $\theta_j = k2\pi$ for some j, with $k \in \mathbb{Z} - \{0\}$. In summary, we obtained the following proposition.

Proposition 3.1. *Given any $n \times n$ skew symmetric matrix Ω and any number $\lambda \in \mathbb{R}$, if $\pm i\theta_j$ with $\theta_j \neq 0$ for $j = 1, \ldots, m$ $(2m \leq n)$ are the nonzero eigenvalues of Ω, then $V = \int_0^1 e^{\lambda tI} e^{t\Omega} dt$ is invertible if either $\lambda \neq 0$, or for $j = 1, \ldots, m$, we have $\theta_j \neq k2\pi$ for all $k \in \mathbb{Z} - \{0\}$.*

Note that if $\Omega = 0$ and $\lambda = 0$, then V is invertible, since in this case $V = I$. We can now prove that the exponential $\exp\colon \mathfrak{sim}(n) \to \mathbf{SIM}(n)$ is surjective.

Theorem 3.2. *The exponential map* $\exp\colon \mathfrak{sim}(n) \to \mathbf{SIM}(n)$ *is surjective for all* $n \geq 1$.

Proof. Given any matrix

$$A = \begin{pmatrix} \alpha R & w \\ 0 & 1 \end{pmatrix},$$

we must show that there is some matrix

$$B = \begin{pmatrix} \Gamma & u \\ 0 & 0 \end{pmatrix},$$

with $\Gamma = \lambda I + \Omega$ and Ω skew-symmetric, so that

$$A = e^B = \begin{pmatrix} e^\Gamma & Vu \\ 0 & 1 \end{pmatrix}.$$

From Proposition 2.3, we have

$$e^\Gamma = e^\lambda e^\Omega.$$

Therefore, we must find Ω, λ, and u such that

$$e^\lambda e^\Omega = \alpha R$$
$$Vu = w.$$

Case 1. $R = I$.

Since $\alpha > 0$, we let $\lambda = \log \alpha$, and we pick $\Omega = 0$. In this case, $\theta = 0$, so $V = \frac{1}{\lambda}(e^\lambda - 1)I$ ($V = I$ when $\lambda = 0$) is invertible, and $u = V^{-1}w$.

Case 2. $R \neq I$.

Since $\alpha > 0$, we let $\lambda = \log \alpha$. Since $R \neq I$, it is known (for example, see Gallier [4] Theorem 18.1) that the exponential map $\exp\colon \mathfrak{so}(n) \to \mathbf{SO}(n)$ is surjective, and by the reasoning in the proof of Proposition 18.3 in Gallier [4], we may assume that the nonzero eigenvalues $i\theta_j$ of the skew symmetric matrix Ω such that $e^\Omega = R$ are such that $\theta_j \neq k2\pi$ for all $k \in \mathbb{Z}$. In fact, we may assume that $0 < \theta_j < \pi$. By Proposition 3.1, the matrix V is invertible, so $u = V^{-1}w$.

Since all cases have been covered, we proved that the map $\exp\colon \mathfrak{sim}(n) \to \mathbf{SIM}(n)$ is surjective. □

For $n = 3$, since we have an explicit formula for V and for e^Ω, we can compute logarithms explicitly, as we now explain.

4 Computing Logarithms in SIM(3)

From Section 2, given any matrix

$$A = \begin{pmatrix} \alpha R & w \\ 0 & 1 \end{pmatrix},$$

in **SIM**(3), in order to compute logarithms of A, we need to compute logarithms of rotation matrices $R \in \mathbf{SO}(3)$ and to invert the matrix V, which is given by the formulae of Theorem 2.3. Computing logarithms of 3×3 rotation matrices is well-known (for example, see Gallier [4]). For the sake of completeness, we review the procedure.

(1) The case $R = I$ is trivial.

(2) If $R \neq I$ and $\mathrm{tr}(R) \neq -1$, then

$$\exp^{-1}(R) = \left\{ \frac{\theta}{2\sin\theta}(R - R^\top) \;\middle|\; 1 + 2\cos\theta = \mathrm{tr}(R) \right\}.$$

In particular, there is a unique skew-symmetric

$$B = \frac{\theta}{2\sin\theta}(R - R^\top)$$

with corresponding θ satisfying $0 < \theta < \pi$ such that $e^B = R$, called the *principal logarithm* of R (and with θ called the *principal angle*). All other determinations of the logarithm of R are of the form

$$(\theta + k2\pi)\frac{B}{\theta}, \quad k \in \mathbb{Z}.$$

(3) If $R \neq I$ and $\mathrm{tr}(R) = -1$, then R is symmetric and there exists a skew symmetric matrix

$$U = \begin{pmatrix} 0 & -d & c \\ d & 0 & -b \\ -c & b & 0 \end{pmatrix}$$

so that

$$U^2 = S = \frac{1}{2}(R - I).$$

Furthermore, $b^2 + c^2 + d^2 = 1$, and

$$\exp^{-1}(R) = \left\{ (2k+1)\pi \begin{pmatrix} 0 & -d & c \\ d & 0 & -b \\ -c & b & 0 \end{pmatrix},\ k \in \mathbb{Z} \right\}.$$

To find a skew symmetric matrix U so that $U^2 = S = \frac{1}{2}(R - I)$, we can solve the system

$$\begin{pmatrix} b^2 - 1 & bc & bd \\ bc & c^2 - 1 & cd \\ bd & cd & d^2 - 1 \end{pmatrix} = S.$$

We immediately get b^2, c^2, d^2, and then, since one of b, c, d is nonzero, say b, if we choose the positive square root of b^2, we can determine c and d from bc and bd.

This procedure is easily implemented in `Matlab`, but tolerance factors are needed to handle the cases where R is very close to the identity or when the angle θ is small. Because we typically compute the principal logarithm, $0 < \theta < \pi$, the matrix V is invertible and we never encountered numerical problems computing the inverse of V. However, we will see in the next section that problems arise when computing interpolating splines, because in this case, we may need to consider angles whose value is close to a multiple of 2π.

5 Finding Interpolating Splines in $\mathfrak{sim}(3)$

Having an algorithm to compute logarithms of transformations in $\mathbf{SIM}(3)$ and a formula for computing the exponential map $\exp\colon \mathfrak{sim}(3) \to \mathbf{SIM}(3)$, in principle, we can find interpolating splines in $\mathbf{SIM}(3)$ by computing logarithms and finding interpolating splines in $\mathfrak{sim}(3)$. However, there are unexpected problems due to the fact that the logarithm is multivalued.

Given two transformations $A_1, A_2 \in \mathbf{SIM}(3)$, since the logs X_1 and X_2 in $\mathfrak{sim}(3)$ such that $A_1 = e^{X_1}$ and $A_2 = e^{X_2}$ are not uniquely determined, there are several interpolating curves between A_1 and A_2 arising from the various affine interpolants $t \mapsto (1-t)X_1 + tX_2$ in $\mathfrak{sim}(3)$, so we need to decide which curve is the *intended motion*. It seems reasonable to assume that the intended motion is the one that *minimizes* the length of the curve segment $t \mapsto e^{(1-t)X_1 + tX_2}$ in $\mathbf{SIM}(3)$. To make this precise, we need to introduce a Riemannian metric. We will return to this point shortly, but right now let us proceed more intuitively.

