Lie Groups and Lie Algebras

David Eberly, Geometric Tools, Redmond WA 98052
https://www.geometrictools.com/

Created: May 21, 2024

Contents

1 Lie Groups

The topic of *Lie groups* is in the realm of advanced mathematics, but currently Geometric Tools support is provided only for groups related to rigid motions (rotations and translations). In this context, a Lie group is a set of $n \times n$ matrices that is parameterized by k variables, $1 \le k < n^2$, and is closed under multiplication: given two matrices in the set, their product is also in the set. The matrices are also required to be invertible. A brief summary of Lie groups and Lie algebras with fewer details but more groups is found online in *Lie Groups for Computer Vision* [1].

2 Lie Algebras

Given a Lie group of $n \times n$ matrices parameterized by k parameters, the *Lie algebra* is the vector space of differential transformations at the identity matrix I. The Lie algebra is also referred to as the *tangent space* at the identity. It is a k-dimensional vector space with a matrix representation that has basis $\{G_i\}_{i=0}^{k-1}$, where each G_i is an $n \times n$ matrix called a *generator*. A Lie algebra element is represented by a k-tuple

$$\boldsymbol{x} = (x_0, \dots, x_{k-1}) \tag{1}$$

whose components are the parameters for the space. At times in the discussion, the Lie algebra element is referred to as a $k \times 1$ vector when the element requires vector or matrix-vector operations; this supports operations that involve vectors or matrices. The Lie algebra element $\boldsymbol{x}$ can be mapped to a matrix representation using the linear combination,

$$L(\boldsymbol{x}) = X = \sum_{i=0}^{k-1} x_i G_i \tag{2}$$

Given the matrix X, the equation $L(\boldsymbol{x}) = X$ can be solved for the components x_i of $\boldsymbol{x}$. The function that represents this is an inverse operation,

$$\boldsymbol{x} = L^{-1}(X) = L^{-1}\left(\sum_{i=0}^{k-1} x_i G_i\right) \tag{3}$$

3 Exponential and Logarithm Maps

The *exponential map* of a Lie group element X is

$$\exp(X) = \sum_{k=0}^{\infty} \frac{X^k}{k!} = I + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \cdots \tag{4}$$

A closed-form representation exists for some matrices such as diagonal matrices and skew-symmetric matrices. Generally, the power series converges and it is always possible to compute $\exp(X)$ using the $S + N$ decomposition [3, Chapter 6]. The matrix X can be decomposed into $X = S + N$ where S is semisimple (diagonalizable), where N is nilpotent ($N^p = 0$ for some $p \ge 1$ and $N^q \ne 0$ for $q < p$) and where $SN = NS$. For real numbers x and y, it is true that $\exp(x + y) = \exp(x)\exp(y)$. The same equality is generally not true for matrices A and B; that is, $\exp(A + B)$ is not always equal to $\exp(A)\exp(B)$. The equality is true, however, when $AB = BA$. In the $S + N$ decomposition, the condition $SN = NS$ ensures

that $\exp(X) = \exp(S+N) = \exp(S)\exp(N)$. The matrix S is diagonalizable, which means $S = QDQ^{-1}$ for some diagonal matrix $D = \text{Diagonal}(d_0, \ldots, d_{n-1})$ and an invertible matrix Q. It follows that

$$\exp(S) = \exp(QDQ^{-1}) = Q\exp(D)Q^{-1} = Q\,\text{Diagonal}(\exp(d_0), \ldots, \exp(d_{n-1}))Q^{-1} \tag{5}$$

The matrix N is nilpotent: $N^k = 0$ for $k \geq p \geq 0$ and $N^i \neq 0$ for $k < p$, which ensures $\exp(N)$ is a finite sum,

$$\exp(N) = \sum_{k=0}^{\infty} \frac{N^k}{k!} = \sum_{k=0}^{p-1} \frac{N^k}{k!} \tag{6}$$

If $Y = \exp(X)$, the *logarithm map* produces X from a specified Y, say,

$$Y = \exp(X), \quad X = \log(Y) \tag{7}$$

The following suggestive notation is used. If $\boldsymbol{x}$ is the Lie algebra element corresponding to the Lie group element X, define $Y = \widehat{\exp}(\boldsymbol{x}) = \exp(L(\boldsymbol{x})) = \exp(X)$, where L is defined in equation (2). Similarly, define $L^{-1}(X) = \boldsymbol{x} = \widehat{\log}(Y)$, where L^{-1} is defined in equation (3). Equivalently, $\log(Y) = X = L(\boldsymbol{x}) = L(\widehat{\log}(Y))$. In terms of composition notation for functions, $\widehat{\exp} = \exp \circ L$ and $\widehat{\log} = L^{-1} \circ \log$. The Geometric Tools code implements

$$Y = \widehat{\exp}(\boldsymbol{x}), \quad \boldsymbol{x} = \widehat{\log}(Y) \tag{8}$$

instead of $Y = \exp(X)$ and $X = \log(Y)$ in equation (7).

4 Geodesics and Shortest-Path Interpolation

The exponential and logarithm maps allows creating a geodesic path connecting two Lie group matrices, which supports interpolation between the matrices along a shortest path. This generalizes the concepts of linear interpolation between two points along the line segment connecting them and of spherical linear interpolation between two quaternions along the great circle arc connecting them.

Generally, if M_0 and M_1 are Lie group elements, the geodesic path connecting them is parameterized by

$$F(t; M_0, M_1) = \exp(t\log(M_1 M_0^{-1}))M_0, \quad t \in [0,1] \tag{9}$$

Observe that $F(0; M_0, M_1) = M_0$ and $F(1; M_0, M_1) = M_1$. The source code implementation has two variations on computing geodesic paths, one where you compute $M_1 M_0^{-1}$ for a single value of t and one where $M_1 M_0^{-1}$ is precomputed and used for multiple values of t.

