UNIVERSITY OF TORONTO
INSTITUTE FOR AEROSPACE STUDIES

Doc NT-2011-PTF002
March 30, 2012

State Estimation for Robotics Using B-Splines

Paul Furgale
University of Toronto
Institute for Aerospace Studies
<paul.furgale@utoronto.ca>

1 Introduction

This note will present derivations useful for adapting B-splines for parameterizing states in robotics estimation problems. In Furgale et al. (2012), we propose moving robotic estimation problems into continuous time. If our goal is to estimate a set of robot states, $\mathbf{x}(t)$, over a time interval, $T = [t_0, t_K]$, we may approximate $\mathbf{x}(t)$ as the weighted sum of a finite set of known temporal basis functions,

$$\boldsymbol{\Phi}(t) := \begin{bmatrix} \boldsymbol{\phi}_1(t) & \dots & \boldsymbol{\phi}_M(t) \end{bmatrix}, \quad \mathbf{x}(t) := \boldsymbol{\Phi}(t)\mathbf{c}. \tag{1}$$

When $\mathbf{x}(t)$ is D-dimensional, each individual basis function, $\boldsymbol{\phi}_j(t)$, is also D-dimensional and the stacked basis matrix, $\boldsymbol{\Phi}(t)$, is $D \times M$. Under the basis function formulation, our goal is to estimate the $M \times 1$ column of coefficients, $\mathbf{c}$.

2 A Matrix Formulation for B-Splines

For our particular problem of state estimation for robotics, we would like the basis functions to satisfy some general properties.

- **Local Support**—we would like the contribution of any single basis function to be local in time. With this property the modification of a single coefficient will have only local effects. This will enable the use of these methods in local batch optimization where only temporally or spatially local parameters are optimized in a single batch.
- **Simple Analytical Derivatives and Integrals**—this is simply an ease-of-use issue. The relationship between sensory inputs and the robot state is often based on derivatives or integrals of the state equations. Having access to simple analytical integrals and derivatives will allow us to evaluate error terms in the estimation equations analytically and avoid expensive numerical schemes.

Under these criteria, B-splines become an attractive choice. A B-spline basis function of order K is a piecewise polynomial function of degree $K - 1$. For simplicity of exposition, we will present our derivations using cubic (fourth-order) B-splines, but the derivations may be easily extended to work for any order. Furthermore, rather than using the recurrence-relation approach to B-splines popularized by de Boor (de Boor, 2002), and others, we will use a matrix formulation that is more amenable to the typical robotics estimation framework. For an introduction to B-splines in general, see Bartels et al. (1987).

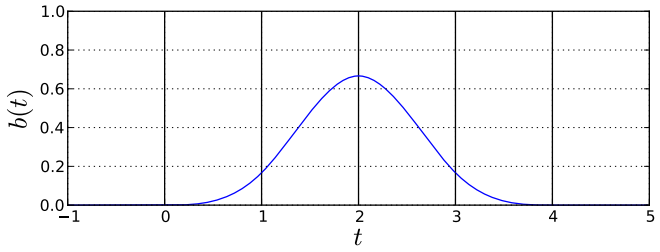

Figure 1: A cubic B-spline basis function is defined as a piecewise cubic polynomial over four segments. The segment transition points are known as knots, shown here as vertical lines. In between each pair of knots, the curve is defined by a cubic polynomial known as the *segment basis polynomial*. By construction, two segment basis polynomials meeting at a knot share the same value, as well as the first and second derivatives. This plot shows a single cubic B-spline basis function on the knot sequence $\{0, 1, 2, 3, 4\}$.

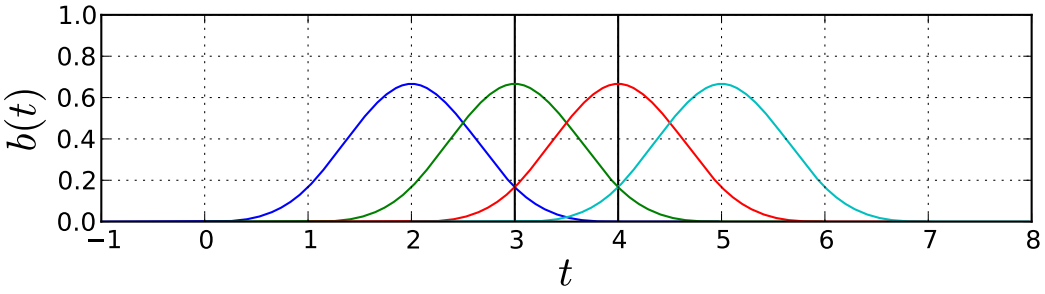

Figure 2: When cubic B-spline basis functions are aligned with a non-decreasing knot sequence, only four basis functions are nonzero for any value of t.

2.1 B-Spline Basis Functions

A cubic B-spline basis function is defined as a piecewise cubic polynomial that is non-zero over four consecutive segments of the real line and zero everywhere else, as shown in Figure 1. The segment transition points, $\{t_i\}$, are known as knots and, in between each pair of knots, the curve is defined by a cubic polynomial known as the *segment basis polynomial*. By construction, two segment basis polynomials meeting at a knot share the same value, as well as the first and second derivatives at that knot. Segment basis polynomial i is defined on the interval $[t_i, t_{i+1})$, and has the form

$$a_i + b_i u_i(t) + c_i u_i(t)^2 + d_i u_i(t)^3, \tag{2}$$

where

$$u_i(t) := \frac{t - t_i}{t_{i+1} - t_i} \tag{3}$$

The coefficients of this polynomial—a_i, b_i, c_i, and d_i—are determined by the spacing of the surrounding knots and a single basis function is defined by 16 coefficients (4 for each segment). We will define $b_i(t)$ to be the basis function whose first nonzero segment starts at knot t_i.

2.2 B-Spline Functions

A cubic B-spline function is a weighted sum of a sequence of B-spline basis functions. When cubic basis functions are aligned with a non-decreasing knot sequence, $\{t_i\}$, only four basis functions are nonzero for any value of t as shown in Figure 2. By convention, we say that the curve is undefined where less than four basis functions are present. Therefore, to have a curve defined over M segments, we need $M+7$ knots and $M+3$ basis functions.

