Interpolation of Rotation and Motion*

Olivier A. Bauchau and Shilei Han
University of Michigan-Shanghai Jiao Tong University Joint Institute
Tel.: +86-21-3420-7937
Fax: +86-21-3420-6525

July 7, 2015

Abstract

In Cosserat solids such as shear deformable beams and shells, the displacement and rotation fields are independent. The finite element implementation of these structural components within the framework of flexible multibody dynamics requires the interpolation of rotation and motion fields. In general, the interpolation process does not preserve fundamental properties of the interpolated field. For instance, interpolation of an orthogonal rotation tensor does not yield an orthogonal tensor, and furthermore, does not preserve the tensorial nature of the rotation field. Consequently, many researchers have been reluctant to apply the classical interpolation tools used in finite element procedures to interpolate these fields. This paper presents a systematic study of interpolation algorithms for rotation and motion. All the algorithms presented here preserve the fundamental properties of the interpolated rotation and motion fields, and furthermore, preserve their tensorial nature. It is also shown that the interpolation of rotation and motion is as accurate as the interpolation of displacement, a widely accepted tool in the finite element method. The algorithms presented in this paper provide interpolation tools for rotation and motion that are accurate, easy to implement, and physically meaningful.

1 Introduction

Rigid multibody systems are characterized by two distinguishing features: the rigid body components of the system undergo finite relative rotations and these components are connected by mechanical joints that impose restrictions on their relative motion [1]. When dealing with flexible multibody systems, each component of the system could be flexible, adding to the complexity and nonlinearity of the problem.

Classical formulations of flexible multibody systems are based on the floating frame of reference approach [2, 3, 4], but the accuracy of these methods is often difficult to ascertain. Consequently, the finite element method has found increasing use in the analysis of flexible multibody systems; the textbook of Géradin and Cardona [5] is devoted to this topic entirely. One of the hallmarks of this approach is its versatility: structures of arbitrary topology can be modeled easily. This important feature is achieved using a unified kinematic representation for all elements. Typically, at each node of the model, discrete displacement components resolved in a common inertial frame are used. Models of the complete system are then obtained through a straightforward Boolean assembly process [6, 7].

Multibody System Dynamics, **31**(3): pp 339-370, 2014

To achieve computational efficiency, flexible components in multibody systems are often idealized as thin structures, such as shear deformable beams or shells. In Euler-Bernoulli beams and Kirchhoff plates, transverse shear strains are assumed to vanish and hence, rotation of the cross-sectional plane and of the normal material line, respectively, are obtained from derivatives of the displacement field, and curvatures are expressed in terms of second derivatives of the same field [8]. In contrast, shear deformable beams and plates, often called Timoshenko beams and Mindlin plates, respectively, are Cosserat solids: the kinematics of these structural components are described in terms of two independent fields, a displacement field and a rotation field, or equivalently, in terms of a motion field.

In two- and three-dimensional elasticity, the rotation field is not independent of the displacement field. Indeed, the polar decomposition theorem can be used to decompose the deformation gradient tensor into a stretch tensor and an orthogonal rotation tensor [9]. Because this decomposition is unique, the deformation gradient tensor, a function of the displacement field only, defines the rotation field unambiguously. Shabana [10] has discussed this problem in details and underlines the pitfalls associated with interpolation of the rotation field in such cases. This contrasts with Cosserat solids for which the displacement and rotation fields are independent.

In flexible multibody systems, it is imperative to adopt a kinematic description that can deal with all system components and their interconnections in a rational manner. Typical multibody systems involve flexible elements such as beams and shells, but also rigid bodies and joints, such as revolute or prismatic joints, among many others. Consider the connection of a rigid body at the tip of a beam. The configuration of the rigid body is described by the three displacement components of one of its points and three rotations, all resolved in an inertial frame. If the beam is shear deformable, the degrees of freedom of its tip node are identical, three displacement and three rotation components resolved in an inertial frame. Connection of the two elements is done through a simple Boolean assembly process, as in any finite element formulation. In contrast, if the beam is an Euler-Bernoulli beam, a mismatch exists between the degrees of freedom of the rigid body and those of the beam. The connection will require special attention: nonlinear relationships must be developed to express the degrees of freedom of the beam (displacements and slopes) in terms of those of the rigid body (displacement and rotations), or vice-versa [11, 12].

In Euler-Bernoulli beams and Kirchhoff plates, curvatures are expressed in terms of second derivatives of the displacement field and this impacts formulations in two important aspects. First, the accuracy of the curvature field is low because accuracy degrades as the order of the derivatives increases. Second, C^1 continuity is required at the inter-element boundaries. While this requirement is not difficult to satisfy for beams, achieving C^1 continuity for plate and shell elements is very arduous. This is the reason why shear deformable beams and plates are now used almost exclusively in commercial finite element packages. Textbook in finite element [6, 7] focus on shear deformable elements only, because of their superior performance and ease of implementation.