The main problem has to do with "crossing π." For example, consider

$$R_1 = \begin{pmatrix} \cos(2\pi/3) & -\sin(2\pi/3) & 0 \\ \sin(2\pi/3) & \cos(2\pi/3) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

corresponding to a rotation by $2\pi/3$ around the z-axis, and

$$R_2 = \begin{pmatrix} \cos(4\pi/3) & -\sin(4\pi/3) & 0 \\ \sin(4\pi/3) & \cos(4\pi/3) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

corresponding to a rotation by $4\pi/3$, also around the z-axis. The principal logarithms of R_1 and R_2 are

$$B_1 = \begin{pmatrix} 0 & -2\pi/3 & 0 \\ 2\pi/3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and

$$B_2 = \begin{pmatrix} 0 & 2\pi/3 & 0 \\ -2\pi/3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Observe that $B_2 = -B_1$. Now, the problem is that if we interpolate in $\mathfrak{so}(3)$ using affine interpolation,

$$B = (1-t)B_1 + tB_2 = (1-t)B_1 - tB_1 = (1-2t)B_1,$$

we see that the corresponding rotation

$$R = e^B = e^{(1-2t)B_1}$$

goes through the identity rotation for $t = 1/2$, and the motion goes the long way from R_1 to R_2 through I, instead of going through the rotation of angle π around the z-axis, which is the intented motion. If we use the other determination of the logarithm of R_2 corresponding to $2\pi - 2\pi/3 = 4\pi/3$, namely

$$B_2' = \begin{pmatrix} 0 & -4\pi/3 & 0 \\ 4\pi/3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

this time we have the interpolant

$$(1-t)B_1 + tB_2' = \begin{pmatrix} 0 & -(1+t)2\pi/3 & 0 \\ (1+t)2\pi/3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and for $t = 1/2$, we get the rotation of angle π around the z-axis, as desired.

The above example shows that if we want to have an interpolation which corresponds to the intended motion, then we can't systematically use the principal logarithm. This leads to another problem. Suppose the next rotation to be interpolated is

$$R_3 = \begin{pmatrix} \cos(5\pi/3) & -\sin(5\pi/3) & 0 \\ \sin(5\pi/3) & \cos(5\pi/3) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

corresponding to a rotation by $5\pi/3$, also around the z-axis, and whose principal logarithm is

$$B_3 = \begin{pmatrix} 0 & \pi/3 & 0 \\ -\pi/3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

If we interpolate between B_2' and B_3, we obtain

$$B = (1-t)B_2' + tB_3 = \begin{pmatrix} 0 & -(4\pi/3 - t5\pi/3) & 0 \\ 4\pi/3 - t5\pi/3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and for $t = 1/5$, the interpolated rotation $R = e^B$ goes through the rotation of angle π around the z-axis, and through the identity for $t = 4/5$. Thus, the interpolation goes the long way, which is not the intended motion. However, if we interpolate between B_2 and B_3, then we have

$$B = (1-t)B_2 + tB_3 = \begin{pmatrix} 0 & \pi/3(2-t) & 0 \\ -\pi/3(2-t) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and this is the correct (intended) interpolant.

If instead of B_3, we use the determination B_3' of the logarithm corresponding to the angle $2\pi - 5\pi/3 = \pi/3$ given by

$$B_3' = \begin{pmatrix} 0 & -\pi/3 & 0 \\ \pi/3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

then the interpolant between B_2' and B_3' is given by

$$B = (1-t)B_2' + tB_3' = \begin{pmatrix} 0 & -(4\pi/3 - t\pi) & 0 \\ 4\pi/3 - t\pi & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and this is also the intended interpolant.

Let us now make precise what we mean when we say that the resulting interpolated motion is the intended one; that is, does not "go the long way." The problem really has to do with computing the logarithm of the rotation part R_i of the transformation A_i. In order to define the length of a curve in $\mathbf{SO}(3)$, we equip $\mathbf{SO}(3)$ with a Riemannian metric. Because $\mathbf{SO}(3)$ is a Lie group, it suffices to define the inner product

$$\langle B_1, B_2 \rangle = \frac{1}{2}\mathrm{tr}(B_1^\top B_2) = -\frac{1}{2}\mathrm{tr}(B_1 B_2)$$

on $\mathfrak{so}(3)$, and since this inner product is Ad-invariant, it induces a bi-invariant Riemannian metric on $\mathbf{SO}(3)$ (see Gallot, Hullin, Lafontaine [5] or O'Neill [13]). If we write a skew symmetric matrix $B \in \mathfrak{so}(3)$ as

$$B = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix},$$

then we check immediately that the inner product $\langle B_1, B_2 \rangle$ corresponds to the inner product of the vectors (a_1, b_1, c_1) and (a_2, b_2, c_2) associated with B_1 and B_2. Now, for any two rotations $R_1, R_2 \in \mathbf{SO}(3)$ and any two skew symmetric matrices $B_1, B_2 \in \mathfrak{so}(3)$ such that $R_1 = e^{B_1}$ and $R_2 = e^{B_2}$, consider the curve γ in $\mathbf{SO}(3)$ given by

$$\gamma(t) = e^{(1-t)B_1 + tB_2}, \quad t \in [0, 1].$$

We have $\gamma(0) = R_1$ and $\gamma(1) = R_2$, and it is shown in Section 10 that the length $L(\gamma)$ of the curve γ is

$$\begin{aligned} L(\gamma) &= \int_0^1 \langle \gamma'(t), \gamma'(t) \rangle^{\frac{1}{2}} dt \\ &= \left(-\frac{1}{2} \mathrm{tr}((B_2 - B_1)^2) \right)^{\frac{1}{2}} \\ &= \sqrt{\langle B_2 - B_1, B_2 - B_1 \rangle} = \|B_2 - B_1\|. \end{aligned}$$

Therefore, given B_1, in order to minimize the length $L(\gamma) = \|B_2 - B_1\|$ of the curve segment γ, it suffices to pick B_2 so that $\|B_2 - B_1\| = \left(-\frac{1}{2} \mathrm{tr}((B_2 - B_1)^2) \right)^{\frac{1}{2}}$ is minimized.

Then, there appears to be two strategies to compute logarithms of transformations A_i in SIM(3) so that the resulting interpolated motion is the intended one; that is, does not "go the long way."

The two strategies to compute logarithms of rotations (associated with transformations in $\mathbf{SIM}(3)$) are as follows:

(1) Compute the principal logs B_i and B'_{i+1} of R_i and R_{i+1} (with corresponding principal angles θ_i and θ_{i+1}), as well as the determination B''_{i+1} of the log of R_{i+1} corresponding to $2\pi - \theta_{i+1}$ if $\theta_{i+1} \neq 0$, namely

$$B''_{i+1} = -(2\pi - \theta_{i+1}) \frac{B'_{i+1}}{\theta_{i+1}}.$$

Then, pick for the log of R_{i+1} the matrix B_{i+1} among B'_{i+1} and B''_{i+1} which minimizes $-\frac{1}{2}\mathrm{tr}((B_{i+1} - B_i)^2)$. If $\theta_{i+1} = 0$, then $R_{i+1} = I$, and we set $B_{i+1} = 0$.

(2) For the second strategy, we keep track of the angle $\theta'_i = \theta_i + k_i 2\pi$ that was used in determining the logarithm B_i of R_i, where $\mathcal{B}_i$ is the principal logarithm of R_i and θ_i is the principal angle, so that

$$B_i = (\theta_i + k_i 2\pi) \frac{\mathcal{B}_i}{\theta_i}.$$

Then, we choose the angle $\theta'_{i+1} = \theta_{i+1} + k_{i+1} 2\pi$ in such a way that the integer k_{i+1} minimizes

$$-\frac{1}{2} \mathrm{tr} \left(B_i - (\theta_{i+1} + k_{i+1} 2\pi) \frac{\mathcal{B}_{i+1}}{\theta_{i+1}} \right)^2$$

if $\theta_{i+1} \neq 0$. If $\theta_{i+1} = 0$, then $R_{i+1} = I$, but we want to avoid a lot of "winding," so we replace $\mathcal{B}_{i+1} = 0$ by $\mathcal{B}_i/\theta_i$, and find k_{i+1} as above (in this case, $\theta_{i+1} = 0$). This case can cause trouble regarding the invertibility of V.