Given a strictly non-decreasing knot sequence, $\{t_i | i = -3 \dots M+3, t_i \leq t_{i+1}\}$, basis functions, $\{b_i(t) | i = -3 \dots M-1\}$, and a set of coefficients, $\{c_i | i = -3 \dots M-1\}$, a B-spline function, $b(t)$, may be written as

$$b(t) := \begin{bmatrix} b_{-3}(t) & \cdots & b_{M-1}(t) \end{bmatrix} \begin{bmatrix} c_{-3} \\ \vdots \\ c_{M-1} \end{bmatrix}. \tag{4}$$

This function is well defined in the interval $[t_0, t_M)$. Because only four basis functions are nonzero for any value of t, for $t_i \leq t < t_{i+1}$, (4) reduces to

$$b(t) := \begin{bmatrix} b_{i-3}(t) & b_{i-2}(t) & b_{i-1}(t) & b_i(t) \end{bmatrix} \underbrace{\begin{bmatrix} c_{i-3} \\ c_{i-2} \\ c_{i-1} \\ c_i \end{bmatrix}}_{\mathbf{c}_i}, \tag{5}$$

which may be written as

$$b(t) = \mathbf{u}_i(t)^T \mathbf{B}_i \mathbf{c}_i, \tag{6}$$

where $\mathbf{B}_i$ is a 4×4 matrix of basis polynomial coefficients, and

$$\mathbf{u}_i(t) := \begin{bmatrix} 1 \\ u_i \\ u_i^2 \\ u_i^3 \end{bmatrix} \tag{7}$$

A sample curve together with the basis functions and spline coefficients is shown in Figure 3. The entries of the matirx $\mathbf{B}_i$ are dependent on the knot spacing. The general formula for $\mathbf{B}_i$ is given by Qin (2000) for B-splines of any order. For cubic B-splines with uniform knot spacing, all $\mathbf{B}_i$ matrices are equal:

$$\mathbf{B}_i = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \tag{8}$$

2.3 Derivatives of a B-Spline Function

In matrix form, analytical derivatives of the cubic B-spline function become easy to compute. In fact, we may define a matrix, $\mathbf{D}$, that applies the derivative through matrix multiplication. For a given t on the interval $[t_i, t_{i+1})$,

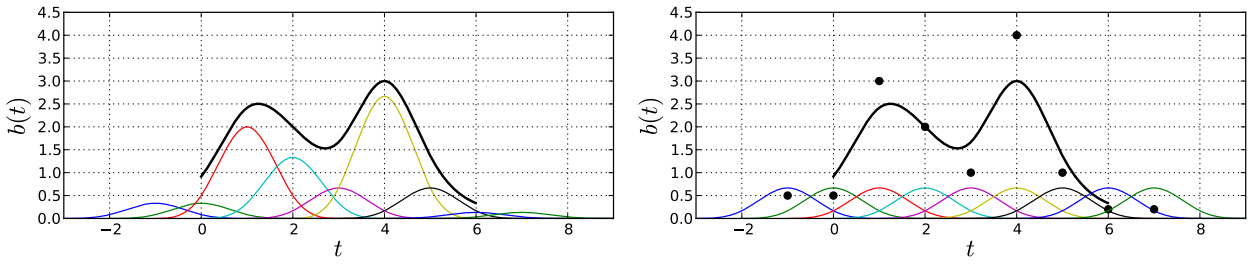

Figure 3: The curve defined by a B-spline (shown in bold black) is the weighted sum of a series of B-spline basis functions. The plot on the left shows the scaled basis functions. The plot on the right shows the unscaled basis functions and the spline coefficients plotted as black dots. Note that the B-spline curve does not generally interpolate the coefficients.

the derivatives are

$$\frac{db(t)}{dt} = \frac{1}{t_{i+1} - t_i} \begin{bmatrix} 0 & 1 & 2u_i & 3u_i^2 \end{bmatrix} \mathbf{B}_i \mathbf{c}_i = \mathbf{u}_i(t)^T \mathbf{D}_i \mathbf{B}_i \mathbf{c}_i, \tag{9}$$

$$\frac{d^2b(t)}{dt^2} = \left(\frac{1}{t_{i+1} - t_i} \right)^2 \begin{bmatrix} 0 & 0 & 2 & 6u_i \end{bmatrix} \mathbf{B}_i \mathbf{c}_i = \mathbf{u}_i(t)^T \mathbf{D}_i \mathbf{D}_i \mathbf{B}_i \mathbf{c}_i, \tag{10}$$

$$\frac{d^3b(t)}{dt^3} = \left(\frac{1}{t_{i+1} - t_i} \right)^3 \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix} \mathbf{B}_i \mathbf{c}_i = \mathbf{u}_i(t)^T \mathbf{D}_i \mathbf{D}_i \mathbf{D}_i \mathbf{B}_i \mathbf{c}_i, \tag{11}$$

$$\frac{d^4b(t)}{dt^4} = 0, \tag{12}$$

where we have defined the matrix

$$\mathbf{D}_i := \frac{1}{t_{i+1} - t_i} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{13}$$

2.4 Definite Integrals of a B-Spline Function

The definite integral of a B-spline function is similarly easy to compute. To evaluate a definite integral, we may compute the integral of each segment separately. Let us consider the integral within a single segment from $t = s_1$ to $t = s_2$ where $t_i \leq s_1 < s_2 < t_{i+1}$,

$$\int_{s_1}^{s_2} b(t)\, dt = \int_{s_1}^{s_2} \mathbf{u}_i(t)^T \mathbf{B}_i \mathbf{c}_i\, dt. \tag{14}$$