The above discussion indicates that it is desirable to formulate beam and shells as Cosserat solids in flexible multibody systems. Even if shear deformations are negligible, formulations based on kinematics involving displacement and rotation components are preferable because of their inherent versatility, accuracy, and low order of continuity requirements.

The manipulation of rotations in rigid systems is at the heart of multibody dynamics formulations; details can be found in many textbooks. When dealing with Cosserat solids, an additional difficulty arises: the rotation field must be interpolated over the span of the element to provide a continuous description of the rotation field and associated strain field. Interpolation of displacement fields within each element is at the heart of the finite element method and has been used for decades [6, 7]. In classical formulations, the displacement field, which forms a linear space, is interpolated based on its nodal values using simple polynomial shape functions.

Application of the same, linear interpolation technique to rotation fields has been the subject of controversy, because finite rotation fields do not form a linear space. Consider two nodal rotation

tensors, denoted $\hat{\underline{\underline{R}}}_1$ and $\hat{\underline{\underline{R}}}_2$, and the following simple, one-dimensional interpolation, $\underline{\underline{R}}(s) = (1 - s)\hat{\underline{\underline{R}}}_1/2 + (1 + s)\hat{\underline{\underline{R}}}_2/2$, where $s \in [-1, +1]$ is the non-dimensional spatial variable. While the nodal rotation tensors are orthogonal, it is clear that the interpolated tensor, $\underline{\underline{R}}(s)$, is not an orthogonal tensor, except for $s =$ - 1 or +1, *i.e.*, at the nodes. This means that the interpolated tensor cannot be interpreted as a rotation tensor. Let array $\underline{\alpha}^T = \{\phi, \theta, \psi\}$ store the three Euler angles, denoted ϕ, θ, and ψ, representing a rotation. If arrays $\hat{\underline{\alpha}}_1$ and $\hat{\underline{\alpha}}_2$ represent nodal rotations, a similar interpolation technique yields the rotation field as $\underline{\alpha}(s) = (1-s)\hat{\underline{\alpha}}_1/2 + (1+s)\hat{\underline{\alpha}}_2/2$. Because Euler angles are sequence dependent [1], it is difficult to interpret $\underline{\alpha}(s)$ as a meaningful rotation, except for $s =$ - 1 or +1.

While the simple interpolation technique described above does not guarantee that the interpolated quantities are rotation tensors, they are of a tensorial nature. Consider two bases, denoted $\mathcal{B}$ and $\mathcal{B}^*$, and let $\underline{\underline{R}}_R$ be the rotation tensor that bring basis $\mathcal{B}$ to $\mathcal{B}^*$. If $\hat{\underline{\underline{R}}}_1$ and $\hat{\underline{\underline{R}}}_2$ denote the components of the nodal rotation tensors resolved in basis $\mathcal{B}$, $\hat{\underline{\underline{R}}}_1^* = \underline{\underline{R}}_R^T \hat{\underline{\underline{R}}}_1 \underline{\underline{R}}_R$ and $\hat{\underline{\underline{R}}}_2^* = \underline{\underline{R}}_R^T \hat{\underline{\underline{R}}}_2 \underline{\underline{R}}_R$ denote the components of the same tensors resolved in basis $\mathcal{B}^*$. The linear nature of the interpolation then implies that $\underline{\underline{R}}^*(s) = (1 - s)\hat{\underline{\underline{R}}}_1^*/2 + (1 + s)\hat{\underline{\underline{R}}}_2^*/2$, where $\underline{\underline{R}}^*(s) = \underline{\underline{R}}_R^T \underline{\underline{R}}(s) \underline{\underline{R}}_R$. In other words, the interpolation respects the tensorial nature of the interpolated quantities. Note that Euler angles are not tensorial components, and hence, interpolations of these quantities has little meaning, as pointed out earlier.

To evaluate the strain energy stored in the structure, the finite element method requires interpolation of the rotation field and of the associated strain or curvature field. For instance, the curvature vector, $\underline{\kappa}$, can be defined as $\underline{\kappa} = \text{axial}(\underline{\underline{R}}' \underline{\underline{R}}^T)$, where notation $(\cdot)'$ indicates a derivative with respect to the spatial variable. This leads to the following question: if the rotation tensor and its spatial derivative are interpolated using suitable techniques, do these techniques respect the tensorial nature of the interpolated curvature vector? This question might look technical, but is a fundamental physical importance. Let the configuration of a structure in its unstrained state be represented by nodal rotations denoted $\hat{\underline{\underline{R}}}_i$ and assume that a suitable interpolation technique yields the correct interpolated strain field, $\underline{\kappa}(s) = \underline{0}$. If a different basis is used to represent the exact same structure, the nodal rotation become $\hat{\underline{\underline{R}}}_i^* = \underline{\underline{R}}_R^T \hat{\underline{\underline{R}}}_i \underline{\underline{R}}_R$ and the interpolation technique should yield $\underline{\kappa}^*(s) = \underline{\underline{R}}_R^T \underline{\kappa}(s) = \underline{0}$. Clearly, if the structure is unstrained, it must remain so when represented in any basis. In conclusion, techniques used for the interpolation of the rotation field and of its spatial derivative must respect the tensorial nature of both rotation tensor and associated curvature field. In this paper, such algorithms will be called "tensorial algorithms."