5 The Adjoint Representation

The Lie algebra elements can be thought of as vectors in the tangent space at the identity element of the Lie group, and the *adjoint representation* transforms tangent vectors $L(\boldsymbol{x})$ to tangent vectors $L(\boldsymbol{y})$. The transformation from $L(\boldsymbol{x})$ to $L(\boldsymbol{y})$ depends on the choice of matrix M,

$$L(\boldsymbol{y}) = ML(\boldsymbol{x})M^{-1} \tag{10}$$

Observe that $ML(\boldsymbol{x}) = L(\boldsymbol{y})M$. The idea is to compute $L(\boldsymbol{y})$ so that the premultiplication of $L(\boldsymbol{x})$ by M becomes a postmultiplication of $L(\boldsymbol{y})$ by M.

Using equation (2), $\boldsymbol{x}$ and $\boldsymbol{y}$ are $k \times 1$ vectors, and the coordinates are the coefficients of linear combinations of the generators. Equation (10) becomes

$$\sum_{i=0}^{k-1} y_i G_i = M\left(\sum_{j=0}^{k-1} x_j G_j\right) M^{-1} = \sum_{j=0}^{k-1} x_j \left(M G_j M^{-1}\right) = \sum_{j=0}^{k-1} x_j \sum_{i=0}^{k-1} a_{ij}(M) G_i = \sum_{i=0}^{k-1}\left(\sum_{j=0}^{k-1} a_{ij}(M) x_j\right) G_i \tag{11}$$

where MG_jM^{-1} is a $k \times k$ matrix that is written as a linear combination of the generators G_i with coefficients given by $a_{ij}(M)$. The notation for the coefficients stresses that they depend on the choice of M.

Equation (11) is a system of linear equations

$$\boldsymbol{y} = A(M)\boldsymbol{x}, \quad y_i = \sum_{j=0}^{k-1} a_{ij}(M) x_j \quad \text{for } 0 \le i < k \tag{12}$$

The matrix M is $n \times n$. The matrix $A(M) = [a_{ij}(M)]$ is $k \times k$ and corresponds to the adjoint representation relative to M. Also, the matrix depends on the order of the generators; a reordering of the generators leads to a permutation of elements of $A(M)$.

6 A Common Class Interface for Lie Groups and Lie Algebras

For each Lie group and Lie algebra, a class is provided that supports the computations of equations (2), (3), (4), (7), (9), and (12). Listing 1 shows the interface.

Listing 1. The C++ interface that is common for the Lie groups and Lie algebras supported by Geometric Tools. For each Lie group, suffix GROUPNAME is chosen appropriately for the implementation of that group's operations. Note: The order of template parameters for Vector and Matrix is for GTL code. GTE code lists the dimensions before the type T.

```
template <typename T>
class LieGROUPNAME
{
public:
    // n is the dimension of the Lie group, k is the dimension of the Lie algebra, c are the Lie algebra element coefficients.
    using AlgebraType = Vector<T, k>;
    using AdjointType = Matrix<T, k, k>;
    using GroupType = Matrix<T, n, n>;

    // Compute the Lie group element X from the Lie algebra element x using X = L(x).
    static GroupType ToGroup(AlgebraType const& x);

    // Compute the Lie algebra element x from the Lie group element X using x = L^{-1}(X).
    static AlgebraType InverseL(GroupType const& G);

    // Compute the exponential map of the Lie algebra element x to produce the Lie group element Y = exp(X) = exp(L(x)).
    static GroupType Exp(AlgebraType const& c);

    // Compute the logarithm map of the Lie group element Y to produce the Lie algebra element x corresponding to the Lie
    // group element X.
    static AlgebraType Log(GroupType const& M);
```

```
    // Compute the adjoint matrix A(M) from the Lie group element M.
    static AdjointType Adjoint(GroupType const& M);

    // Compute log(M1*M0^{-1}) to reduce computation time when you want to evaluate GeodesicPath for multiple values of t.
    static AlgebraType LogM1M0Inv(GroupType const& M0, GroupType const& M1);

    // Compute a point on the geodesic path from M0 to M1. The expression log(M1*M0^{-1}) is computed for each call to the
    // function. Use this GeodesicPath when it is needed for only a single value of t.
    static GroupType GeodesicPath(T const& t, GroupType const& M0, GroupType const& M1);

    // Compute a point on the geodesic path from M0 to M1. The Lie algebra element log(M1*M0^{-1}) must be precomputed
    // by the caller. Use this GeodesicPath when it is needed for multiple values of t.
    static GroupType GeodesicPath(T const& t, GroupType const& M0, AlgebraType const& logM1M0Inv);
};
```

The next sections describe the mathematics for those Lie groups and Lie algebras currently supported by Geometric Tools. Numerical issues are addressed in the implementations, especially those related to extracting an axis and angle from a rotation matrix when the angle is nearly zero.

7 Lie Group $SO(2)$

$SO(2)$ is the Lie group for 2D rotations ($n = 2$) and $so(2)$ is its corresponding Lie algebra ($k = 1$). For each rotation R there is a skew-symmetrix matrix S for which $R = \exp(S)$. R is the Lie group element and S is the Lie algebra matrix corresponding to Lie algebra element $\boldsymbol{x} = (\theta)$. A basis for the generators has only a single element,

$$G_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{13}$$

The skew-symmetric matrix is $S = xG_0 = \theta G_0$,

$$S(\theta) = \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix} \tag{14}$$

where the notation indicates that S depends functionally on θ.

7.1 Exponential

The rotation matrix $R = \exp(S)$ can be computed by using the Taylor series for $\exp(x)$, reducing each term using the identity $S^2 = -\theta^2 I$ for 2×2 identity matrix I, and grouping together the I terms and the S terms,

$$R(\theta) = (\cos\theta)I + \left(\frac{\sin\theta}{\theta}\right)S(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{15}$$

7.2 Logarithm

The logarithm of $R = \exp(S)$ is computed by extracting the angle from R. This is simply $\theta = \text{atan2}(r_{10}, r_{00}) = \text{atan2}(\sin\theta, \cos\theta)$.