Each transformation $A \in \mathbf{SIM}(3)$ is specified by a triple (R, α, u), where $R \in \mathbf{SO}(3)$ is a rotation, $\alpha > 0$ is a scale factor, and $u \in \mathbb{R}^3$ is the translation part. Similarly, each log

$X \in \mathfrak{sim}(3)$ is a triple (B, α, v), where $B \in \mathfrak{so}(3)$, $\alpha \in \mathbb{R}$, and $v \in \mathbb{R}^3$. We call B the rotation part of X. Following the first strategy, given a sequence of transformations

$$A_0, A_1, \ldots, A_m$$

in $\mathbf{SIM}(3)$, first we compute the sequence of principal logs

$$X_0, X_1, \ldots, X_m.$$

Actually, we compute the principal log of the rotation part R_i of A_i as well as $\lambda_i = \log(\alpha_i)$, and then we compute the sequence of pairs

$$(X_0, Y_1), (X_1, Y_2), (X_2, Y_3), \ldots, (X_{m-1}, Y_m),$$

where the rotation part of Y_{i+1} is computed from the rotation part of X_i according to strategy 1 for $i = 0, \ldots, m-1$. Next, we form the longest subsequence

$$(X_0, Y_1, \ldots, Y_{m_1})$$

such that $Y_i = X_i$ for $i = 1, \ldots, m_1$; then the longest subsequence

$$(X_{m_1}, Y_{m_1+1}, \ldots, Y_{m_1+m_2})$$

such that $Y_i = X_i$ for $i = m_1, \ldots, m_1 + m_2$; and so on. At the end of this process, we have K sequences of logs

$$(X_{m_1+\cdots+m_k}, Y_{m_1+\cdots+m_k+1}, \ldots, Y_{m_1+\cdots+m_{k+1}}),$$

which we use for interpolation.

The main drawback of stategy 1 is that we have discontinuities. For example, $Y_{m_1} \neq X_{m_1}$, $Y_{m_1+m_2} \neq X_{m_1+m_2}$, etc. However, we found a way to construct interpolating splines with discontinuities but such that first and second derivatives agree at discontinuity points (see Section 8 and its subsections). Since we apply the exponential map to all these logs and since

$$e^{Y_{m_1}} = e^{X_{m_1}}, e^{Y_{m_1+m_2}} = e^{X_{m_1+m_2}}, \ldots$$

in the end, we obtain C^2 continuous interpolating splines in $\mathbf{SIM}(3)$. A small technical detail should be noted. If a sequence (X_i, Y_{i+1}) consist of just two elements, then in order to be able to do spline interpolation, we insert the average of X_i and Y_{i+1} as a third data point.

Regarding strategy 2, because

$$-\frac{1}{2}\mathrm{tr}\left(B_i - (\theta_{i+1} + k_{i+1}2\pi)\frac{\mathcal{B}_{i+1}}{\theta_{i+1}}\right)^2$$

is a quadratic function in k_{i+1}, we can determine when it achieves a minimum. Let $f(k)$ be the function given by

$$f(k) = -\mathrm{tr}\left(B_i - (\theta_{i+1} + k2\pi)\frac{\mathcal{B}_{i+1}}{\theta_{i+1}}\right)^2.$$

To find where $f(k)$ is minimum, let's compute the derivative of $f(k)$. Since

$$f(k) = -\mathrm{tr}(B_i^2) + 2\mathrm{tr}(B_i\mathcal{B}_{i+1})\frac{(\theta_{i+1} + k2\pi)}{\theta_{i+1}} - (\theta_{i+1} + k2\pi)^2\mathrm{tr}\left(\frac{\mathcal{B}_{i+1}^2}{\theta_{i+1}^2}\right),$$

and since

$$-\mathrm{tr}\left(\frac{\mathcal{B}_{i+1}^2}{\theta_{i+1}^2}\right) = 2,$$

we get

$$f'(k) = \mathrm{tr}(B_i\mathcal{B}_{i+1})\frac{4\pi}{\theta_{i+1}} + \theta_{i+1}8\pi + k16\pi^2.$$

It is adantageous to make the inner product of the unit vectors associated with $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ appear, so if we write

$$c = -\frac{1}{2}\mathrm{tr}\left(\frac{B_i}{\theta_i'}\frac{\mathcal{B}_{i+1}}{\theta_{i+1}}\right) = -\frac{1}{2}\mathrm{tr}\left(\frac{\mathcal{B}_i}{\theta_i}\frac{\mathcal{B}_{i+1}}{\theta_{i+1}}\right),$$

then,

$$f'(k) = -8\pi c\theta_i' + \theta_{i+1}8\pi + k16\pi^2,$$

and $f'(k) = 0$ for

$$k = (c\theta_i' - \theta_{i+1})/2\pi.$$

Since $\theta_i' = \theta_i + k_i 2\pi$, we get

$$k = ck_i + (c\theta_i - \theta_{i+1})/2\pi.$$

The above formula can be used to predict what k_{i+1} is and avoid searching for it. It also shows that if c is small, which means that the axes of the rotations corresponding to B_i and B_{i+1} are close to being perpendicular, then k_{i+1} can be a lot smaller than k_i, and this causes of lot of winding in the interpolation. Therefore, when using strategy 2, we have to make sure that the axes of the rotations associated with our transformations don't change too much from one transformation to the next. However there is a worse problem, which has to do with "crossing I." Indeed, if some θ_i' is a nonzero multiple of 2π, then V is singular if some $\lambda_i = 0$. Even if $\lambda_i \neq 0$, we observed that the inverse of V tends to have large coefficients, and as a consequence this yields interpolations that wind a lot.

Strategy 1 does not suffer from these problems because the angles are in the range $[0, \pi]$. We found that the splines discussed in Section 8 handle the situation very well.

6 Some Implementation Details

The programs implementing motion interpolation in **SIM**(3) perform essentially two tasks:

1. Compute logs and exponentials.

2. Handle the splines needed for interpolation.

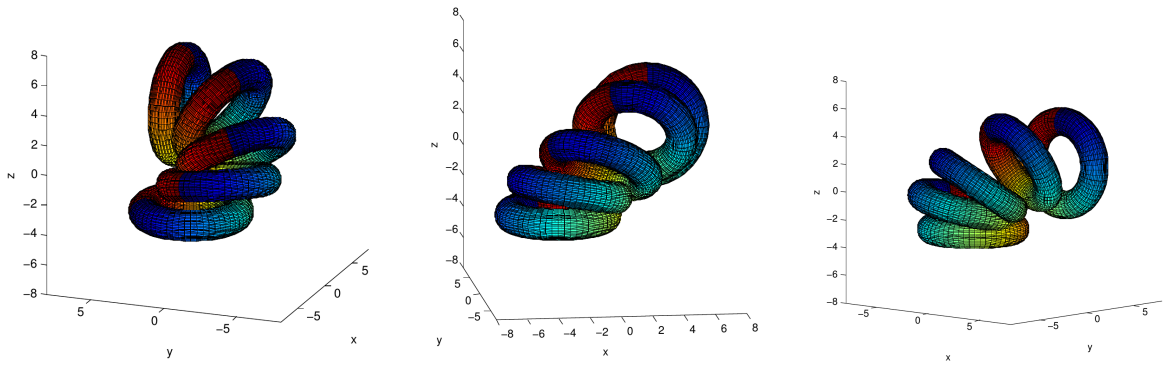
The first task is to compute the list of sequences of log's that constitutes the list of K sequences of data points to be interpolated in $\mathfrak{sim}(3) \approx \mathbb{R}^7$, according to strategy 1 (using the methods of Sections and 2 and 4). A little bit of care has to be exercised with rotation angles close to 0 or close to π (we use tolerance factors). However, because we only deal with angles in $[0, \pi]$, the matrix V is always invertible, without any problem. Next, we insert a third data point (the midpoint) if a sequence has only two elements. We also implemented strategy 2, but as we said before, it does not work well when crossing angles that are multiple of 2π. In this case, we observed that the matrix V is often singular.