Crisfield and Jelenić [13] were the first study the problem of interpolation of rotation fields from a theoretical viewpoint. They required the interpolated strain field to satisfy the "objectivity principle," which Malvern [9] defines as follows.

> Functions and fields whose values are scalars, vectors, or tensors are called **frame-indifferent** or **objective** if both the dependent and the independent vector and tensor variables transform according to Eqs. (6) to (8), while the scalar variables are unchanged.

Equations (6) and (7) Malvern is referring to are the standard equations for transforming the components of first- and second-order tensors. Equation (8), however, applies to the deformation gradient tensor: "This two-point tensor transforms like a vector under a change of basis operation."

In accordance with Malvern's definition of objectivity, Crisfield and Jelenić [13, 14] required the invariance of the interpolated strain tensor under the addition of a rigid body rotation and showed that the classical interpolation formulæ applied to finite rotation fields violate this objectivity criterion. They proved that the direct interpolation of total rotations, incremental rotations, and iterative rotations as used by Ibrahimbegovic [15], Cardona and Géradin [16], and Simo and Vu-Quoc [17], respectively, are not objective. At the same time, Jelenić and Crisfield [14] also mentioned that "The non-invariance and path-dependence in these formulations decrease with both

p-refinement and h-refinement and in practical applications cannot always be easily spotted." In fact, the formulation of Cardona and Géradin [16] is the basis for a commercial multibody dynamics code that has been shown to provide reliable and accurate predictions for over two decades.

The concept of objectivity of the strain field was proposed by researchers aiming at the development of material constitutive laws [9]. For this type of applications, the objectivity of the strain field is indispensable, because without it, material constitutive laws cannot be physically meaningful. The interpolation of rotation fields, however, is a quite different problem, which does not describe a physical phenomenon. Rather, it is mathematical process aimed at approximating the rotation field within one finite element as accurately as possible. Hence, the accuracy and convergence characteristics of this mathematical process are more relevant than its objectivity. In this paper, both objective and tensorial algorithms will be developed and it will be shown that the accuracy and convergence characteristics of the two types of algorithms are indistinguishable.

All the algorithms presented in this paper originate from simple minimization problems. In each case, it is proved that the proposed algorithm preserves the tensorial nature of both rotation and associated strain fields. Some of these algorithms are also shown to be objective. The next logical question is then: how do these algorithms compare to each other in terms of accuracy and convergence characteristics? This paper evaluates the convergence rate of the proposed algorithms as the size of the element is reduced. The conclusion of this study is simple: interpolation of rotation fields is as accurate as that of displacement fields, a widely accepted tool in the finite element method. Furthermore, the rate of convergence and accuracy of the objective algorithms are indistinguishable from those of the algorithms that preserve the tensorial nature of both rotation and associated strain fields.

All the arguments presented above also apply to the interpolation of motion [1]. When the displacement and rotation fields are treated as a single unit through the use of the motion tensor, finite element implementation will require the interpolation of the motion field and of its spatial derivative. To be physically meaningful, the interpolation techniques must preserve the tensorial nature of both motion and associated strain fields. This paper also presents different algorithms for the interpolation of motion fields; it will be proved that the algorithms preserve the tensorial nature of both motion and strain fields, and provide the same accuracy as that observed for the interpolation of displacement fields. Some of these algorithms are also shown to be objective.

This paper is structured as follows. For reference, classical interpolation techniques for displacement fields are reviewed in section 2. Next, interpolation techniques are presented for the rotation and motion fields in sections 3 and 4, respectively. In both cases, numerical results are presented.

2 Interpolation of displacement fields

The present section reviews the basic tools used for the interpolation of displacement fields. For simplicity, the results presented in this paper focus on displacement, rotation, and motion fields that depend on a single spatial variable, as would be the case for beams. Extension to two-dimensional fields, as required for plates and shells, is straightforward. Figure 1 depicts the typical configuration considered here, which consists of a four-noded, one-dimensional element, similar to a typical beam finite element. One-dimensional displacement, rotation, and motion fields are defined over the element, based on their nodal values. Notation $\hat{(\cdot)}$ is used to indicate nodal values, and fig. 1 shows the nodal values of the displacement, rotation, and motion fields, denoted $\hat{\underline{u}}_i$, $\hat{\underline{\underline{R}}}_i$, and $\hat{\underline{\underline{C}}}_i$, respectively, where $i = 1, 2, \ldots, N$, and N is the number of nodes of the element. Curvilinear variable $x_1 \in [0, \ell]$ defines length along the line that characterizes the geometry of the element of length ℓ.