7.3 Adjoint

In equations (10) and (10), let $M = R(\theta)$ and let $\boldsymbol{x} = [\phi]$; then

$$MG_0M^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = 1 \cdot G_0 \tag{16}$$

in which case $a_{00} = 1$. The adjoint matrix representation is

$$A_M = [1], \quad A_M\boldsymbol{x} = [1][\phi] = [\phi] = \boldsymbol{y} \tag{17}$$

8 Lie Group $SE(2)$

$SE(2)$ is the Lie group for 2D rigid transformations stored as homogeneous 3×3 matrices ($n = 3$) where the upper-left 2×2 block is a rotation matrix R, the upper-right 2×1 block is a translation vector $\boldsymbol{T}$, the lower-left block is a 1×2 vector of zeros, and the lower-right block is a 1×1 block with element 1. $se(2)$ is the corresponding Lie algebra ($k = 3$). R is associated with the skew-symmetric matrix of equation (14) with $R = \exp(S)$. $\boldsymbol{T}$ has an associated 2×1 vector $\boldsymbol{u} = (u_0, u_1)$, written here as a 2-tuple, which occurs in the exponential map that represents the rigid transformation.

A basis for the generators is

$$G_0 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad G_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{18}$$

The first matrix generates the rotations and the last two matrices generate the translations. The Lie algebra element is a 3-tuple $\boldsymbol{x} = (\theta, u_0, u_1)$. The corresponding matrix generated by $L(\boldsymbol{x})$ is

$$X = \begin{bmatrix} S(\theta) & \boldsymbol{u} \\ \boldsymbol{0}^\mathsf{T} & 0 \end{bmatrix} \tag{19}$$

8.1 Exponential

The rigid transformation is $M = \exp(X)$,

$$M(\theta) = \begin{bmatrix} \exp(S(\theta)) & P(\theta)\boldsymbol{u} \\ \boldsymbol{0}^\mathsf{T} & 1 \end{bmatrix} = \begin{bmatrix} R(\theta) & \boldsymbol{T} \\ \boldsymbol{0}^\mathsf{T} & 1 \end{bmatrix} \tag{20}$$

where $\boldsymbol{u}$ is the 2×1 vector with rows u_0 and u_1, $\boldsymbol{0}^\mathsf{T}$ is the 1×2 zero vector, $R(\theta)$ is the rotation matrix of equation (15), and

$$P(\theta) = \begin{bmatrix} \frac{\sin\theta}{\theta} & -\frac{1-\cos\theta}{\theta} \\ \frac{1-\cos\theta}{\theta} & \frac{\sin\theta}{\theta} \end{bmatrix} = \begin{bmatrix} F_0(\theta) & -\theta F_1(\theta) \\ \theta F_1(\theta) & F_0(\theta) \end{bmatrix} \tag{21}$$

where $F_0(\theta) = (\sin\theta)/\theta$, which is also known as the *sinc* function, and $F_1(\theta) = (1-\cos\theta)/\theta^2$. It is the case that $\boldsymbol{T} = P(\theta)\boldsymbol{u}$ and $\boldsymbol{u} = P^{-1}(\theta)\boldsymbol{T}$.

8.2 Logarithm

The angle for the rotation matrix R is extracted using $x_0 = \theta = \text{atan2}(r_{10}, r_{00})$, which is the $SO(2)$ logarithm of R. The $(x_1, x_2) = \boldsymbol{u}$ portion of the $se(2)$ element is the solution to $P(\theta)\boldsymbol{u} = \boldsymbol{T}$ where $\boldsymbol{T}$ is the translation and $P(\theta)$ is the matrix of equation (21). A simple inversion produces $\boldsymbol{u} = P^{-1}(\theta)\boldsymbol{T}$.

8.3 Adjoint

In equations (10) and (11), let

$$M = \begin{bmatrix} R & \boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{bmatrix}, \quad M^{-1} = \begin{bmatrix} R^{\mathsf{T}} & -R^{\mathsf{T}}\boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_0 \\ \hline x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \phi \\ \hline \boldsymbol{v} \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_0 \\ \hline y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \psi \\ \hline \boldsymbol{w} \end{bmatrix} \tag{22}$$

The adjoint representation in terms of the generators is shown next,

$$\begin{aligned} \textstyle\sum_{i=0}^{2} y_i G_i &= M\left(\textstyle\sum_{i=0}^{2} x_i G_i\right) M^{-1} \\ \begin{bmatrix} S(\psi) & \boldsymbol{w} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} &= \begin{bmatrix} R & \boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} S(\phi) & \boldsymbol{v} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} R^{\mathsf{T}} & -R^{\mathsf{T}}\boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{bmatrix} \\ &= \begin{bmatrix} RS(\phi)R^{\mathsf{T}} & R\boldsymbol{v} - RS(\phi)R^{\mathsf{T}}\boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} \\ , \\ &= \begin{bmatrix} S(\phi) & R\boldsymbol{v} - S(\phi)\boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} \end{aligned} \tag{23}$$

The upper-left blocks are equal, $S(\psi) = S(\phi)$, which implies $\psi = \phi$. The upper-right blocks are equal, which implies $\boldsymbol{w} = R\boldsymbol{v} - S(\phi)\boldsymbol{T} = R\boldsymbol{v} + \phi\boldsymbol{T}^{\perp}$ where as 2-tuples, $\boldsymbol{T} = (t_0, t_1)$ and $\boldsymbol{T}^{\perp} = (t_1, -t_0)$. The adjoint representation is therefore

$$\boldsymbol{y} = \begin{bmatrix} \psi \\ \hline \boldsymbol{w} \end{bmatrix} = \left[\begin{array}{c|c} 1 & \boldsymbol{0}^{\mathsf{T}} \\ \hline \boldsymbol{T}^{\perp} & R \end{array}\right] \begin{bmatrix} \phi \\ \hline \boldsymbol{v} \end{bmatrix} \boldsymbol{x} = A_M \boldsymbol{x} \tag{24}$$

where 1 is the 1×1 scalar and $\boldsymbol{0}^{\mathsf{T}}$ is the 1×2 vector with both components equal to 0. Observe that A_M of equation (24) is a permutation of the one provided in [1]. The generators in this document are G_0, G_1, and G_2. These occur in a different order in [1]: G_1, G_2, and G_0.