We may simplify the integral with a variable substitution,

$$v = \frac{t - t_i}{t_{i+1} - t_i}, \quad v_1 = \frac{s_1 - t_i}{t_{i+1} - t_i}, \quad v_2 = \frac{s_2 - t_i}{t_{i+1} - t_i}, \quad dt = (t_{i+1} - t_i)\, dv, \tag{15}$$

which gives us

$$\int_{s_1}^{s_2} b(t)\, dt = (t_{i+1} - t_i) \int_{v_1}^{v_2} \underbrace{\begin{bmatrix} 1 & v & v^2 & v^3 \end{bmatrix} \mathbf{B}_i \mathbf{c}_i}_{=: v_i(v)}\, dv\,. \tag{16}$$

The anti-derivative of $v_i(v)$ is

$$B_i(v) := \begin{bmatrix} v & \frac{v^2}{2} & \frac{v^3}{3} & \frac{v^4}{4} \end{bmatrix} \mathbf{B}_i \mathbf{c}_i, \tag{17}$$

and the integral on this interval is

$$\int_{s_1}^{s_2} v_i(t)\, dt = (t_{i+1} - t_i) \int_{v_1}^{v_2} v_i(v)\, dv = (t_{i+1} - t_i) \left(B_i(v_2) - B_i(v_1) \right). \tag{18}$$

When s_1 is equal to t_i, $v_1 = 0$ and hence $B(v_1) = 0$. In the general case, integrating the B-spline function with arbitrary limits a and b where $t_i \leq a < t_{i+1} < \cdots < t_j \leq b < t_{j+1}$, breaks down into the following form:

$$\int_a^b b(t)\, dt = \underbrace{-(t_{i+1} - t_i) B_i \left(\frac{a - t_i}{t_{i+1} - t_i} \right)}_{\text{LHS remainder}} + \underbrace{\sum_{s=i}^{j-1} (t_{s+1} - t_s) B_s(1)}_{\text{full segments}} + \underbrace{(t_{j+1} - t_j) B_j \left(\frac{b - t_j}{t_{j+1} - t_j} \right)}_{\text{RHS remainder}} \tag{19}$$

2.5 Multidimensional B-Spline Functions

For many problems in robotics, the state is multidimensional at each time instant. Consequently, we must lift these equations into multiple dimensions. To achieve this, we first arrange our coefficients so that temporally consecutive parameters are adjacent to each other. In this cace, the full column of coefficients, $\mathbf{c}$, is assembled as,

$$\mathbf{c} := \begin{bmatrix} \mathbf{d}_{-3} \\ \mathbf{d}_{-2} \\ \vdots \\ \mathbf{d}_{M-2} \\ \mathbf{d}_{M-1} \end{bmatrix} \tag{20}$$

where each $\mathbf{d}_i$ is a $D \times 1$ coefficient column associated with basis function i,

$$\mathbf{d}_i =: \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,D} \end{bmatrix}. \tag{21}$$

For a particular t with $t_i \leq t < t_{i+1}$, we may define the columns of the B-spline basis matrix to be

$$\mathbf{B}_i =: \begin{bmatrix} \mathbf{b}_{i1} & \mathbf{b}_{i2} & \mathbf{b}_{i3} & \mathbf{b}_{i4} \end{bmatrix}, \tag{22}$$

which allows us to write the general D-dimensional version of (6) as

$$\mathbf{b}(t) := \underbrace{\begin{bmatrix} \mathbf{u}(t)^T \mathbf{b}_{i1} \mathbf{1} & \mathbf{u}(t)^T \mathbf{b}_{i2} \mathbf{1} & \mathbf{u}(t)^T \mathbf{b}_{i3} \mathbf{1} & \mathbf{u}(t)^T \mathbf{b}_{i4} \mathbf{1} \end{bmatrix}}_{=: \mathbf{\Phi}_i(t)} \mathbf{c}_i, \tag{23}$$

where $\mathbf{1}$ is the $D \times D$ identity matrix and

$$\mathbf{c}_i := \begin{bmatrix} \mathbf{d}_{i-3} \\ \mathbf{d}_{i-2} \\ \mathbf{d}_{i-1} \\ \mathbf{d}_i \end{bmatrix}. \tag{24}$$

Equation (23) may be rewritten in blocks. Defining

$$\mathbf{U}(t) := \begin{bmatrix} \mathbf{u}(t) & & \\ & \ddots & \\ & & \mathbf{u}(t) \end{bmatrix}, \quad \mathbf{B}_{ij} := \begin{bmatrix} \mathbf{b}_{ij} & & \\ & \ddots & \\ & & \mathbf{b}_{ij} \end{bmatrix}, \tag{25}$$

where both $\mathbf{U}(t)$ and $\mathbf{B}_{ij}$ are $4D \times D$ matrices, (23) becomes

$$\begin{aligned} \mathbf{b}(t) &= \mathbf{\Phi}_i(t)\mathbf{c}_i && \text{(26a)} \\ &= \begin{bmatrix} \mathbf{U}(t)^T\mathbf{B}_{i1} & \mathbf{U}(t)^T\mathbf{B}_{i2} & \mathbf{U}(t)^T\mathbf{B}_{i3} & \mathbf{U}(t)^T\mathbf{B}_{i4} \end{bmatrix} \mathbf{c}_i && \text{(26b)} \\ &= \mathbf{U}(t)^T \underbrace{\begin{bmatrix} \mathbf{B}_{i1} & \mathbf{B}_{i2} & \mathbf{B}_{i3} & \mathbf{B}_{i4} \end{bmatrix}}_{=:\mathbf{M}_i} \mathbf{c}_i, && \text{(26c)} \end{aligned}$$

where we have defined $\mathbf{M}_i$, the multidimensional block version of the B-spline basis matrix $\mathbf{B}_i$.