The evaluation of the stiffness matrix for the finite element will require integration of the strain energy over the element. Typically, this integration is performed numerically using Gaussian quadra-

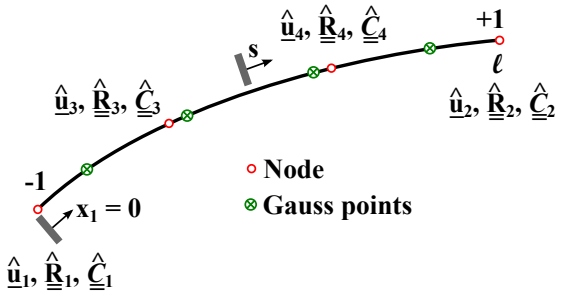

Figure 1: Rotation interpolation problem.

ture [6, 7], using the Gauss points indicated in fig. 1, a process that requires the knowledge of the kinematic and deformation fields at those Gauss points. Hence, these kinematic fields and their spatial derivative must be interpolated based on their nodal values to find the deformations at the Gauss points.

In the finite element method, the displacement field is interpolated based on its nodal values using polynomial shape functions

$$\underline{u}(x_1) = \sum_{i=1}^{N} h_i \hat{\underline{u}}_i, \tag{1a}$$

$$\underline{u}'(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J} \hat{\underline{u}}_i, \tag{1b}$$

where $h_i(s)$ are the shape functions and $h_i^+(s)$ their derivatives with respect to non-dimensional variable s. For convenience, the shape functions are expressed in terms of non-dimensional variable $s \in [-1, +1]$ and $J = \mathrm{d}x_1/\mathrm{d}s$ is the Jacobian of the coordinate transformation from variable x_1 to s. Notation $(\cdot)'$ indicates a derivative with respect to variable x_1. The expressions for the shape functions can be found in textbooks [6, 7].

2.1 Interpolation of displacements: numerical results

To illustrate the concepts described in the previous section, consider a displacement field, $\underline{u}^T = \{u_1, u_2, u_3\}$, defined as follows: $u_1 = \sin \omega s$, $u_2 = \cos \omega s - 1$, and $u_3 = 0.5s + \sin 2\omega s$, where $\omega = 2$ rad/m. Figure 2 shows this displacement field versus s. For this simple example, the corresponding strain field is defined as

$$\underline{\epsilon}(s) = \frac{\mathrm{d}\underline{u}}{\mathrm{d}s}, \tag{2}$$

and fig. 3 depicts the components of the strain vector, denoted $\underline{\epsilon}^T = \{\epsilon_1, \epsilon_2, \epsilon_3\}$.

Interval $s \in [-1, +1]$ was divided into N_e elements of equal length, and within each element, the displacement field was interpolated using linear, quadratic, cubic, and quartic shape functions, corresponding to elements of order $o_e = 1$, 2, 3, and 4, respectively.

To assess the accuracy of the algorithm, the interpolated strain, denoted $\underline{\epsilon}_a$, was compared to its exact counterpart, denoted $\underline{\epsilon}_e$. The strain error measure was selected as

$$e = \frac{1}{N_{gp}} \sum_{k=1}^{N_{gp}} \frac{\|\underline{\epsilon}_a(s_k) - \underline{\epsilon}_e(s_k)\|}{\|\underline{\epsilon}_e(s_k)\|}. \tag{3}$$

The exact and approximate strain components were computed at N_{gp} sampling points denoted s_k. Within each element, $o_e + 1$ sampling points were selected to coincide with the location of the

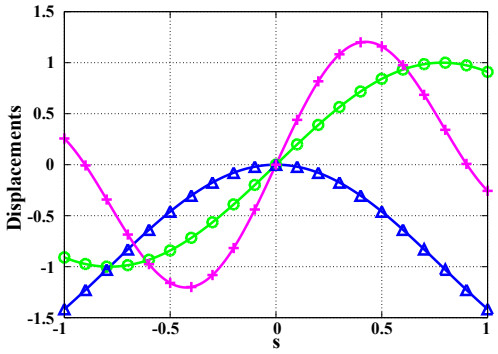

Figure 2: Displacement field versus s. Displacement components u_1 ($\circ$), u_2 ($\triangle$), u_3 ($+$).

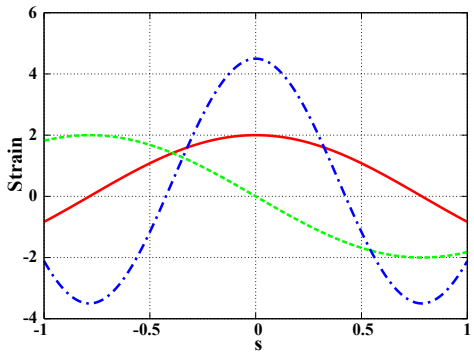

Figure 3: Strain field versus s: ϵ_1 (solid line), ϵ_2 (dashed line), ϵ_3 (dashed-dotted line).