9 Lie Group $SO(3)$

$SO(3)$ is the Lie group for 3D rotations ($n = 3$) and $so(3)$ is its corresponding Lie algebra ($k = 3$). For each rotation R there is a skew-symmetric matrix S for which $R = \exp(S)$. R is the Lie group element and S is the Lie algebra matrix corresponding to Lie algebra element $\boldsymbol{x} = (x_0, x_1, x_2)$. A basis for the generators is

$$G_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad G_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{25}$$

The skew-symmetric matrix is

$$S(\boldsymbol{x}) = x_0 G_0 + x_1 G_1 + x_2 G_2 = \begin{bmatrix} 0 & -x_2 & +x_1 \\ +x_2 & 0 & -x_0 \\ -x_1 & +x_0 & 0 \end{bmatrix} \tag{26}$$

9.1 Exponential

The rotation matrix $R = \exp(S(\boldsymbol{x}))$ can be computed by using the Taylor series for $\exp(x)$. Define $\theta = |\boldsymbol{x}|$ and observe that $S^3(\boldsymbol{x}) = -\theta^2 S(\boldsymbol{x})$, which allows reduction of the Taylor series to

$$R = I + \left(\frac{\sin\theta}{\theta}\right) S(\boldsymbol{x}) + \left(\frac{1-\cos\theta}{\theta^2}\right) S^2(\boldsymbol{x}) = I + F_0(\theta) S(\boldsymbol{x}) + F_1(\theta) S^2(\boldsymbol{x}) \tag{27}$$

where I is the 3×3 identity matrix and where $F_0(\theta) = (\sin\theta)/\theta$ and $F_1(\theta) = (1 - \cos\theta)/\theta^2$

9.2 Logarithm

The angle θ is computed using the observation that the trace of the rotation matrix R in equation (27) is $\text{Trace}(R) = 3 - 2((1-\cos\theta)/\theta^2)(x_0^2 + x_1^2 + x_2^2) = 1 + 2\cos\theta$. This can be solved for $\cos\theta = (\text{Trace}(R) - 1)/2$.

When $\cos\theta \in (-1, 1)$, the angle is chosen to be $\theta = \text{acos}((\text{Trace}(R) - 1)/2)$. The skew-symmetric matrix is $S = (\theta/(2\sin\theta))(R - R^{\mathsf{T}})$. The Lie algebra element is

$$(x_0, x_1, x_2) = \frac{1}{2F_0(\theta)} \, (r_{21} - r_{12}, r_{02} - r_{20}, r_{10} - r_{01}) \tag{28}$$

When $\theta = 0$, the Lie algebra element is $(x_0, x_1, x_2) = (0, 0, 0)$ because R is the identity matrix.

When $\theta = \pi$, observe that $R = I + (2/\pi^2)S^2$. Representing $\boldsymbol{x}$ as a 3×1 vector, it is the case that $(2/\pi^2)\boldsymbol{x}\boldsymbol{x}^{\mathsf{T}} = R + I$. The right-hand side is a symmetric matrix with positive diagonal entries and rank 1. For a numerically robust implementation, choose the row of $R + I$ that has the largest diagonal term and normalize that row. Multiply it by $\pi/\sqrt{2}$ to obtain $\boldsymbol{x}$ from which $S = L(\boldsymbol{x})$. The vector $-\boldsymbol{x}$ is also a candidate, but it is irrelevant because $\boldsymbol{x}$ and $-\boldsymbol{x}$ produce the same rotation matrix when $\theta = \pi$. Knowing $R + I$ is symmetric, and wanting to avoid numerical bias, use $(r_{ij} + r_{ji})/2$ for off-diagonal entries rather than r_{ij} by itself.

9.3 Adjoint

In equations (10) and (11), let $M = R$, $M^{-1} = R^{\mathsf{T}}$, and let $\boldsymbol{x}$ and $\boldsymbol{y}$ be 3-tuple Lie algebra elements. The adjoint representation in terms of the generators is shown next, where $\boldsymbol{\rho}_i$ is the i-th row of R,

$$S(\boldsymbol{y}) = \sum_{i=0}^{2} y_i G_i = M \left(\sum_{i=0}^{2} x_i G_i \right) M^{-1} = RS(\boldsymbol{x})R^{\mathsf{T}} \tag{29}$$

Some algebraic manipulation will show that

$$RS(\boldsymbol{x})R^{\mathsf{T}} = S(\boldsymbol{\rho}_0 \cdot \boldsymbol{x}, \boldsymbol{\rho}_1 \cdot \boldsymbol{x}, \boldsymbol{\rho}_2 \cdot \boldsymbol{x}) = S\left(R\boldsymbol{x}\right) \tag{30}$$

The manipulation uses $\boldsymbol{\rho}_0 = \boldsymbol{\rho_1} \times \boldsymbol{\rho}_2$, $\boldsymbol{\rho}_1 = \boldsymbol{\rho_2} \times \boldsymbol{\rho}_0$, and $\boldsymbol{\rho}_2 = \boldsymbol{\rho_0} \times \boldsymbol{\rho}_1$. The equation $S(\boldsymbol{y}) = S(R\boldsymbol{x})$ implies that the adjoint representation is

$$\boldsymbol{y} = R\boldsymbol{x} = A_M \boldsymbol{x} \tag{31}$$

10 Lie Group $SE(3)$

$SE(3)$ is the Lie group for 3D rigid transformations stored as homogeneous 4×4 matrices ($n = 4$) where the upper-left 3×3 block is a rotation matrix R, the upper-right 3×1 block is a translation vector $\boldsymbol{T}$, the lower-left block is a 1×3 vector of zeros, and the lower-right block is a 1×1 block with element 1. $se(3)$ is the corresponding Lie algebra ($k = 6$).