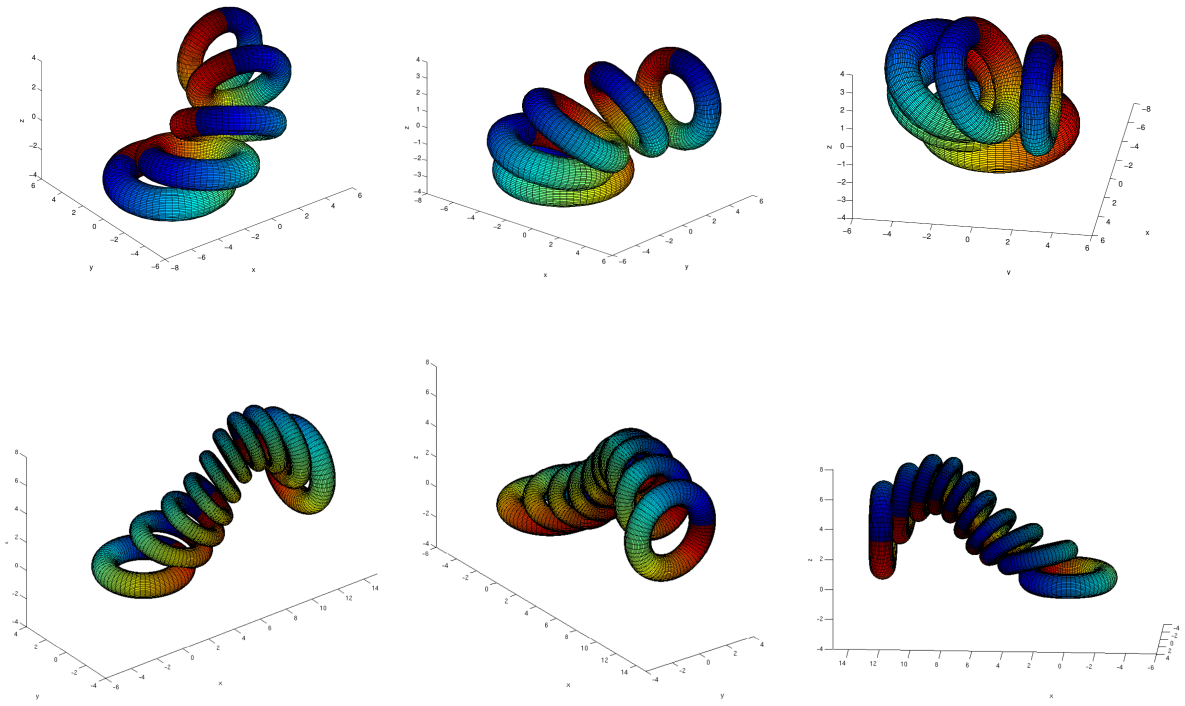
The second stage is to compute a list of sequence of de Boor control points, as explained in Section 8. For this, we solve a tridiagonal linear system using a fast elimination method. This is the most complicated program of the whole project. Next, we compute a list of sequences of Bézier control polygons from the de Boor control points.

The third stage is to compute the list of interpolated transformations. For this we use the exponential formula of Section 2. We subdivide a Bézier control polygon using either the Bernstein polynomials, or using the de Casteljau algorithm in $\mathbb{R}^7$. The second method can use the intermediate control points (which are close to the curve,) and appears to be faster.

The last step is to display the motion. In `Matlab`, we can either use the `hold on`, `hold off` method which produces a "lava flow" rendering of the motion, or create a movie.

7 Examples of Simulations of Deformations





The next example is a motion interpolation involving the four transformations

$$\begin{aligned} A_1 &= (u_1, 2\pi/3, 1, w_1) \\ A_2 &= (u_1, 4\pi/3, 3/2, w_2) \\ A_3 &= (u_2, 0, 1, w_3) \\ A_4 &= (u_3, -\pi/3, 3/2, w_4), \end{aligned}$$

where the rotation component is defined by a pair (u, θ), where u is nonzero vector specifying the axis of rotation and θ is the angle of rotation, with

$$\begin{array}{llll} u_1 = (1,0,0) & u_2 = (0,1,0) & u_3 = (0,0,1) & \\ w_1 = (-3,0,-3) & w_2 = (3,3,2) & w_3 = (-3,3,3) & w_4 = (4,0,-3). \end{array}$$

Observe that the rotation part of A_2 is the identity rotation. The successive angles of rotation are $2\pi/3, 4\pi/3, 0, -\pi/3$, so there is a passage through π which causes a discontinuity. Indeed, we obtain two splines, the first one from A_1 to A_2, and the second one through A_2, A_3, A_4. The transformations A_i are applied to an ellipsoid, and Figure 4 shows the bodies to be interpolated.

Figure 5 shows the trajectory of a point on the original ellipsoid, using the C^2-continuous spline computed from the correct logs.

Figure 6 shows the trajectory of a point on the original ellipsoid, using the two splines joined with C^0-continuity, computed from the correct logs. The second red point on the

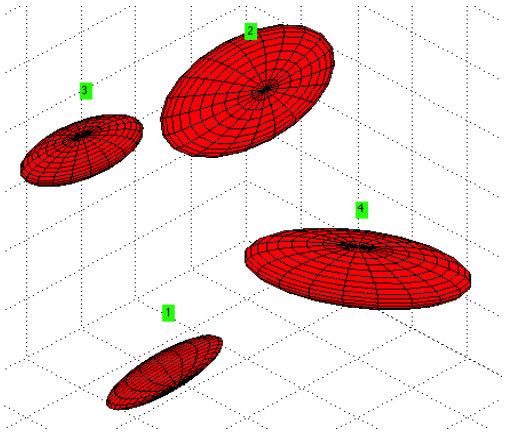


Figure 4: Four initial bodies.

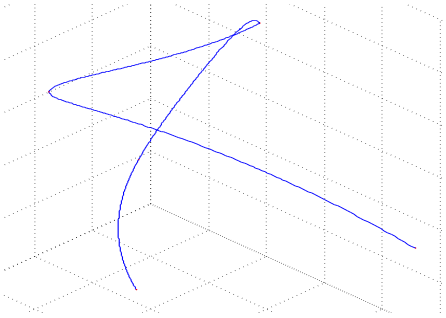


Figure 5: C^2-continous trajectory of a point during the motion.

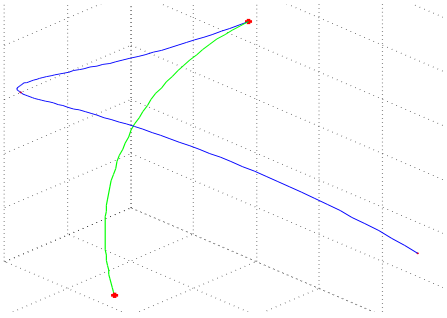


Figure 6: C^0-continuous trajectory of a point during the motion.

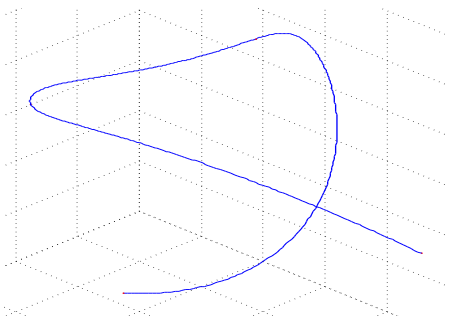


Figure 7: Wrong trajectory (obtained from wrong logs) of a point during the motion.

green arc (which corresponds to the spline interpolating A_1 and A_2) is a place where the curve is not even C^1.

Figure 7 shows the trajectory of a point on the original ellipsoid, using the single C^2-continuous spline computed from the (incorrect) logs of A_1, A_2, A_3, A_4. The first arc from A_1 to A_2 curves the wrong way and overturns because the log at A_2 is wrong.

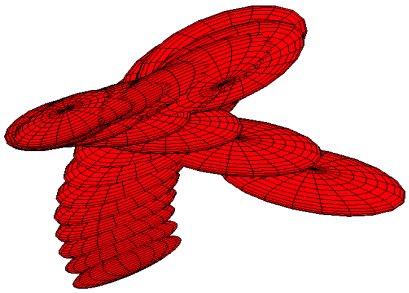


Figure 8: Interpolating motion (coarse).

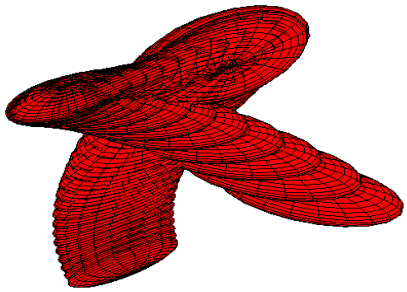


Figure 9: Interpolating motion (fine).

The motion interpolation is shown in Figure 8, using a coarse interpolation step, and in Figure Figure 9, using a finer interpolation step.