Gauss-Legendre quadrature points within the element. Figure 4 shows the strain error measure defined by eq. (3) versus the number of elements, $N_e \in [2, 256]$, on a logarithmic plot, when eq. (2) is used to compute the strain field. Results are shown for linear, quadratic, cubic, and quartic shape functions.

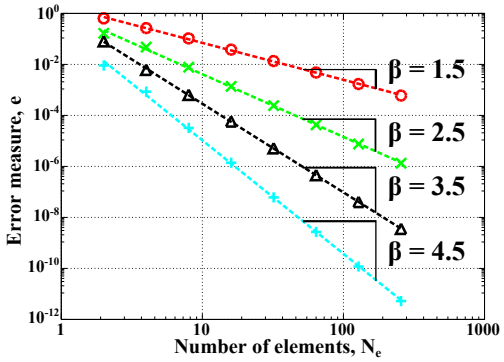

Figure 4: Strain error measure (3) versus number of elements when using algorithm 14 with linear ($\circ$), quadratic ($\times$), cubic ($\triangle$), and quartic ($+$) shape functions.

Linear regression of the numerical data was performed in all cases and are shown in dashed lines in fig. 4. Because logarithmic scales are used, this linear regression is of the following form: $\log e = \alpha - \beta \log N_e$. As the number of elements increases, the accuracy of the strain interpolation algorithm increases, yielding a positive coefficient β. The following relationship is observed from the data presented in fig. 4,

$$\beta \approx o_e + \frac{1}{2}, \tag{4}$$

where o_e is the order of the shape functions used in the interpolation.

3 Interpolation of rotation fields

In this section, a number of algorithms are presented for the interpolation of rotation. These algorithms are based on the interpolation of the rotation tensor, section 3.1, of Euler parameters, section 3.2, of the rotation parameter vector, section 3.3, and of relative rotations, section 3.4. The performance of these various algorithms is assessed in section 3.5.

3.1 Interpolation of the rotation tensor

Consider the following interpolation of the rotation field defined by discrete rotation tensors at N nodes

$$\underline{\underline{T}}(x_1) = \sum_{i=1}^{N} h_i \underline{\underline{\hat{R}}}_i, \tag{5a}$$

$$\underline{\underline{T}}'(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J} \underline{\underline{\hat{R}}}_i, \tag{5b}$$

where $\underline{\underline{\hat{R}}}_i$ is the orthogonal rotation tensor at node i, and h_i the shape functions. Although eqs. (5a) and (5b) give reasonable approximations for the interpolated rotation field and its derivative, respectively, they suffer a major drawback: the interpolated quantities are not orthogonal rotation tensors. In particular, except at the nodes, the interpolated rotation matrix is not orthogonal, *i.e.*, $\underline{\underline{T}}\,\underline{\underline{T}}^T \neq \underline{\underline{I}}$ and furthermore, $\underline{\underline{T}}'\underline{\underline{T}}^T \neq \tilde{\kappa}$, *i.e.*, the product $\underline{\underline{T}}'\underline{\underline{T}}^T$ does not yield a skew-symmetric matrix. It seems natural to use the polar decomposition theorem to extract an orthogonal rotation tensor from tensor $\underline{\underline{T}}$. This approach was used by Betsch and Steinmann [18], and by Romero *et al.* [19, 20]. The section below shows that this approach derives from a minimization procedure and is both tensorial and objective.

3.1.1 Extraction of the rotation tensor through singular value decomposition

The following question can be asked: is it possible to extract an orthogonal tensor, $\underline{\underline{R}}$, that is as close as possible to matrix $\underline{\underline{T}}$? Of course, the answer to this question is not unique as long as the meaning of "as close as possible" is not defined in a precise mathematical sense. To that effect, the following minimization problem is defined: $\min_{\underline{\underline{R}},\, \underline{\underline{R}}\underline{\underline{R}}^T=\underline{\underline{I}}} \|\underline{\underline{T}} - \underline{\underline{R}}\|_F$. Once a norm is chosen, the orthogonal rotation tensor can be evaluated. In this effort, the norm is selected to be the *Frobenius norm*. The Frobenius norm of matrix $\underline{\underline{A}}$ is defined as $\|\underline{\underline{A}}\|_F = \sqrt{\mathrm{tr}(\underline{\underline{A}}\,\underline{\underline{A}}^T)}$.