A basis for the generators is

$$G_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G_3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{32}$$

The Lie algebra element is a 6-tuple $\boldsymbol{x} = (z_0, z_1, z_2, u_0, u_1, u_2) = (\boldsymbol{z}, \boldsymbol{u})$, where $\boldsymbol{z}$ is associated with rotations and $\boldsymbol{u}$ is associated with translations. The corresponding matrix generated by $L(\boldsymbol{X})$ is

$$X = \begin{bmatrix} S(\boldsymbol{z}) & \boldsymbol{u} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} \tag{33}$$

where $S(\boldsymbol{z})$ is defined by equation (26). The powers of X are

$$X^0 = I, \;\; X^i = \begin{bmatrix} S(\boldsymbol{z}) & \boldsymbol{u} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix}^i = \begin{bmatrix} S^i(\boldsymbol{z}) & S^{i-1}(\boldsymbol{z})\boldsymbol{u} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix} \text{ for } i > 0 \tag{34}$$

10.1 Exponential

The rigid transformation is $M = \exp(X)$,

$$M = \begin{bmatrix} \exp(S(\boldsymbol{z})) & Q(\boldsymbol{z})\boldsymbol{u} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} = \begin{bmatrix} R & \boldsymbol{T} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \tag{35}$$

where $\theta = |\boldsymbol{x}|$ is the length of the Lie algebra element and

$$Q(\boldsymbol{z}) = I + \left(\frac{1-\cos\theta}{\theta^2}\right) S(\boldsymbol{z}) + \left(\frac{\theta - \sin\theta}{\theta^3}\right) S^2(\boldsymbol{z}) = I + F_1(\theta)S(\boldsymbol{z}) + F_2(\theta)S^2(\boldsymbol{z}) \tag{36}$$

where $F_1(\theta) = (1-\cos\theta)/\theta^2$ and $F_3(\theta) = (\theta - \sin\theta)/\theta^3$. The function $F_1(\theta)$ was introduced in equation (21) for computing the matrix $P(\theta)$. The construction of $Q(\boldsymbol{z})$ uses $\theta = |\boldsymbol{z}|$ and $S^3(\boldsymbol{z}) = -\theta^2 S(\boldsymbol{z})$ in the Taylor series for $\exp(M)$.

10.2 Logarithm

Let M be the rigid transformation of equation (35) with rotation matrix R and translation vector $\boldsymbol{T}$. The function $\log(M)$ can be computed by extracting the Lie algebra element $\boldsymbol{z} = \log(R)$. The matrix $Q(\boldsymbol{z})$ is computed using equation (36) and then inverted and applied to the translation vector to obtain $\boldsymbol{u} = Q^{-1}(\boldsymbol{z})\boldsymbol{T}$.

10.3 Adjoint

In equations (10) and (11), let

$$M = \begin{bmatrix} R & \boldsymbol{T} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}, \;\; M^{-1} = \begin{bmatrix} R^\mathsf{T} & -R^\mathsf{T}\boldsymbol{T} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}, \;\; \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \hline x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \boldsymbol{d} \\ \hline \boldsymbol{v} \end{bmatrix}, \;\; \boldsymbol{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \hline y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e} \\ \hline \boldsymbol{w} \end{bmatrix} \tag{37}$$

The adjoint representation in terms of the generators is shown next, where $\boldsymbol{\rho}_i$ is the i-th row of R,

$$\begin{aligned}
\textstyle\sum_{i=0}^{5} y_i G_i &= M\left(\textstyle\sum_{i=0}^{5} x_i G_i\right) M^{-1} \\
\begin{bmatrix} S(\boldsymbol{e}) & \boldsymbol{w} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} &= \begin{bmatrix} R & \boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} S(\boldsymbol{d}) & \boldsymbol{v} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} R^{\mathsf{T}} & -R^{\mathsf{T}}\boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{bmatrix} \\
&= \begin{bmatrix} RS(\boldsymbol{d})R^{\mathsf{T}} & R\boldsymbol{v} - RS(\boldsymbol{d})R^{\mathsf{T}}\boldsymbol{T} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix}
\end{aligned} \tag{38}$$

Some algebraic manipulation will show that

$$RS(\boldsymbol{d})R^{\mathsf{T}} = S(\boldsymbol{\rho}_0 \cdot \boldsymbol{d}, \boldsymbol{\rho}_1 \cdot \boldsymbol{d}, \boldsymbol{\rho}_2 \cdot \boldsymbol{d}) = S\left(R\boldsymbol{d}\right), \;\; -RS(\boldsymbol{d})R^{\mathsf{T}}\boldsymbol{T} = S(\boldsymbol{T})R\boldsymbol{d} \tag{39}$$

The manipulation uses $\boldsymbol{\rho}_0 = \boldsymbol{\rho_1} \times \boldsymbol{\rho}_2$, $\boldsymbol{\rho}_1 = \boldsymbol{\rho_2} \times \boldsymbol{\rho}_0$, and $\boldsymbol{\rho}_2 = \boldsymbol{\rho_0} \times \boldsymbol{\rho}_1$. Equation (38) reduces to

$$\begin{bmatrix} S(\boldsymbol{e}) & \boldsymbol{w} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} = \begin{bmatrix} S\left(R\boldsymbol{d}\right) & R\boldsymbol{v} + S(\boldsymbol{T})R\boldsymbol{d} \\ \boldsymbol{0}^{\mathsf{T}} & 0 \end{bmatrix} \tag{40}$$

The upper-left blocks are equal, $S(\boldsymbol{e}) = S(R\boldsymbol{d})$, which implies $\boldsymbol{e} = R\boldsymbol{d}$. The upper-right blocks are equal, which implies $\boldsymbol{w} = R\boldsymbol{v} + S(\boldsymbol{T})R\boldsymbol{d}$. The adjoint representation is therefore

$$\boldsymbol{y} = \left[\begin{array}{c} \boldsymbol{e} \\ \hline \boldsymbol{w} \end{array}\right] = \left[\begin{array}{c|c} R & Z \\ \hline S(\boldsymbol{T})R & R \end{array}\right] \left[\begin{array}{c} \boldsymbol{d} \\ \hline \boldsymbol{v} \end{array}\right] \boldsymbol{x} = A_M \boldsymbol{x} \tag{41}$$

where Z is the 3×3 matrix whose elements are all 0.