The last example shows a simulation of a screw motion. The input consists of 10 bodies obtained by successsive rotations by $2\pi/3$ and translation along the z-axis by $(0, 0, 2)$; see Figure 10. Except for the fourth and the fifth transformations where the scale factor is 1.2, the scale factor is 1.5. The trajectory of a point is shown in Figure 11.

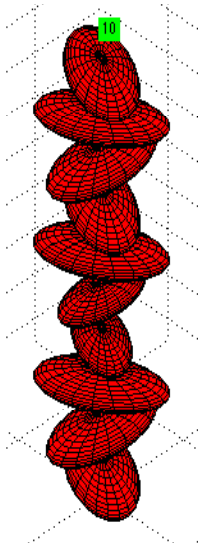


Figure 10: Initial position of the bodies (screw motion).

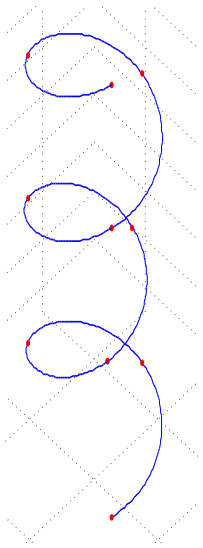


Figure 11: Trajectory of a point (screw motion).

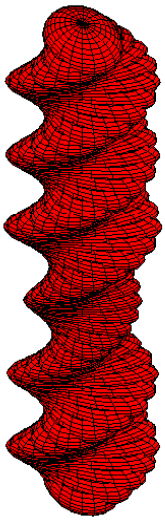


Figure 12: Interpolated screw motion.

The interpolated screw motion is shown in Figure 12. The trajectory of a point using the single C^2-continuous spline computed from the incorrect logs is shown in Figure 13.

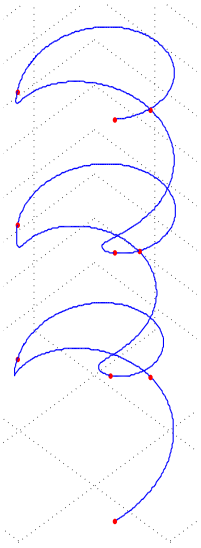


Figure 13: Wrong trajectory of a point (screw motion).

8 Splicing Interpolating Splines with First and Second Derivatives Agreement

8.1 K Splines, Each With at Least Four Data Points

Let us begin with the case $K = 2$. Let $(x_0, \ldots, x_M)$ and $(y_0, \ldots, y_N)$ be two sequences of $M + 1$ and $N + 1$ data points, and assume that $M, N \geq 3$. In order to find a cubic spline S_1 interpolating the data points x_i, and a cubic spline S_2 interpolating the data points y_j, we can solve for $M + 1$ (resp. $N + 1$) de Boor control points $d_0, \ldots, d_M$ (resp. $f_0, \ldots, f_N$), which are solutions of the system

$$\begin{pmatrix} \frac{7}{2} & 1 & & & \\ 1 & 4 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & 1 & 4 & 1 \\ & & & 1 & \frac{7}{2} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{M-2} \\ d_{M-1} \end{pmatrix} = \begin{pmatrix} 6x_1 - \frac{3}{2}d_0 \\ 6x_2 \\ \vdots \\ 6x_{M-2} \\ 6x_{M-1} - \frac{3}{2}d_M \end{pmatrix},$$

and similarly for $f_0, \ldots, f_N$.

The derivation of the above system assumes that $M \geq 4$. If $M = 3$, this system reduces to

$$\begin{pmatrix} \frac{7}{2} & 1 \\ 1 & \frac{7}{2} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 6x_1 - \frac{3}{2}d_0 \\ 6x_2 - \frac{3}{2}d_3 \end{pmatrix}.$$

Actually, d_0 and d_M are free parameters (as well as f_0 and f_N) and we are really solving for $d_1, \ldots, d_{M-1}$ in terms of d_0, d_M (resp. $f_1, \ldots, f_{N-1}$ in terms of f_0, f_N). The control points d_0 and d_M can be determined by prescribing end conditions (and similarly for f_0 and f_N).

In our situation, in general, $x_M \neq y_0$, yet we would like to find d_M and f_0 so that the first and second derivatives at x_M and y_0 agree. This can indeed be done.

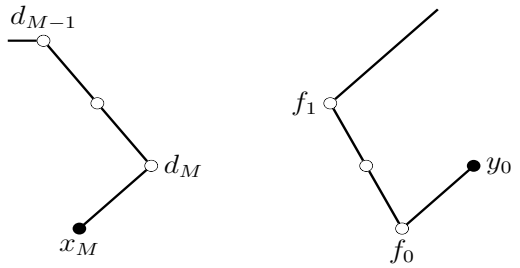


Figure 14: Splicing splines with first and second derivatives agreement.

Recall (refer to Figure 14) that the first derivatives at x_M and y_0 are

$$S_1'(x_M) = 3(x_M - d_M)$$
$$S_2'(y_0) = 3(f_0 - y_0),$$

and the second derivatives are given by

$$S_1'(x_M) = 6\left(\frac{1}{2}d_{M-1} + \frac{1}{2}d_M - 2d_M + x_M\right)$$
$$S_2''(y_0) = 6\left(y_0 - 2f_0 + \frac{1}{2}f_0 + \frac{1}{2}f_1\right);$$

that is,

$$S_1'(x_M) = 6\left(\frac{1}{2}d_{M-1} - \frac{3}{2}d_M + x_M\right)$$
$$S_2''(y_0) = 6\left(y_0 - \frac{3}{2}f_0 + \frac{1}{2}f_1\right).$$

Thus, the conditions

$$S_1'(x_M) = S_2'(y_0)$$
$$S_1''(x_M) = S_2''(y_0)$$

yield

$$x_M - d_M = f_0 - y_0$$

and

$$\frac{1}{2}d_{M-1} - \frac{3}{2}d_M + x_M = y_0 - \frac{3}{2}f_0 + \frac{1}{2}f_1.$$

The first equation can be written in a way that has geometric meaning, namely

$$\frac{1}{2}(d_M + f_0) = \frac{1}{2}(x_M + y_0),$$

which says that the midpoints of the line segments (d_M, f_0) and (x_M, y_0) coincide. We can also use this equation to express d_M in terms of f_0 to obtain an equation for f_0 in terms of f_1 and f_{M-1}. By substituting $d_M = x_M + y_0 - f_0$ in the equation

$$\frac{1}{2}d_{M-1} - \frac{3}{2}d_M + x_M = y_0 - \frac{3}{2}f_0 + \frac{1}{2}f_1,$$

we get

$$d_{M-1} - 3(x_M + y_0 - f_0) + 2x_M = 2y_0 - 3f_0 + f_1;$$

that is,

$$f_0 = \frac{1}{6}x_M + \frac{5}{6}y_0 - \frac{1}{6}d_{M-1} + \frac{1}{6}f_1. \qquad (*)$$

Using $d_M = x_M + y_0 - f_0$, we obtain

$$d_M = \frac{5}{6}x_M + \frac{1}{6}y_0 + \frac{1}{6}d_{M-1} - \frac{1}{6}f_1. \qquad (**)$$

Observe that equations $(*)$ and $(**)$ also hold if $M = 2$ or $N = 2$.