The singular decomposition of matrix $\underline{\underline{T}}$ is performed, leading to $\underline{\underline{T}} = \underline{\underline{U}}\,\underline{\underline{\Sigma}}\,\underline{\underline{V}}^T$, where matrices $\underline{\underline{U}}$ and $\underline{\underline{V}}$ are orthogonal matrices. Matrix $\underline{\underline{\Sigma}} = \mathrm{diag}(\sigma_1, \sigma_2, \sigma_3)$ stores the positive singular values of $\underline{\underline{T}}$, denoted σ_1, σ_2, and σ_3. The Frobenius norm of the difference, $(\underline{\underline{T}} - \underline{\underline{R}})$, now becomes $\|\underline{\underline{T}} - \underline{\underline{R}}\|_F^2 = \mathrm{tr}[(\underline{\underline{T}} - \underline{\underline{R}})(\underline{\underline{T}} - \underline{\underline{R}})^T] = \mathrm{tr}[\underline{\underline{U}}(\underline{\underline{\Sigma}} - \underline{\underline{\bar{R}}})(\underline{\underline{\Sigma}} - \underline{\underline{\bar{R}}}^T)\underline{\underline{U}}^T] = \mathrm{tr}[(\underline{\underline{\Sigma}} - \underline{\underline{\bar{R}}})(\underline{\underline{\Sigma}} - \underline{\underline{\bar{R}}}^T)]$, where the last equality follows from the orthogonality of matrix $\underline{\underline{U}}$. Orthogonal matrix $\underline{\underline{\bar{R}}}$ is defined as $\underline{\underline{\bar{R}}} = \underline{\underline{U}}^T\underline{\underline{R}}\,\underline{\underline{V}}$. Evaluating the trace operator yields $\|\underline{\underline{T}} - \underline{\underline{R}}\|_F^2 = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 + 3 - 2(\sigma_1\bar{R}_{11} + \sigma_2\bar{R}_{22} + \sigma_3\bar{R}_{33})$. Because matrix $\underline{\underline{\bar{R}}}$ is orthogonal, is can be expressed in terms of its Euler parameters, denoted e_1, e_2, and e_3, and hence, $\|\underline{\underline{T}} - \underline{\underline{R}}\|_F^2 = (\sigma_1 - 1)^2 + (\sigma_2 - 1)^2 + (\sigma_3 - 1)^2 + 4e_1^2(\sigma_2 + \sigma_3) + 4e_2^2(\sigma_1 + \sigma_3) + 4e_3^2(\sigma_1 + \sigma_2)$. Since the three singular values are positive, the Frobenius norm is minimized for $e_1 = e_2 = e_3 = 0$, leading to $\underline{\underline{\bar{R}}} = \underline{\underline{I}}$, and finally,

$$\underline{\underline{R}} = \underline{\underline{U}}\,\underline{\underline{V}}^T. \tag{6}$$

This orthogonal rotation tensor minimizes the Frobenius norm of the difference $(\underline{\underline{T}} - \underline{\underline{R}})$.

The same result can also be obtained with the help of the polar decomposition theorem, see eq. (58), which implies that matrix $\underline{\underline{T}}$ can be decomposed as $\underline{\underline{T}} = \underline{\underline{R}}\,\underline{\underline{U}}$, where $\underline{\underline{R}}$ is an orthogonal matrix and $\underline{\underline{U}}$ a positive-definite, symmetric matrix. A minimization problem similar to that presented in the previous paragraph reveals that the minimum of the Frobenius norm is obtained when $\underline{\underline{U}} = \underline{\underline{I}}$, leading to $\underline{\underline{T}} = \underline{\underline{R}}$.

Algorithm 1 (Finite rotation interpolation) *Interpolation of a rotation field defined by nodal rotation tensors, $\hat{\underline{\underline{R}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use interpolation formula, eq.* (5a)*, to obtain matrix $\underline{\underline{T}}$.* ***Step (2)****. Use singular value or polar decomposition to extract orthogonal rotation matrix $\underline{\underline{R}}$, see eq.* (6)*.*

It is easy to prove that algorithm 1 is tensorial. Under a change of basis, the nodal rotations now become $\hat{\underline{\underline{R}}}_i^* = \underline{\underline{R}}_R^T \hat{\underline{\underline{R}}}_i \underline{\underline{R}}_R$, where notation $(\cdot)^*$ indicates a change of basis operation. Interpolation equation (5a) then yields $\bar{\underline{\underline{T}}} = \underline{\underline{R}}_R^T \underline{\underline{T}}\,\underline{\underline{R}}_R$, where notation $\bar{(\cdot)}$ indicates interpolated quantities after the change of basis. The rotation tensors extracted from matrices $\underline{\underline{T}}$ and $\bar{\underline{\underline{T}}}$ using the singular value decomposition are $\underline{\underline{R}} = \underline{\underline{U}}\,\underline{\underline{V}}^T$ and $\bar{\underline{\underline{R}}} = \bar{\underline{\underline{U}}}\,\bar{\underline{\underline{V}}}^T$, respectively. Because the singular value decomposition is unique, $\bar{\underline{\underline{U}}} = \underline{\underline{R}}_R^T \underline{\underline{U}}\,\underline{\underline{R}}_R$ and $\bar{\underline{\underline{V}}} = \underline{\underline{R}}_R^T \underline{\underline{V}}\,\underline{\underline{R}}_R$, leading to $\bar{\underline{\underline{R}}} = \underline{\underline{R}}_R^T \underline{\underline{R}}\,\underline{\underline{R}}_R = \underline{\underline{R}}^*$. Clearly, under a change of basis, algorithm 1 yields an interpolated rotation resolved in the new basis. Hence, algorithm 1 is tensorial.