2.6 B-Spline Function Quadratic Integrals

In this section we examine quadratic forms which arise in motion-constraint terms, such as the minimum acceleration model used in Furgale et al. (2012). The term required in that paper was quadratic in the second time derivative of the basis functions, $\ddot{\mathbf{\Phi}}(t)$. However, in Section 2.3 we showed that the derivatives of $\mathbf{\Phi}(t)$ may be expressed through multiplication by a constant matrix, $\mathbf{D}_i$. Hence, it is sufficient to derive the equations for the most general form of quadratic integral for a B-spline of dimension D,

$$\int_{t_0}^{t_K} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t) dt, \tag{27}$$

where $\mathbf{W}$ is some $D \times D$ weighting matrix

$$\mathbf{W} =: \begin{bmatrix} w_{11} & \cdots & w_{1D} \\ \vdots & \ddots & \vdots \\ w_{D1} & \cdots & w_{DD} \end{bmatrix} \tag{28}$$

Just as in Section 2.4, the integral in (27) may be split up into a sum of integrals over the segments:

$$\int_{t_0}^{t_K} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t) dt = \sum_{i=0}^{K-1} \int_{t_i}^{t_{i+1}} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t)\, dt \tag{29}$$

For a single segment, substituting (26) into (29) gives us

$$\int_{t_i}^{t_{i+1}} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t)\, dt \;= \int_{t_i}^{t_{i+1}} \mathbf{M}_i^T \mathbf{U}(t) \mathbf{W} \mathbf{U}(t)^T \mathbf{M}_i\, dt = \mathbf{M}_i^T \int_{t_i}^{t_{i+1}} \mathbf{U}(t) \mathbf{W} \mathbf{U}(t)^T\, dt\, \mathbf{M}_i \tag{30}$$

The integral on the right-hand side may be expanded to

$$\int_{t_i}^{t_{i+1}} \mathbf{U}(t) \mathbf{W} \mathbf{U}(t)^T\, dt = \int_{t_i}^{t_{i+1}} \begin{bmatrix} w_{11}\mathbf{u}(t)\mathbf{u}(t)^T & \cdots & w_{1D}\mathbf{u}(t)\mathbf{u}(t)^T \\ \vdots & \ddots & \vdots \\ w_{D1}\mathbf{u}(t)\mathbf{u}(t)^T & \cdots & w_{DD}\mathbf{u}(t)\mathbf{u}(t)^T \end{bmatrix} dt, \tag{31}$$

which shows that the final subproblem we must solve is

$$\int_{t_i}^{t_{i+1}} \mathbf{u}(t)\mathbf{u}(t)^T \, dt \,. \tag{32}$$

Applying the same change of variables we used in Section 2.4, (32) becomes

$$(t_{i+1} - t_i) \int_0^1 \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} dt = (t_{i+1} - t_i) \int_0^1 \begin{bmatrix} 1 & u & u^2 & u^3 \\ u & u^2 & u^3 & u^4 \\ u^2 & u^3 & u^4 & u^5 \\ u^3 & u^4 & u^5 & u^6 \end{bmatrix} dt, \tag{33}$$

with the solution

$$\mathbf{V}_i := (t_{i+1} - t_i) \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}, \tag{34}$$

which we have designated $\mathbf{V}_i$. Putting it all together, the solution to the initial problem is

$$\int_{t_0}^{t_K} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t) \, dt = \sum_{i=0}^{K-1} \mathbf{M}_i^T \begin{bmatrix} w_{11}\mathbf{V}_i & \cdots & w_{1D}\mathbf{V}_i \\ \vdots & \ddots & \vdots \\ w_{D1}\mathbf{V}_i & \cdots & w_{DD}\mathbf{V}_i \end{bmatrix} \mathbf{M}_i \tag{35}$$

3 A Continuous-Time Representation for Vehicles in Three-Dimensional Space—A Privileged-Frame Approach

If we assume a privileged navigation frame, $\underrightarrow{\mathcal{F}}_n$, we can define a spline that encodes the parameters of the time-varying transformation, $\boldsymbol{T}_{in}(t)$, between the inertial frame and the body frame—a frame attached to the robot, $\underrightarrow{\mathcal{F}}_i(t)$. The most straightforward way to do this is to choose a rotation parameterization and have three dimensions of the spline encode rotation and the other three encode translation. The drawback to this method is the same faced any time one must choose a rotation parameterization—rotation parameterizations are subject to singularities. However, for a particular problem, it is often possible to choose a parameterization with singularities in improbable or impossible positions with respect to the domain. Hence, we will proceed with a general exposition that covers all three-parameter rotation representations.

Our transformation matrix is constructed as

$$\boldsymbol{T}_{in}(t) := \begin{bmatrix} \mathbf{C}\left(\boldsymbol{\theta}(t)\right) & \mathbf{t}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{36}$$

where $\boldsymbol{\theta}(t)$ is a continuous function representing our rotation parameters, $\mathbf{C}(\cdot)$ is a function that builds a rotation matrix from our parameters, and $\mathbf{t}(t)$ is a continuous function representing our translation parameters. In the notation of Section **??**, these functions may be written as

$$\boldsymbol{\theta} := \mathbf{\Phi}(t)\mathbf{c}_\theta, \qquad \mathbf{t}(t) := \mathbf{\Phi}(t)\mathbf{c}_t \,, \tag{37}$$

and in the notation of Section 2.5, on the interval $t_i < t < t_{i+1}$, they may be written as

$$\boldsymbol{\theta} := \mathbf{U}_i(t)\mathbf{B}_i\mathbf{c}_{\theta_i}, \qquad \mathbf{t}(t) := \mathbf{U}_i(t)\mathbf{B}_i\mathbf{c}_{t_i} \,. \tag{38}$$

For the representation to be useful for state estimation, we must show two things:

- what the relationship is between (36) and other useful quantities, namely the angular velocity, $\boldsymbol{\omega}(t)$, linear velocity, $\mathbf{v}(t)$, and linear acceleration of the platform $\mathbf{a}(t)$, and
- how to linearize equations involving (36) with respect to small changes in $\mathbf{c} := \begin{bmatrix}\mathbf{c}_t^T & \mathbf{c}_\theta^T\end{bmatrix}^T$.