Observe that A_M of equation (41) is a permutation of the one provided in [1]. The generators in this document are G_0 through G_5. These occur in a different order in [1]: G_3, G_4, G_5, G_0, G_1, and G_2.

11 Robust Computation of F_0, F_1, and F_2

The functions $F_0(\theta)$, $F_1(\theta)$, and $F_2(\theta)$ have removable singularities at $\theta = 0$. The values at zero can be computed using Taylor series expansions at $\theta = 0$,

$$\begin{aligned}
F_0(\theta) &= \textstyle\sum_{i=0}^{\infty} \frac{(-1)^i \theta^{2i}}{(2i+1)!} &&= 1 - \frac{\theta^2}{3!} + \frac{\theta^4}{5!} - \ldots \\
F_1(\theta) &= \textstyle\sum_{i=0}^{\infty} \frac{(-1)^i \theta^{2i}}{(2i+2)!} &&= \frac{1}{2!} - \frac{\theta^2}{4!} + \frac{\theta^4}{6!} - \ldots \\
F_2(\theta) &= \textstyle\sum_{i=0}^{\infty} \frac{(-1)^i \theta^{2i}}{(2i+3)!} &&= \frac{1}{3!} - \frac{\theta^2}{5!} + \frac{\theta^6}{7!} - \ldots
\end{aligned} \tag{42}$$

Evaluating the Taylor series expansions at 0 leads to $F_0(0) = 1$, $F_1(0) = 1/2$, and $F_2(0) = 1/6$.

11.1 Robust Computation of F_0

Evaluation of the expressions $\sin\theta/\theta$, $(1-\cos\theta)/\theta^2$, and $(\theta-\sin\theta)/\theta^3$ near a removable singularity can be a problem when computing with floating-point arithmetic. Based on an analysis in [2, Section 6.1.1], $F_0(\theta)$ is accurate for float numbers near 0, even when those numbers are subnormal. It is sufficient to implement $F_0(\theta)$ by the code in listing 2,

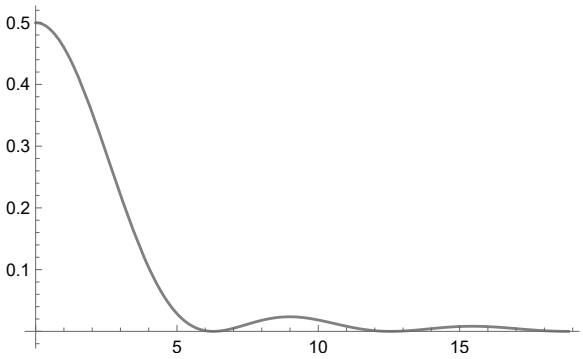
Listing 2. An accurate implementation of $F_0(\theta)$ for $\theta \geq 0$ using floating-point type float.

```
float F0(float theta)
{
    return (theta != 0.0f ? std::sin(theta) / theta : 1.0f);
}
```

11.2 Robust Computation of F_1

Evaluation of $F_1(\theta)$ has significant problems. The analysis of the direct evaluation is also in [2, Section 6.1.1]. At a large scale, the graph of $F_1(\theta)$ is shown in figure 1

Figure 1. The graph of $F_1(\theta)$ for $\theta > 0$ drawn at large scale. The plot was drawn using Mathematica [5].



The näive implementation of $F_1(\theta)$ for floating-point type float is shown in listing 3,

Listing 3. The näive implementation of $F_1(\theta)$ for floating-point type float.

```
float F1(float theta)
{
    return (theta != 0.0f ? (1.0f - std::cos(theta)) / (theta * theta) : 0.5f);
}
```

The function can be evaluated for all finite and positive float numbers. The experiment was performed using Microsoft Visual Studio 2022 17.9.6 in the Debug x64 configuration. The encodings are according to the IEEE Standard for Floating-Point Arithmetic 2019 [4].

For $\theta \in [0, 2.64697796 * 10^{-23}]$ (encoding [0x00000000, 0x1a000000]), $F_1(\theta)$ is not-a-number (NaN) which is reported by the debugger as -nan(ind). The std::cos function evaluates to 1, and θ^2 is smaller than the smallest positive subnormal and is rounded to 0. This leads to an indeterminate of the form $0/0$, which is flagged as a NaN.

For $\theta \in [2.64697828 * 10^{-23}, 2.44140625 * 10^{-4}]$ (encoding [0x1a00000, 0x39800000]), $F_1(\theta)$ evaluates to 0 because std::cos evaluates to 1 and θ^2 evaluates to a positive number, leading to a number of the form $0/p$ for $p > 0$.

For $\theta \in [2.44140654 * 10^{-4}, 4.88281250 * 10^{-4}]$ (encoding [0x39800001, 0x3a000000]), $F_1(\theta)$ evaluates to numbers in $[1/3, 1]$, which is certainly not what is expected, because $F_1(\theta)$ has a maximum of $1/2$. As θ increases, $F_1(\theta)$ decreases from 1 to $1/3$. A discontinuity appears where $F_1(\theta)$ jumps to $2/3$, after which as θ increases, $F_1(\theta)$ decreases from $2/3$ to $1/2$. Figure 2 shows a graph of $F_1(\theta)$ for $\theta \in [2.44140625 * 10^{-4}, 4.88281250 * 10^{-4}]$ (encoding [0x1a00000, 0x3a000000]).

Figure 2. The graph of $F_1(\theta)$ for $\theta \in [2.44140654 * 10^{-4}, 4.88281250 * 10^{-4}]$ using the floating-point type float. The plot was drawn using Mathematica [5].