Now, if $M, N \geq 3$, we also have the equations

$$d_{M-2} + \frac{7}{2}d_{M-1} = 6x_{M-1} - \frac{3}{2}d_M$$
$$\frac{7}{2}f_1 + f_2 = 6y_1 - \frac{3}{2}f_0.$$

From the first equation and $(**)$, we get

$$d_{M-2} + \frac{7}{2}d_{M-1} = 6x_{M-1} - \frac{5}{4}x_M - \frac{1}{4}y_0 - \frac{1}{4}d_{M-1} + \frac{1}{4}f_1,$$

which yields

$$4d_{M-2} + 15d_{M-1} - f_1 = 24x_{M-1} - 5x_M - y_0.$$

From the second equation and $(*)$, we get

$$\frac{7}{2}f_1 + f_2 = 6y_1 - \frac{1}{4}x_M - \frac{5}{4}y_0 + \frac{1}{4}d_{M-1} - \frac{1}{4}f_1,$$

which yields

$$-d_{M-1} + 15f_1 + 4f_2 = -x_M - 5y_0 + 24y_1.$$

In summary, we obtain the equations

$$\begin{array}{rrrcl} 4d_{M-2} & +15d_{M-2} & -f_1 & = & 24x_{M-1} - 5x_M - y_0 \\ & -d_{M-1} & +15f_1 \quad +4f_2 & = & -x_M - 5y_0 + 24y_1, \end{array}$$

and d_M, f_0 are given by

$$d_M = \frac{5}{6}x_M + \frac{1}{6}y_0 + \frac{1}{6}d_{M-1} - \frac{1}{6}f_1$$
$$f_0 = \frac{1}{6}x_M + \frac{5}{6}y_0 - \frac{1}{6}d_{M-1} + \frac{1}{6}f_1.$$

If we use the natural end conditions for d_0 (resp. f_N), then the first equation of our system is

$$4d_1 + d_2 = 6x_1 - x_0,$$

and the last equation is

$$f_{N-2} + 4f_{N-1} = 6y_{N-1} - y_N.$$

The matrix of the full system is of the form

$$\begin{pmatrix}
4 & 1 & 0 & & & & & & & \\
1 & 4 & 1 & & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & & \\
 & & 1 & 4 & 1 & & & & & \\
 & & & 4 & 15 & -1 & & & & \\
 & & & & -1 & 15 & 4 & & & \\
 & & & & & 1 & 4 & 1 & & \\
 & & & & & & \ddots & \ddots & \ddots & \\
 & & & & & & & 1 & 4 & 1 \\
 & & & & & & & 0 & 1 & 4
\end{pmatrix},$$

where the first submatrix has $M-1$ rows (the last one containing $4\,15\,-1$) and the second submatrix has $N-1$ rows (the first one containing $-1\,15\,4$), and when $M=N=3$, it is

$$\begin{pmatrix}
4 & 1 & 0 & 0 \\
4 & 15 & -1 & 0 \\
0 & -1 & 15 & 4 \\
0 & 0 & 1 & 4
\end{pmatrix}.$$

In the general case where $K \geq 2$, we have K sequences of data points $(x_0^k, \ldots, x_{M_k}^k)$, with $M_k \geq 3$. For each sequence of data points, we can compute de Boor points $d_0^1, \ldots, d_{M_k}^k$ determining an interpolating spline S_k. Again, for each k, the points d_0^k and $d_{M_k}^k$ are free, but we would like to enforce the conditions

$$\begin{aligned}
S_k'(x_{M_k}^k) &= S_{k+1}'(x_0^{k+1}) \\
S_k''(x_{M_k}^k) &= S_{k+1}''(x_0^{k+1}),
\end{aligned}$$

for $k = 1, \ldots, K-1$.

Our previous work in the case $K=2$ yields an interpolating spline with two free de Boor points d_0 and f_N, so we may use an inductive method and construct a system of equations for an interpolating spline S^{K-1} for $K-1$ sequences, and then use our method for two splines on S^{K-1} and S_K. Finally, d_0^0 and $d_{M_K}^K$ are determined using the natural end condition. We obtain a tridiagonal matrix of dimension $M_1 + \cdots + K_m - K$. For $k = 2, \ldots, K-1$, the row of index $M_1 + \cdots + M_{k-1} - k + 2$ contains the nonzero entries $-1, 15, 4$, and the row of index $M_1 + \cdots + M_k - k$ contains the nonzero entries $4, 15, -1$; these rows correspond to the first and the last $((M_k - 1)\text{th})$ equations associated with the kth sequence of data points. For $k=1$ the first row contains the nonzero entries $4, 1$ and row $M_1 - 1$ contains $4, 15, -1$, and for $k = K$, the last row contains the nonzero entries $1, 4$, and row $M_1 + \cdots + M_{K-1} - K + 2$ contains $-1, 15, 4$. Such matrices are row diagonally dominant, so they are invertible. Once we have computed the de Boor points $(d_1^k, \ldots, d_{M_k-1}^k)$ for $k = 1, \ldots, K$ by solving the linear system

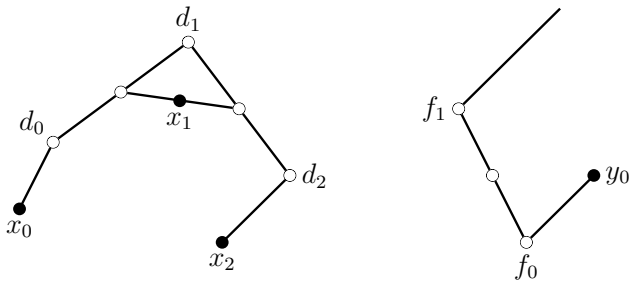


Figure 15: Splicing splines; the first spline has 3 data points.

that we just defined, we compute the de Boor points $d^k_{M_k}$ and d^{k+1}_0, for $k = 1, \ldots, K-1$, using formula $(*)$ and $(**)$; that is,

$$d^k_{M_k} = \frac{5}{6}x^k_{M_k} + \frac{1}{6}x^{k+1}_0 + \frac{1}{6}d^k_{M_k-1} - \frac{1}{6}d^{k+1}_1$$
$$d^{k+1}_0 = \frac{1}{6}x^k_{M_k} + \frac{5}{6}x^{k+1}_0 - \frac{1}{6}d^k_{M_k-1} + \frac{1}{6}d^{k+1}_1.$$

Finally, d^1_0 and $d^K_{M_K}$ are determined by using the natural end conditions.

If $K = 4$ and if the sequences of data points consists of $4, 4, 4, 5$ points, we obtain the following 9×9 matrix:

$$\begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 15 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 15 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 15 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 15 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 15 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 15 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{pmatrix}.$$

8.2 The First or the Last Spline Has 3 Data Points

If a cubic spline interpolates three data points, say x_0, x_1, x_2, then we have an equation relating d_0, d_1, d_2, namely

$$d_0 + 2d_1 + d_2 = 4x_1;$$

Suppose first that the first sequence has 3 data points; see Figure 15. Then, equation $(**)$ from the previous section still holds, so

$$d_2 = \frac{5}{6}x_2 + \frac{1}{6}y_0 + \frac{1}{6}d_1 - \frac{1}{6}f_1,$$

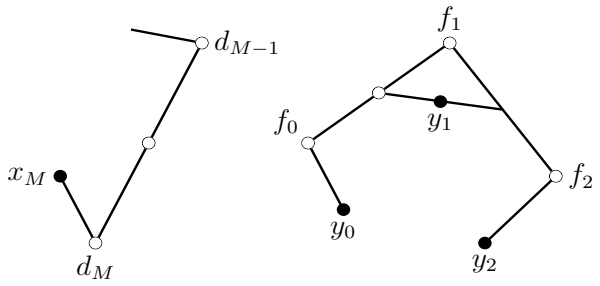


Figure 16: Splicing splines; the last spline has 3 data points.

so we get

$$d_0 + 2d_1 = 4x_1 - d_2 = 4x_1 - \frac{5}{6}x_2 - \frac{1}{6}y_0 - \frac{1}{6}d_1 + \frac{1}{6}f_1,$$

which yields

$$6d_0 + 13d_1 - f_1 = 24x_1 - 5x_2 - y_0.$$

If we prescribe that d_0 is given by the natural end condition, then

$$d_0 = \frac{2}{3}x_0 + \frac{1}{3}d_1,$$

so we obtain

$$15d_1 - f_1 = -4x_0 + 24x_1 - 5x_2 - y_0.$$

Next, if we assume that the last sequence has 3 data points (see Figure 16), then we may assume without loss of generality that $K = 2$, so

$$f_0 + 2f_1 + f_2 = 4y_1.$$

Equation $(*)$ from the previous section still holds, so

$$f_0 = \frac{1}{6}x_M + \frac{5}{6}y_0 - \frac{1}{6}d_{M-1} + \frac{1}{6}f_1,$$

so we get

$$2f_1 + f_2 = 4y_1 - f_0 = 4y_1 - \frac{1}{6}x_M - \frac{5}{6}y_0 + \frac{1}{6}d_{M-1} - \frac{1}{6}f_1,$$

which yields

$$-d_{M-1} + 13f_1 + 6f_2 = -x_M - 5y_0 + 24y_1.$$

If we prescribe that f_2 is given by the natural end condition, then

$$f_2 = \frac{1}{3}f_1 + \frac{2}{3}y_2,$$

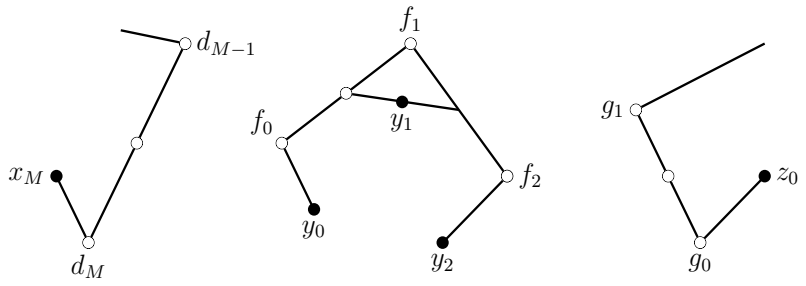


Figure 17: Splicing splines; some intermediate spline has 3 data points.