The first point is simple. Because we are encoding the transformation matrix that takes points from the inertial frame, to the body frame, the linear velocity and acceleration are

$$\mathbf{v}(t) = \dot{\mathbf{t}}(t)\,, \quad \mathbf{a}(t) = \ddot{\mathbf{t}}(t)\,. \tag{39}$$

Linearizing these terms with respect to small changes in $\mathbf{c}_v$ is straightforward. For a given rotation parameterization, the relationship to angular velocity is of the form

$$\boldsymbol{\omega}(t) = \mathbf{S}\left(\boldsymbol{\theta}(t)\right)\dot{\boldsymbol{\theta}}(t)\,, \tag{40}$$

where $\mathbf{S}(\cdot)$ for many common rotation parameterizations may be found in Table 2.3 on page 31 of Hughes (1986). The linearization of this term with respect to small changes in $\mathbf{c}_\theta$ must be computed for each choice of parameters. We will present an example in the next section.

The only remaining question is how to linearize expressions involving $\mathbf{C}\left(\boldsymbol{\theta}(t)\right)$ with respect to small changes in $\mathbf{c}_\theta$. Considering an interval $t_i < t < t_{i+1}$, we define

$$\mathbf{c}_{\theta_i} = \bar{\mathbf{c}}_{\theta_i} + \delta\mathbf{c}_{\theta_i}\,, \tag{41}$$

where $\bar{\mathbf{c}}_{\theta_i}$ is the nominal value of our spline coefficients and $\delta\mathbf{c}_{\theta_i}$ is a perturbation. This gives us

$$\boldsymbol{\theta}(t) := \underbrace{\mathbf{U}_i(t)\mathbf{B}_i\bar{\mathbf{c}}_{\theta_i}}_{\bar{\boldsymbol{\theta}}(t)} + \underbrace{\mathbf{U}_i(t)\mathbf{B}_i\delta\mathbf{c}_{\theta_i}}_{\delta\boldsymbol{\theta}(t)}\,. \tag{42}$$

These definitions may be substituted into the derivations in Barfoot et al. (2011) such that expressions involving rotation matrices may be linearized using

$$\mathbf{C}\left(\boldsymbol{\theta}(t)\right) = \left(\mathbf{1} - \left(\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\times\right)\mathbf{C}\left(\bar{\boldsymbol{\theta}}(t)\right)\,, \tag{43}$$

where $\mathbf{1}$ is the identity matrix, $(\cdot)^\times$ is the skew-symmetric operator that may be used to implement the cross product,

$$\begin{bmatrix}x\\y\\z\end{bmatrix}^\times := \begin{bmatrix}0 & -z & y\\ z & 0 & -x\\ -y & x & 0\end{bmatrix}\,, \tag{44}$$

and, again, the form of $\mathbf{S}(\cdot)$ is available in Hughes (1986). Using (43) we may write the equations that describe

how small changes in our coefficients become changes in an arbitrary homogeneous vector $\boldsymbol{v} := \begin{bmatrix}\mathbf{v}^T & s\end{bmatrix}^T$:

$$\boldsymbol{T}(t)\boldsymbol{v} = \begin{bmatrix}\left(\mathbf{1} - \left(\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\times\right)\mathbf{C}\left(\overline{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t) + \delta\mathbf{t}(t) \\ \mathbf{0}^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ s\end{bmatrix} \tag{45a}$$

$$= \begin{bmatrix}\mathbf{1} - \left(\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\times & \delta\mathbf{t}(t) + \left(\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\times\bar{\mathbf{t}}(t) \\ \mathbf{0}^T & \mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{C}\left(\overline{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t)\\ \mathbf{0}^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ s\end{bmatrix} \tag{45b}$$

$$= \begin{bmatrix}\mathbf{1} - \left(\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\times & \delta\mathbf{t}(t) - \bar{\mathbf{t}}^\times(t)\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t) \\ \mathbf{0}^T & \mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{C}\left(\overline{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t)\\ \mathbf{0}^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ s\end{bmatrix} \tag{45c}$$

$$= \left(\mathbf{1} - \underbrace{\begin{bmatrix}\left(\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\times & \bar{\mathbf{t}}^\times(t)\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t) - \delta\mathbf{t}(t) \\ \mathbf{0}^T & \mathbf{0}\end{bmatrix}}_{=:\delta\mathbf{d}^\boxplus(t)}\right)\begin{bmatrix}\mathbf{C}\left(\overline{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t)\\ \mathbf{0}^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ s\end{bmatrix}, \tag{45d}$$

where we have defined

$$\delta\mathbf{d}(t) := \begin{bmatrix}\mathbf{1} & -\bar{\mathbf{t}}^\times(t)\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\\ \mathbf{0} & \mathbf{S}\left(\overline{\boldsymbol{\theta}}(t)\right)\end{bmatrix}\begin{bmatrix}\delta\mathbf{t}(t)\\ \delta\boldsymbol{\theta}(t)\end{bmatrix}, \tag{46}$$

and have used the $(\cdot)^\boxplus$ operator defined as

$$\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}^\boxplus := \begin{bmatrix}\mathbf{b}^\times & -\mathbf{a}\\ \mathbf{0}^T & 0\end{bmatrix}, \tag{47}$$

where $\mathbf{a}$ and $\mathbf{b}$ are both 3×1 columns. Relating this back to the perturbations of our spline coefficients, at a particular time, $\mathbf{t}_k$, in the interval $[t_i, t_{i+1})$ this gives us

$$\delta\mathbf{d}(t_k) := \underbrace{\begin{bmatrix}-\mathbf{1} & \bar{\mathbf{t}}^\times(t_k)\mathbf{S}\left(\overline{\boldsymbol{\theta}}(t_k)\right)\\ \mathbf{0} & \mathbf{S}\left(\overline{\boldsymbol{\theta}}(t_k)\right)\end{bmatrix}\begin{bmatrix}\mathbf{U}_i(t_k)\mathbf{B}_i & \mathbf{0}\\ \mathbf{0} & \mathbf{U}_i(t_k)\mathbf{B}_i\end{bmatrix}}_{=:\mathbf{D}_i(t_k)}\underbrace{\begin{bmatrix}\delta\mathbf{c}_{t_i}\\ \delta\mathbf{c}_{\theta_i}\end{bmatrix}}_{=:\delta\mathbf{c}_i}. \tag{48}$$

This allows us to use some nice algebra associated with linearized transformation matrices. Defining the $(\cdot)^\boxminus$ operator as

$$\boldsymbol{v}^\boxminus = \begin{bmatrix}\mathbf{v}\\ s\end{bmatrix}^\boxminus := \begin{bmatrix}s\mathbf{1} & \mathbf{v}^\times\\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}, \tag{49}$$

where $\boldsymbol{v}$ is a vector expressed as a 4×1 column of homogeneous coordinates, we may use the identity,

$$-\delta\mathbf{d}^\boxplus\boldsymbol{v} = \boldsymbol{v}^\boxminus\delta\mathbf{d} \tag{50}$$

to simplify expressions involving linearized transformation matrices.