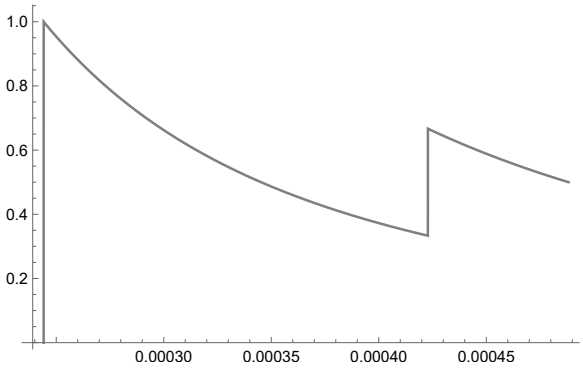
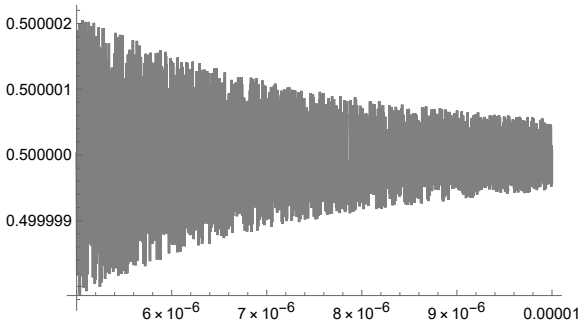


Figure 2 shows the discontinuity where $F_1(\theta)$ jumps from 0 to 1 and the discontinuity where $F_1(\theta)$ jumps from $1/3$ to $2/3$. Regardless, the values of $F_1(\theta)$ for $\theta \in [0, 4.88281250 * 10^{-4}]$ are not accurate.

Mathematica [5] has its own problems with rounding errors near 0. Figure 3 shows the graph of $F_1(\theta)$ for small θ.

Figure 3. The graph of $F_1(\theta)$ for small θ that is computed and plotted using Mathematica [5].



The graph is densely drawn because the rounding errors vary with a very small interval centered at $1/2$, but notice that the function values are all in the interval $[0.499998, 0.500002]$. The näive implementation for $F_1(\theta)$ can be modified to that shown in listing 4 for floating-point type float,

Listing 4. An alternate implementation of $F_1(\theta)$ for floating-point type float. The domain is restricted to $\theta \in [-\pi, \pi]$.

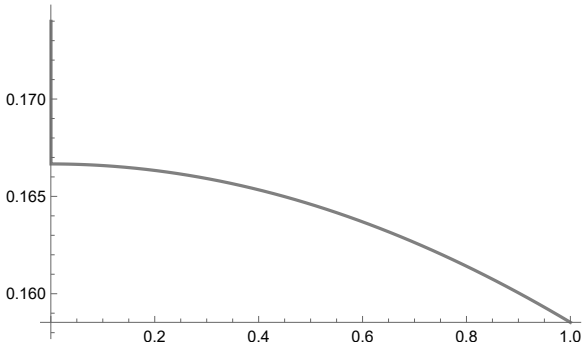
```
float F0(float theta)
{
    return std::fabs(theta) > 4.88281250e−04f ? (1.0f − std::cos(theta)) / (theta * theta) : 0.5f);
}
```

The problem, though, is how to choose the cutoff value for θ when the floating-point type is double. It is also not clear what the maximum error is compared to the theoretical function $(1 - \cos\theta)/\theta^2$. The sawtooth-like behavior continues past ths coded cutoff, although the approximation errors are smaller than before the coded cutoff.

11.3 Robust Computation of F_2

Evaluation of $F_2(\theta)$ also has significant problems. At a large scale, the graph of $F_2(\theta)$ is shown in figure 4.

Figure 4. The graph of $F_2(\theta)$ for $\theta > 0$ drawn at large scale. The plot was drawn using Mathematica [5]. Notice the spike-like behavior near $\theta = 0$, which indicates a floating-point problem. Theoretically, $F_2(\theta) \leq 1/6$ but the spike shows values larger than $1/6$.



The function can be evaluated for all positive and finite float numbers. However, for the sake of analysis, only those numbers $\theta \in [0, 1]$ are considered. The experiment was performed using Microsoft Visual Studio 2022 17.9.6 in the Debug x64 configuration. The encodings are according to the IEEE Standard for Floating-Point Arithmetic 2019 [4].

For $\theta \in [0, 8.88178420 * 10^{-16}]$ (encoding [0x00000000, 0x26800000]), $F_2(\theta)$ is not-a-number (NaN) which is reported by the debugger as -nan(ind). The expression $\theta - \sin\theta$ evaluates to 0, and θ^3 is smaller than the smallest positive subnormal and is rounded to 0. This leads to an indeterminate of the form $0/0$, which is flagged as a NaN.

For $\theta \in [8.88178526 * 10^{-16}, 4.43632947 * 10^{-4}]$ (encoding [0x26800001, 0x39E89768]), $F_2(\theta)$ evaluates to 0 because the expression $\theta - \sin\theta$ evaluates to 0 and θ^3 evaluates to a positive number, leading to a number of the form $0/p$ for $p > 0$.

For $\theta \in [4.43632976 * 10^{-4}, 4.88281250 * 10^{-4}]$ (encoding [0x39E89769, 0x3a000000]), $F_2(\theta)$ evaluates to numbers in $[1/4, 1/3]$, which is certainly not what is expected, because $F_2(\theta)$ has a maximum of $1/6$. As θ increases, $F_2(\theta)$ decreases from $1/3$ to $1/4$.

For $\theta \in [4.88281308 * 10^{-4}, 5.58942498 * 10^{-4}]$ (encoding [0x3a000001, 0x3A1285FF]), $F_2(\theta)$ evaluates to 0.