and we obtain

$$-d_{M-1} + 15f_1 = -x_M - 5y_0 + 24y_1 - 4y_2.$$

In conclusion, when the first (resp. the last) sequence of data points has 3 points, the first row 4 1 (resp. the last row 1 4) is replaced by 15 -1 (resp. -1 15), and the right hand side is also updated. These matrices are still row diagonally dominant, so they are invertible.

For example, if $K = 4$, and if the sequences of data points consist of $3, 5, 5, 3$ points, we get the following 8×8 matrix:

$$\begin{pmatrix} 15 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 15 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 15 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 15 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 15 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 15 \end{pmatrix}.$$

Last, we consider the case where $K \geq 3$ and the kth sequence of data points has 3 points, with $2 \leq k \leq K - 1$.

8.3 Some Intermediate Spline Has 3 Data Points

In this case, we are considering three consecutive sequences of data points and we may assume that the first sequence is $(x_0, \ldots, x_M)$, the second is (y_0, y_1, y_2), and the third is $(z_0, \ldots, z_N)$; see Figure 17. The corresponding de Boor control points are $(d_0, \ldots, d_N)$, (f_0, f_1, f_2), and $(g_0, \ldots, g_N)$. Here $M, N \geq 2$. Since equations $(*)$ and $(**)$ hold in all cases, applying them

to the mid sequence (for f_0 and f_2) , we get

$$f_0 = \frac{1}{6}x_M + \frac{5}{6}y_0 - \frac{1}{6}d_{M-1} + \frac{1}{6}f_1$$
$$f_2 = \frac{5}{6}y_2 + \frac{1}{6}z_0 + \frac{1}{6}f_1 - \frac{1}{6}g_1.$$

Since the mid sequence has 3 data points, we have

$$f_0 + 2f_1 + f_2 = 4y_1,$$

and by substituting the expressions for f_0 and f_2 in the above, we obtain

$$-d_{M-1} + 14f_1 - g_1 = -x_M - 5y_0 + 24y_1 - 5y_2 - z_0.$$

Therefore, every sequence (x_0^k, x_1^k, x_2^k) consisting of three data points, except the first and the last, yields a row of index $M_1 + \cdots + K_{k-1} + 2 - k$ of the matrix containing the nonzero entries $-1, 14, -1$. Other than that, the matrix of the system has the structure as in Section 8.2. Such matrices are row diagonally dominant, so they are invertible.

For example, if $K = 4$, and if the sequences of data points have, $5, 3, 3, 4$ elements, we obtain the following 7×7 matrix:

$$\begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 15 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 14 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 14 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 15 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{pmatrix}.$$

Figure 18 shows two splines with a discontinuity. The first spline interpolates 5 points and the second interpolates 6 points. Figure 19 shows three splines with two discontinuities. The first spline interpolates 3 points, the second interpolates 4 points, and the third interpolates 5 points. Figure 20 shows four splines with three discontinuities. The first spline interpolates 4 points, the second interpolates 3 points, the third interpolates 5 points, and the fourth interpolates 3 points.

9 Related Work

The problem of motion interpolation has been studied quite extensively both in the robotics and computer graphics communities. Since rotations in $\mathbf{SO}(3)$ can be represented by quaternions, the problem of quaternion interpolation has been investigated, an approach apparently initiated by Shoemake [17, 18], who extended the de Casteljau algorithm to the 3-sphere. Related work was done by Barr, Currin, Gabriel, and Hughes [2]. Kim, M.-J., Kim, M.-S.

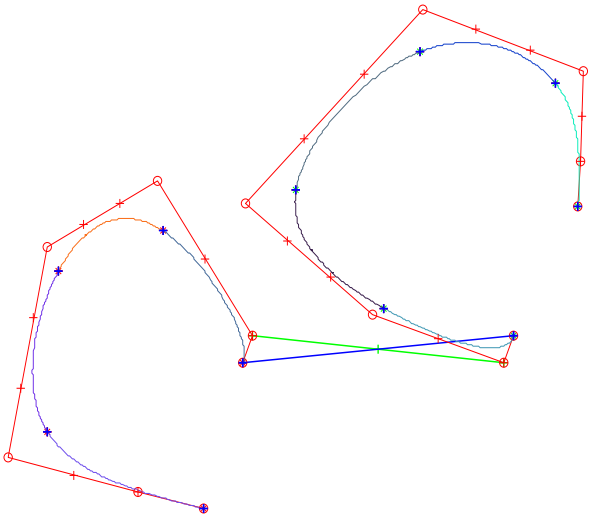


Figure 18: Examples of two splines with a discontinuity.

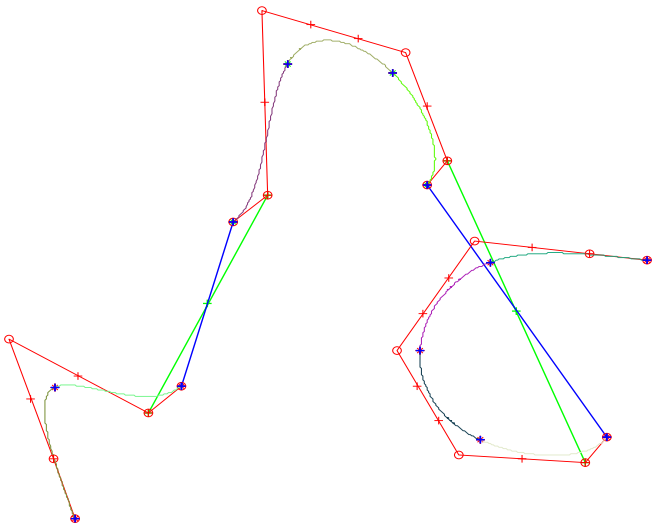


Figure 19: Examples of three splines with discontinuities.

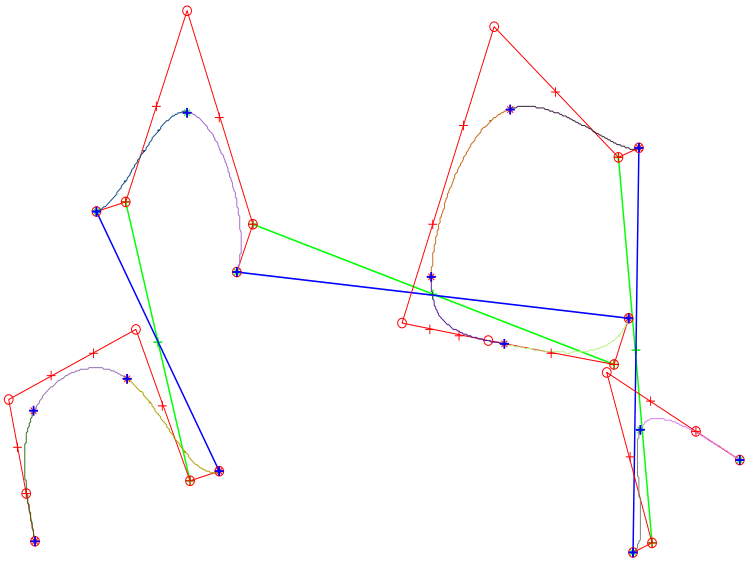


Figure 20: Example of four splines with discontinuities.

and Shin [11, 12] corrected bugs in Shoemake and introduced various kinds of splines on S^3, using the exponential map. Motion interpolation and rational motions have been investigated by Jüttler [7, 8], Jüttler and Wagner [9, 10], Horsch and Jüttler [6], and Röschel [16]. Park and Ravani [14, 15] also investigated Bézier curves on Riemannian manifolds and Lie groups, $\mathbf{SO}(3)$ in particular. Zefran, Kumar and Croke [19] formulated the problem interpolating between two given positions of a rigid body as a variational problem on the group of rigid body motions $\mathbf{SE}(3)$. The functional under consideration was a measure of smoothness of the three dimensional trajectory of the rigid body. Further work in this direction is presented in Belta and Kumar [3], and Altafini [1] gives a version of de Casteljau's algorithm for $\mathbf{SE}(3)$. None of these papers deal with the problem caused by the fact that the logarithm is multi-valued.