4 Quaternion Splines

In this section, we outline the approach for unit quaternion B-splines presented in Kim et al. (1995), derive the Jacobians required for state estimation, and present the equations for angular velocity and acceleration. Fundamental

to the approach in Kim et al. (1995) is the use of the matrix exponential and matrix logarithm operations unit length quaternions, $\left\{\mathbf{q}|\mathbf{q}\in\mathbb{R}^4, \mathbf{q}^T\mathbf{q}=1\right\}$. We will rely on the quaternion algebra presented in Barfoot et al. (2011). In this notation, we define the components of the quaternion to be

$$\mathbf{q} =: \begin{bmatrix}\boldsymbol{\epsilon}\\ \eta\end{bmatrix}^T, \tag{51}$$

where $\boldsymbol{\epsilon}$ is 3×1 and η is a scalar. The quaternion *left-hand compound* operator, $+$, and the *right-hand compound* operator, $\oplus$, will be defined as

$$\mathbf{q}^+ := \begin{bmatrix}\eta\mathbf{1}-\boldsymbol{\epsilon}^\times & \boldsymbol{\epsilon}\\ -\boldsymbol{\epsilon}^T & \eta\end{bmatrix} \text{ and } \mathbf{q}^\oplus := \begin{bmatrix}\eta\mathbf{1}+\boldsymbol{\epsilon}^\times & \boldsymbol{\epsilon}\\ -\boldsymbol{\epsilon}^T & \eta\end{bmatrix}. \tag{52}$$

Under these definitions, the *multiplication* of quaternions, $\mathbf{q}$ and $\mathbf{r}$, which is typically written as $\mathbf{q}\otimes\mathbf{r}$ (Shuster, 1993), may be written equivalently as either

$$\mathbf{q}^+\mathbf{r} \text{ or } \mathbf{r}^\oplus\mathbf{q}, \tag{53}$$

which are both products of a 4×4 matrix with a 4×1 column. The *conjugate* operator for quaternions, -1, will be defined by

$$\mathbf{q}^{-1} := \begin{bmatrix}-\boldsymbol{\epsilon}\\ \eta\end{bmatrix}. \tag{54}$$

The set of quaternions forms a *non-commutative group* under both the $+$ and $\oplus$ operations (Shuster, 1993). The *identity element* of this group, $\boldsymbol{\iota} := \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$, is such that

$$\boldsymbol{\iota}^+ = \boldsymbol{\iota}^\oplus = \mathbf{1}, \tag{55}$$

where $\mathbf{1}$ is the 4×4 identity matrix. None of the preceding definitions require the quaternions to be of unit length. However, given two unit-length quaternions, $\mathbf{q}$ and $\mathbf{r}$,

$$\mathbf{q}^T\mathbf{q}=1, \quad \mathbf{r}^T\mathbf{r}=1, \tag{56}$$

both the $+$ and $\oplus$ operators preserve the unit length:

$$\left(\mathbf{q}^+\mathbf{r}\right)^T\left(\mathbf{q}^+\mathbf{r}\right)=1, \quad \left(\mathbf{q}^\oplus\mathbf{r}\right)^T\left(\mathbf{q}^\oplus\mathbf{r}\right)=1 \tag{57}$$

For unit-length quaternions, we define the $\log$ and $\exp$ functions to be

$$\mathbf{q} := \exp(\boldsymbol{\phi}) = \begin{bmatrix}\sin\frac{\phi}{2}\mathbf{a}\\ \cos\frac{\phi}{2}\end{bmatrix}, \quad \boldsymbol{\phi} := \log(\mathbf{q}) = \frac{2\arccos\eta}{\sqrt{1-\eta^2}}\boldsymbol{\epsilon}, \tag{58}$$

where $\boldsymbol{\phi}$ is 3×1, $\phi := \sqrt{\boldsymbol{\phi}^T\boldsymbol{\phi}}$, and $\mathbf{a} := \boldsymbol{\phi}/\phi$. Note that

$$\log(\mathbf{q}^{-1}) = -\boldsymbol{\phi}, \qquad \exp(-\boldsymbol{\phi}) = \mathbf{q}^{-1}, \tag{59}$$

and the rotation matrix, $\mathbf{C}$, associated with $\mathbf{q}$ may be built using

$$\mathbf{q}^+\mathbf{q}^{-1^\oplus} = \begin{bmatrix}\mathbf{C} & \mathbf{0}\\ \mathbf{0}^T & 1\end{bmatrix}. \tag{60}$$

4.1 Quaternion Curves

Kim et al. (1995) define a B-spline quaternion curve based on *cumulative* B-spline basis functions. Cumulative basis functions represent an alternative but equivalent method of evaluating a B-spline function. The example in Section 2.2 showed that a B-spline function at time t could be written as

$$b(t) = \begin{bmatrix} b_{i-3}(t) & b_{i-2}(t) & b_{i-1}(t) & b_i(t) \end{bmatrix} \begin{bmatrix} c_{i-3} \\ c_{i-2} \\ c_{i-1} \\ c_i \end{bmatrix}. \tag{61}$$