For $\theta \in [5.58942556 * 10^{-4}, 8.06134543 * 10^{-4}]$ (encoding [0x3A128600, 0x3A5352C6]), $F_2(\theta)$ evaluates to numbers in $[1/9, 1/3]$, which is also not what is expected, because $F_2(\theta)$ has a maximum of $1/6$. As θ increases, $F_2(\theta)$ decreases from $1/3$ to $1/9$. At this time, another discontinuity occurs and the value $F_2(\theta)$ jumps to $2/9$ followed by decreasing values.

Figure 5 shows a graph of $F_2(\theta)$ for $\theta \in (0, 0.00095)$.

Figure 5. The graph of $F_2(\theta)$ for $\theta \in (0, 0.00095)$ using the floating-point type float. The plot was drawn using Mathematica [5].

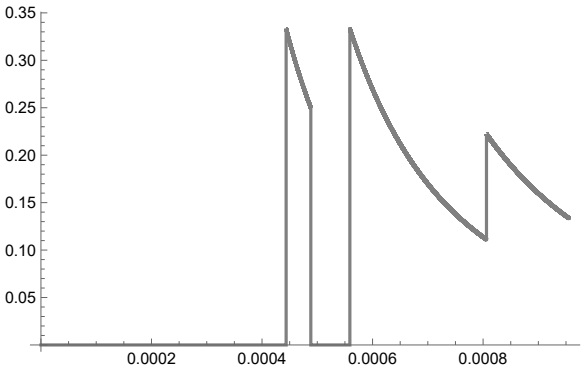
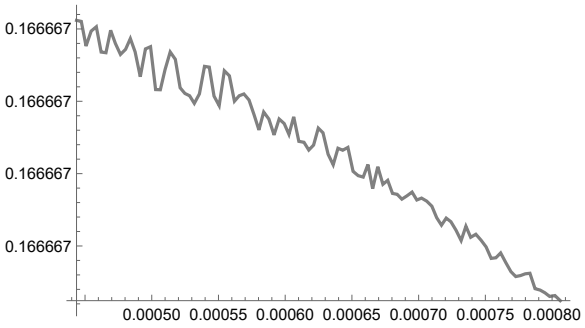


Figure 5 shows four discontinuities, not counting the discontinuity from NaN to 0 for θ nearly 0. The sawtooth patterns continue, but with smaller approximation errors with each sawtooth.

A plot of $(\theta - \sin\theta)/\theta^3$ is shown in figure 6.

Figure 6. A plot of $(\theta - \sin\theta)/\theta^3$ computed and drawn by Mathematica [5].



The evaluations are accurate, but the algorithm used by Mathematica appears not to be a näive implementation using floating-point arithmetic.

11.4 Minimax Polynomial Approximations for Small Angles

A reasonable approach to avoiding the problems with removable singularities at $\theta = 0$ uses an approximation, whether a Taylor polynomial about $\theta = 0$ or a minimax polynomial. Minimax polynomial approximations are preferred instead. A Taylor polynomial provides local error bounds near the point of expansion, so the errors generally become large at inputs far from the point of expansion. A minimax polynomial provides a global error bound that applies to the entire domain of the polynomial. Global error bounds make it

simpler to choose an angle threshold for switching between the minimax polynomial for small θ and using the trigonometric functions in the direct evaluations for large θ.

The Geometric Tools file RotationEstimate.h has minimax approximations for various functions related to rotations. Details on how these were generated are in Approximations to Rotation Matrices and Their Derivatives. The Geometric Tools implementation for $SE(2)$ and $se(2)$ use minimax polynomials of degree 16 for $\theta \in [-\pi, \pi]$. The coefficients for the approximation to $F_0(\theta)$ are listed in the array C_ROTC0_EST_COEFF. The global error bound is listed in the array C_ROTC0_EST_MAX_ERROR and is on the order of 10^{-16}. The coefficients for the approximation to $F_1(\theta)$ are listed in the array C_ROTC1_EST_COEFF. The global error bound is listed in the array C_ROTC1_EST_MAX_ERROR and is on the order of 10^{-16}. The coefficients for the approximation to $F_2(\theta)$ are listed in the array C_ROTC4_EST_COEFF. The global error bound is listed in C_ROTC4_EST_MAX_ERROR and is on the order of 10^{-22}.

The implementations for $F_0(\theta)$, $F_1(\theta)$, and $F_2(\theta)$ are shown in listing 5.

Listing 5. Implementations for $F_0(\theta)$, $F_1(\theta)$, and $F_2(\theta)$. The θ cutoff is chosen sufficiently large to produce robust and accurate estimates. Moreover, the large cutoff avoids numerical problems when θ is small and θ^2 or θ^3 becomes subnormal.

```
template <typename T>
T F0(T theta)
{
    if (std::fabs(theta) >= static_cast<T>(0.0625))
    {
        return std::sin(theta) / theta;
    }
    else
    {
        return static_cast<T>(1);
    }
}

template <typename T>
T F1(T theta)
{
    if (std::fabs(theta) >= static_cast<T>(0.0625))
    {
        return (static_cast<T>(1) - std::cos(theta)) / theta / theta;
    }
    else
    {
        return static_cast<T>(0.5);
    }
}

template <typename T>
T F2(T theta)
{
    if (std::fabs(theta) >= static_cast<T>(0.0625))
    {
        return (theta - std::sin(theta)) / theta / theta / theta;
    }
    else
    {
        return static_cast<T>(1.0 / 6.0);
    }
}
```

References

[1] Ethan Eade. Lie Groups for Computer Vision.
http://ethaneade.com/lie_groups.pdf, 2014.

[2] Dave Eberly. *Robust and Error-Free Geometric Computing*. CRC Press, Taylor & Francis Group LLC, Boca Raton, FL, 2020.

[3] Morris W. Hirsch and Stephen Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, Inc., Orlando, FL, 1 edition, 1974.

[4] IEEE Computer Society and Microprocessor Standards Association.
IEEE 754™-2019 - IEEE Standard for Floating-Point Arithmetic.
https://standards.ieee.org/ieee/754/6210/.

[5] Wolfram Research, Inc. *Mathematica 14.0.0*. Wolfram Research, Inc., Champaign, Illinois, 2024.