10 Computing the Length of Curves in $\mathbf{SO}(n)$

Assume that $\mathfrak{so}(n)$ is equipped with the inner product

$$\langle B_1, B_2 \rangle = \frac{1}{2}\mathrm{tr}(B_1^\top B_2) = -\frac{1}{2}\mathrm{tr}(B_1 B_2),$$

for any two elements $B_1, B_2 \in \mathfrak{so}(n)$ in the Lie algebra of $\mathbf{SO}(n)$. This inner product is $\mathrm{Ad}(\mathbf{SO}(n))$-invariant, so we obtain a bi-invariant metric on $\mathbf{SO}(n)$ (see Gallot, Hullin, Lafontaine [5] or O'Neill [13]). Let us consider the curve γ in $\mathbf{SO}(n)$ given by

$$\gamma(t) = e^{(1-t)B_1 + tB_2}, \quad t \in [0, 1],$$

with $R_1 = e^{B_1}$ and $R_2 = e^{B_2}$. We have $\gamma(0) = R_1$ and $\gamma(1) = R_2$. The length $L(\gamma)$ of the curve γ is

$$L(\gamma) = \int_0^1 \langle \gamma'(t), \gamma'(t) \rangle^{\frac{1}{2}} dt,$$

so we have

$$\begin{aligned}
L(\gamma) &= \int_0^1 \langle (e^{(1-t)B_1+tB_2})', (e^{(1-t)B_1+tB_2})' \rangle^{\frac{1}{2}} dt \\
&= \int_0^1 \langle (B_2 - B_1)e^{(1-t)B_1+tB_2}, (B_2 - B_1)e^{(1-t)B_1+tB_2}) \rangle^{\frac{1}{2}} dt \\
&= \int_0^1 \left(\frac{1}{2} \mathrm{tr}(((B_2 - B_1)e^{(1-t)B_1+tB_2})^\top (B_2 - B_1)e^{(1-t)B_1+tB_2}) \right)^{\frac{1}{2}} dt \\
&= \int_0^1 \left(\frac{1}{2} \mathrm{tr}(e^{((1-t)B_1+tB_2)^\top} (B_2 - B_1)^\top (B_2 - B_1)e^{(1-t)B_1+tB_2}) \right)^{\frac{1}{2}} dt \\
&= \int_0^1 \left(\frac{1}{2} \mathrm{tr}((B_2 - B_1)^\top (B_2 - B_1)e^{(1-t)B_1+tB_2} e^{-((1-t)B_1+tB_2)}) \right)^{\frac{1}{2}} dt \\
&= \Big(\frac{1}{2} \mathrm{tr}((B_2 - B_1)^\top (B_2 - B_1)) \Big)^{\frac{1}{2}} = \Big(\frac{1}{2} \mathrm{tr}(-(B_2 - B_1)^2) \Big)^{\frac{1}{2}}.
\end{aligned}$$

Thus, we obtain

$$L(\gamma) = \Big(-\frac{1}{2} \mathrm{tr}((B_2 - B_1)^2) \Big)^{\frac{1}{2}}.$$

Unless B_1 and B_2 commute, in which case

$$R_1^\top R_2 = (e^{B_1})^\top e^{B_2} = e^{-B_1} e^{B_2} = e^{B_2 - B_1},$$

the curve γ is generallly not a geodesic between R_1 and R_2. A geodesic from R_1 to R_2 is given by the curve

$$\gamma_g(t) = R_1 e^{tB}, \quad t \in [0, 1],$$

where $B \in \mathfrak{so}(n)$ is any log of $R_1^\top R_2$; that is, $e^B = R_1^\top R_2$ (see Gallot, Hullin, Lafontaine [5] or O'Neill [13]). Essentially the same computation as above yields

$$L(\gamma_g) = \Big(-\frac{1}{2} \mathrm{tr}(B^2) \Big)^{\frac{1}{2}}.$$

References

[1] Claudio Altafini. The de casteljau algorithm on **SE**(3). In *Nonlinear control in the year 2000*, pages 23–34. Springer, 2001.

[2] A.H. Barr, B. Currin, S. Gabriel, and J.F. Hughes. Smooth Interpolation of Orientations with Angular Velocity Constraints using Quaternions. In *Computer Graphics Proceedings, Annual Conference Series*, pages 313–320. ACM, 1992.

[3] Calin Belta and Vijay Kumar. An efficient geometric approach to rigid body motion interpolation. In *Proc. ASME 2000 Design Engineering Tech. Conf.* Citeseer, 2000.

[4] Jean H. Gallier. *Geometric Methods and Applications, For Computer Science and Engineering*. TAM, Vol. 38. Springer, second edition, 2011.

[5] S. Gallot, D. Hulin, and J. Lafontaine. *Riemannian Geometry*. Universitext. Springer Verlag, second edition, 1993.

[6] Thomas Horsch and Bert Jüttler. Cartesian spline interpolation for industrial robots. *Computer-Aided Design*, 30(3):217–224, 1998.

[7] Bert Jüttler. Visualization of moving objects using dual quaternion curves. *Computers & Graphics*, 18(3):315–326, 1994.

[8] Bert Jüttler. An osculating motion with second order contact for spacial Euclidean motions. *Mech. Mach. Theory*, 32(7):843–853, 1997.

[9] Bert Jüttler and M.G. Wagner. Computer-aided design with spacial rational B-spline motions. *Journal of Mechanical Design*, 118:193–201, 1996.

[10] Bert Jüttler and M.G. Wagner. Rational motion-based surface generation. *Computer-Aided Design*, 31:203–213, 1999.

[11] M.-J. Kim, M.-S. Kim, and S.Y. Shin. A general construction scheme for unit quaternion curves with simple high-order derivatives. In *Computer Graphics Proceedings, Annual Conference Series*, pages 369–376. ACM, 1995.

[12] M.-J. Kim, M.-S. Kim, and S.Y. Shin. A compact differential formula for the first derivative of a unit quaternion curve. *Journal of Visualization and Computer Animation*, 7:43–57, 1996.

[13] Barrett O'Neill. *Semi-Riemannian Geometry With Applications to Relativity*. Pure and Applies Math., Vol 103. Academic Press, first edition, 1983.

[14] F.C. Park and B. Ravani. Bézier curves on Riemannian manifolds and Lie groups with kinematic applications. *ASME J. Mech. Des.*, 117:36–40, 1995.

[15] F.C. Park and B. Ravani. Smooth invariant interpolation of rotations. *ACM Transactions on Graphics*, 16:277–295, 1997.

[16] Otto Röschel. Rational motion design: A survey. *Computer-Aided Design*, 30(3):169–178, 1998.

[17] Ken Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH'85*, volume 19, pages 245–254. ACM, 1985.

[18] Ken Shoemake. Quaternion calculus for animation. In *Math for SIGGRAPH*, pages 1–19. ACM, 1991. Course Note No. 2.

[19] Milos Zefran, Vijay Kumar, and Christopher Croke. On the generation of smooth three-dimensional rigid body motions. *IEEE T. Robotics and Automation*, 14(4):576–589, 1998.