This same function may be written in cumulative form as

$$b(t) = \begin{bmatrix} \beta_{i-3}(t) & \beta_{i-2}(t) & \beta_{i-1}(t) & \beta_i(t) \end{bmatrix} \begin{bmatrix} c_{i-3} \\ c_{i-2} - c_{i-3} \\ c_{i-1} - c_{i-2} \\ c_i - c_{i-1} \end{bmatrix}, \tag{62}$$

where the cumulative basis functions, $\beta_i(t)$, are defined as

$$\begin{bmatrix} \beta_{i-3}(t) \\ \beta_{i-2}(t) \\ \beta_{i-1}(t) \\ \beta_i(t) \end{bmatrix} := \begin{bmatrix} b_{i-3}(t) + b_{i-2}(t) + b_{i-1}(t) + b_i(t) \\ b_{i-2}(t) + b_{i-1}(t) + b_i(t) \\ b_{i-1}(t) + b_i(t) \\ b_i(t) \end{bmatrix} = \begin{bmatrix} 1 \\ b_{i-2}(t) + b_{i-1}(t) + b_i(t) \\ b_{i-1}(t) + b_i(t) \\ b_i(t) \end{bmatrix}. \tag{63}$$

The B-spline quaternion curve in Kim et al. (1995) is built from (62) by analogy using the Lie algebra associated with quaternions. Rather than scalar coefficients, c_i, the curve is defined by a set of quaternion-valued coefficients, $\mathbf{q}_i$. The quaternion equivalent of (62) becomes

$$\mathbf{q}(t) := \mathbf{q}_{i-3}^{+} \mathbf{r}_{i-2}(t)^{+} \mathbf{r}_{i-1}(t)^{+} \mathbf{r}_i(t), \tag{64}$$

where

$$\mathbf{r}_i(t) := \exp\left(\beta_i(t) \boldsymbol{\varphi}_i\right), \tag{65}$$

and

$$\boldsymbol{\varphi}_i := \log\left(\mathbf{q}_{i-1}^{-1\,+} \mathbf{q}_i\right). \tag{66}$$

Defining a curve to represent SO(3) in this way has two properties that make it desirable: (a) it has no singularities, and (b) it satisfies the property of *bi-invariance* as described in Park and Ravani (1997). To use these curves for estimation, we require several pieces not derived in Kim et al. (1995), Jacobians and the equations for angular velocity and angular acceleration.

4.2 Jacobians of Quaternion Curves

The Levenberg Marquardt method is the predominant method of estimation in robotics. To implement this method analytically, we must have access to an expression for the Jacobian (with respect to small changes in the state variables) for any nonlinear function that appears in our error terms. In this case, the state variables are the unit-length quaternion-valued coefficients. Unit-length quaternions use four parameters for three degrees of freedom,

but rather than performing constrained estimation, it is usual to use a *minimal perturbation* when using quaternions in unconstrained estimation. This is often written as

$$\mathbf{q} \approx (\iota + \mathbf{V}\delta\boldsymbol{\phi})^{+}\,\overline{\mathbf{q}}, \tag{67}$$

where

$$\mathbf{V} := \frac{1}{2}\begin{bmatrix}\mathbf{1}\\ \mathbf{0}^T\end{bmatrix} \tag{68}$$

is a 4×3 matrix, $\overline{\mathbf{q}}$ is our current guess, and $\delta\boldsymbol{\phi}$ is a minimal, 3×1, perturbation. When an iteration of Levenberg-Marquardt produces an answer for $\delta\boldsymbol{\phi}$, the update is applied as

$$\overline{\mathbf{q}} \leftarrow \exp(\delta\boldsymbol{\phi})^{+}\overline{\mathbf{q}}. \tag{69}$$

This update is *constraint sensitive* in that the updated $\overline{\mathbf{q}}$ is still of unit length. We would like to use the same method to perform unconstrained optimization using quaternion curves. The question is, what is the Jacobian of (64) with respect to small changes in the individual quaternion-valued coefficients?

To derive this, we require some intermediate results. First, we define the Jacobian of the $\log(\cdot)$ function with respect to perturbations of the form in (67),

$$\log\left((\iota + \mathbf{V}\delta\boldsymbol{\phi})^{+}\,\overline{\mathbf{q}}\right) \approx \log(\overline{\mathbf{q}}) + \mathbf{L}(\overline{\mathbf{q}})\delta\boldsymbol{\phi}, \tag{70}$$

where

$$\mathbf{L}(\mathbf{q}) := \mathbf{1} + \frac{1}{2}\boldsymbol{\phi}^{\times} + \left(1 - \frac{1}{2\tan\frac{1}{2}\phi}\right)\mathbf{a}^{\times}\mathbf{a}^{\times}, \tag{71}$$

and we have used $\boldsymbol{\phi} := \log(\mathbf{q})$, $\phi := \sqrt{\boldsymbol{\phi}^T\boldsymbol{\phi}}$, and $\mathbf{a} := \boldsymbol{\phi}/\phi$. Next, we need the Jacobian of the $\exp(\cdot)$ function with respect to perturbations in θ. This can be written as

$$\exp\left(\overline{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}\right) \approx \left(\iota + \mathbf{VS}\left(\overline{\boldsymbol{\theta}}\right)\delta\boldsymbol{\theta}\right)^{+}\exp\left(\overline{\boldsymbol{\theta}}\right) = \exp\left(\overline{\boldsymbol{\theta}}\right) + \exp\left(\overline{\boldsymbol{\theta}}\right)^{\oplus}\mathbf{VS}\left(\overline{\boldsymbol{\theta}}\right)\delta\boldsymbol{\theta}, \tag{72}$$

where

$$\mathbf{S}(\boldsymbol{\theta}) := \mathbf{1} - \frac{2}{\theta}\sin^2\tfrac{1}{2}\theta\,\mathbf{a}^{\times} + \frac{1}{\theta}(\theta - \sin\theta)\mathbf{a}^{\times}\mathbf{a}^{\times}. \tag{73}$$