It is easy to prove that algorithm 1 is also objective. Assume that a rigid body rotation characterized by rigid rotation tensor $\underline{\underline{R}}_R$ is superposed to the rotation field. The nodal rotations now become $\hat{\underline{\underline{R}}}_i^+ = \underline{\underline{R}}_R \hat{\underline{\underline{R}}}_i$, where notation $(\cdot)^+$ indicates the composed rotation. Interpolation equation (5a) then yields $\bar{\underline{\underline{T}}} = \underline{\underline{R}}_R \underline{\underline{T}}$. The rotation tensors extracted from matrices $\underline{\underline{T}}$ and $\bar{\underline{\underline{T}}}$ using the singular value decomposition are $\underline{\underline{R}} = \underline{\underline{U}}\,\underline{\underline{V}}^T$ and $\bar{\underline{\underline{R}}} = \bar{\underline{\underline{U}}}\,\bar{\underline{\underline{V}}}^T$, respectively. Because the singular value decomposition is unique, $\bar{\underline{\underline{U}}} = \underline{\underline{R}}_R \underline{\underline{U}}$ and $\bar{\underline{\underline{V}}} = \underline{\underline{V}}$, leading to $\bar{\underline{\underline{R}}} = \underline{\underline{R}}_R \underline{\underline{R}} = \underline{\underline{R}}^+$. Clearly, algorithm 1 yields the correct composition of rotation for the interpolated field. Hence, algorithm 1 is objective.

3.1.2 Extraction of the rotation tensor through Cayley's decomposition

The extraction procedure presented above is based on the singular value decomposition, or alternatively, on the polar decomposition. Because both approaches are computationally expensive, an alternative approach is presented here, based on Cayley's decomposition. Let $\underline{\underline{R}}$ be an orthogonal tensor; Cayley's decomposition [1] states that $\tilde{a} = (\underline{\underline{R}} - \underline{\underline{I}})(\underline{\underline{R}} + \underline{\underline{I}})^{-1}$, where $\underline{a}$ are Cayley's parameters of rotation tensor $\underline{\underline{R}}$. On the other hand, if $\underline{\underline{T}}$ is an arbitrary matrix, $\underline{\underline{A}} = (\underline{\underline{T}} - \underline{\underline{I}})(\underline{\underline{T}} + \underline{\underline{I}})^{-1}$, where matrix $\underline{\underline{A}}$ is no longer a skew-symmetric matrix.

The following minimization problem is defined: $\min_{\underline{a}} \|\tilde{a} - \underline{\underline{A}}\|_F$, where $\underline{\underline{A}}$ is an arbitrary, non-skew-symmetric matrix. Here again, the Frobenius norm is selected and tedious algebra reveals that $\|\tilde{a} - \underline{\underline{A}}\|_F^2 = 2\underline{a}^T\underline{a} - 4\underline{a}^T \mathrm{axial}(\underline{\underline{A}}) + \|\underline{\underline{A}}\|_F^2$. Imposing the vanishing of the derivative of the Frobenius norm with respect to $\underline{a}$ then yields $4\underline{a} - 4\mathrm{axial}(\underline{\underline{A}}) = \underline{0}$, and finally

$$\underline{a} = \mathrm{axial}(\underline{\underline{A}}) \Longleftrightarrow \tilde{a} = \mathrm{skew}(\underline{\underline{A}}). \tag{7}$$

In summary, the skew-symmetric matrix that approximates a non-skew-symmetric matrix most closely in the space of the Frobenius norm is given by eq. (7).

The above result is now applied to the problem at hand. Matrix $\underline{\underline{A}} = (\underline{\underline{T}} - \underline{\underline{I}})(\underline{\underline{T}} + \underline{\underline{I}})^{-1}$ is not a skew-symmetric matrix but it is closely approximated by skew-symmetric matrix $\tilde{a} = \mathrm{skew}[(\underline{\underline{T}} - \underline{\underline{I}})(\underline{\underline{T}} + \underline{\underline{I}})^{-1}]$, leading to the following expression for Cayley's parameters

$$\underline{a} = \mathrm{axial}[(\underline{\underline{T}} - \underline{\underline{I}})(\underline{\underline{T}} + \underline{\underline{I}})^{-1}]. \tag{8}$$

The orthogonal rotation tensor then becomes

$$\underline{\underline{R}} = \frac{1}{1+\underline{a}^T\underline{a}}\left[(1+\underline{a}^T\underline{a})\underline{\underline{I}} + 2\,\tilde{a} + 2\tilde{a}\tilde{a}\right]. \tag{9}$$

Algorithm 2 (Finite rotation interpolation) *Interpolation of a rotation field defined by nodal rotation tensors,* $\underline{\underline{\hat{R}}}_i$, $i = 1, 2, \ldots, N$. ***Step (1)****. Use interpolation formula, eq.* (5a)*, to obtain matrix* $\underline{\underline{T}}$. ***Step (2)****. Use eq.* (8) *to find Cayley's rotation parameters,* $\underline{a}$. ***Step (3)****. Use eq.* (9) *to find the interpolated orthogonal rotation tensor,* $\underline{\underline{R}}$.