We note, without proof, that $\mathbf{L}(\mathbf{q}) = \mathbf{S}(\log(\mathbf{q}))^{-1}$. Next, we see how perterbations of the form (67) become perturbations in $\boldsymbol{\varphi}_i$ from (66):

$$\begin{aligned}
\boldsymbol{\varphi}_i &= \log\left(\mathbf{q}_{i-1}^{-1\,+}\mathbf{q}_i\right) && \text{(74a)}\\
&= \log\left(\overline{\mathbf{q}}_{i-1}^{-1\,+}\left(\boldsymbol{\iota} - \mathbf{V}\delta\boldsymbol{\phi}_{i-1}\right)^{+}\left(\boldsymbol{\iota} + \mathbf{V}\delta\boldsymbol{\phi}_i\right)^{+}\overline{\mathbf{q}}_i\right) && \text{(74b)}\\
&\approx \log\left(\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i - \overline{\mathbf{q}}_{i-1}^{-1\,+}(\mathbf{V}\delta\boldsymbol{\phi}_{i-1})^{+}\overline{\mathbf{q}}_i + \overline{\mathbf{q}}_{i-1}^{-1\,+}(\mathbf{V}\delta\boldsymbol{\phi}_i)^{+}\overline{\mathbf{q}}_i\right) && \text{(74c)}\\
&= \log\left(\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i - (\mathbf{V}\overline{\mathbf{C}}_{i-1}^{T}\delta\boldsymbol{\phi}_{i-1})^{+}\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i + (\mathbf{V}\overline{\mathbf{C}}_{i-1}^{T}\delta\boldsymbol{\phi}_i)^{+}\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i\right) && \text{(74d)}\\
&= \log\left(\left(\iota + \left(\mathbf{V}\overline{\mathbf{C}}_{i-1}^{T}\delta\boldsymbol{\phi}_i - \mathbf{V}\mathbf{C}_{i-1}^{T}\delta\boldsymbol{\phi}_{i-1}\right)\right)^{+}\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i\right) && \text{(74e)}\\
&= \log\left(\left(\iota + \left(\mathbf{V}\begin{bmatrix}\overline{\mathbf{C}}_{i-1}^{T} & -\overline{\mathbf{C}}_{i-1}^{T}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\phi}_i\\ \delta\boldsymbol{\phi}_{i-1}\end{bmatrix}\right)\right)^{+}\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i\right) && \text{(74f)}\\
&\approx \log\left(\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i\right) + \mathbf{L}\left(\overline{\mathbf{q}}_{i-1}^{-1\,+}\overline{\mathbf{q}}_i\right)\begin{bmatrix}\overline{\mathbf{C}}_{i-1}^{T} & -\overline{\mathbf{C}}_{i-1}^{T}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\phi}_i\\ \delta\boldsymbol{\phi}_{i-1}\end{bmatrix} && \text{(74g)}
\end{aligned}$$

In the above derivation we used many identities from the quaternion algebra presented in Barfoot et al. (2011). We have also defined $\overline{\mathbf{C}}_{i-1}$ to be the rotation matrix built from the quaternion $\overline{\mathbf{q}}_{i-1}$. The trickiest manipulation was between (74c) and (74d). The subexpressions were simplified as follows (subscripts omitted for clarity):

$$\mathbf{q}^{+}(\mathbf{V}\delta\boldsymbol{\phi}) = \mathbf{q}^{+}(\mathbf{V}\delta\boldsymbol{\phi})^{+}\underbrace{\mathbf{q}^{-1^{+}}\mathbf{q}}_{\boldsymbol{\iota}} \tag{75a}$$

$$= \left(\mathbf{q}^{+}(\mathbf{V}\delta\boldsymbol{\phi})^{+}\mathbf{q}^{-1}\right)^{+}\mathbf{q} \tag{75b}$$

$$= \left(\mathbf{q}^{+}\mathbf{q}^{-1^{\oplus}}\mathbf{V}\delta\boldsymbol{\phi}\right)^{+}\mathbf{q} \tag{75c}$$

$$= \left(\begin{bmatrix}\mathbf{C} & \mathbf{0}\\ \mathbf{0}^{T} & 1\end{bmatrix}\frac{1}{2}\begin{bmatrix}\mathbf{1}\\ \mathbf{0}^{T}\end{bmatrix}\delta\boldsymbol{\phi}\right)^{+}\mathbf{q} \tag{75d}$$

$$= \left(\frac{1}{2}\begin{bmatrix}\mathbf{C}\\ \mathbf{0}^{T}\end{bmatrix}\delta\boldsymbol{\phi}\right)^{+}\mathbf{q} \tag{75e}$$

$$= \left(\frac{1}{2}\begin{bmatrix}\mathbf{1}\\ \mathbf{0}^{T}\end{bmatrix}\mathbf{C}\delta\boldsymbol{\phi}\right)^{+}\mathbf{q} \tag{75f}$$

$$= (\mathbf{V}\mathbf{C}\delta\boldsymbol{\phi})^{+}\mathbf{q} \tag{75g}$$

In the above, $\mathbf{V}$ is defined in (68) and $\mathbf{C}$ is the rotation matrix built from the quaternion, $\mathbf{q}$.

References

Barfoot, T. D., Forbes, J. R., and Furgale, P. T. (2011). Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica*, 68(1-2):101–112.

Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1987). *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers Inc., Los Altos, California, USA.

de Boor, C. (2002). *The Handbook of Computer Aided Geometric Design*, chapter Spline Basics, pages 141–163. Elsevier Science B.V., Amsterdam, The Netherlands.

Furgale, P. T., Barfoot, T. D., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (to appear)*, St. Paul, MN.

Hughes, P. C. (1986). *Spacecraft Attitude Dynamics*. John Wiley & Sons, New York.

Kim, M.-J., Kim, M.-S., and Shin, S. Y. (1995). A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 369–376, New York, NY, USA. ACM.

Park, F. C. and Ravani, B. (1997). Smooth invariant interpolation of rotations. *ACM Trans. Graph.*, 16:277–295.

Qin, K. (2000). General matrix representations for B-splines. *The Visual Computer*, 16:177–186.

Shuster, M. D. (1993). A survey of attitude representations. *Journal of the Astronautical Sciences*, 41(4):439–517.