Under a change of basis, the nodal rotations now become $\underline{\underline{\hat{R}}}_i^* = \underline{\underline{R}}_R^T\underline{\underline{\hat{R}}}_i\underline{\underline{R}}_R$. Let $\underline{\underline{\bar{T}}}$ be the interpolated rotation field after a change of basis operation and $\underline{\bar{a}}$ Cayley's parameters obtained from eq. (8). Note that eq. (8) is tensorial in nature, *i.e.*, $\underline{\bar{a}} = \text{axial}[(\underline{\underline{\bar{T}}} - \underline{\underline{I}})(\underline{\underline{\bar{T}}} + \underline{\underline{I}})^{-1}] = \text{axial}[\underline{\underline{R}}_R^T(\underline{\underline{T}} - \underline{\underline{I}})\underline{\underline{R}}_R\underline{\underline{R}}_R^T(\underline{\underline{T}} + \underline{\underline{I}})^{-1}\underline{\underline{R}}_R] = \text{axial}[\underline{\underline{R}}_R^T(\underline{\underline{T}} - \underline{\underline{I}})(\underline{\underline{T}} + \underline{\underline{I}})^{-1}\underline{\underline{R}}_R] = \underline{\underline{R}}_R^T\text{axial}[(\underline{\underline{T}} - \underline{\underline{I}})(\underline{\underline{T}} + \underline{\underline{I}})^{-1}] = \underline{\underline{R}}_R^T\underline{a}$. Because Cayley's rotation parameters form a vectorial parameterization of rotation, eq. (9) yields $\underline{\underline{\bar{R}}} = \underline{\underline{R}}_R^T\underline{\underline{R}}\,\underline{\underline{R}}_R = \underline{\underline{R}}^*$, which proves that algorithm 2 is tensorial. This algorithm is based on the identification of Cayley's rotation parameters, a specific parameterization of rotation, and is hence not intrinsic. Note that rotation tensors extracted by algorithms 1 and 2 approximate interpolated matrix $\underline{\underline{T}}$ closely, but are not equal to each other. No unique solution exists for the extraction of an orthogonal rotation tensor from matrix $\underline{\underline{T}}$.

3.1.3 Computation of curvature

The curvature vector is defined as $\tilde{\kappa} = \underline{\underline{R}}'\underline{\underline{R}}^T$ and its evaluation requires the knowledge of the spatial derivative of the rotation field. Equation (5b) yields an approximation to the spatial derivative of the rotation field, denoted $\underline{\underline{T}}'$. For the exact rotation field, $\underline{\underline{R}}\,\underline{\underline{R}}^T = \underline{\underline{I}}$ implies that $\underline{\underline{R}}'\underline{\underline{R}}^T + (\underline{\underline{R}}'\underline{\underline{R}}^T)^T = \underline{\underline{0}}$, *i.e.*, matrix $\underline{\underline{R}}'\underline{\underline{R}}^T$ is skew-symmetric. Due to the approximation inherent to the interpolation process, $\underline{\underline{T}}'\underline{\underline{T}}^T + (\underline{\underline{T}}'\underline{\underline{T}}^T)^T \neq \underline{\underline{0}}$. Even after extraction of an orthogonal rotation matrix using algorithm 1 or 2, $\underline{\underline{T}}'\underline{\underline{R}}^T + (\underline{\underline{T}}'\underline{\underline{R}}^T)^T \neq \underline{\underline{0}}$.

Matrix $\underline{\underline{T}}'\underline{\underline{R}}^T$ is not a skew-symmetric matrix but in view of eq. (7), it is closely approximated by skew-symmetric matrix $\tilde{\kappa} = \text{skew}(\underline{\underline{T}}'\underline{\underline{R}}^T)$, leading to the following expression for the curvature vector

$$\tilde{\kappa} = \text{skew}(\underline{\underline{T}}'\underline{\underline{R}}^T), \quad \underline{\kappa} = \text{axial}(\underline{\underline{T}}'\underline{\underline{R}}^T). \tag{10}$$

Algorithm 3 (Curvature interpolation) *Interpolation of a curvature field defined by nodal rotation tensors,* $\underline{\underline{\hat{R}}}_i$, $i = 1, 2, \ldots, N$. ***Step (1)****. Use algorithm 1 to obtain orthogonal rotation matrix* $\underline{\underline{R}}$. ***Step (2)****. Use interpolation formula, eq.* (5b)*, to obtain matrix* $\underline{\underline{T}}'$. ***Step (3)****. Use eq.* (10) *to find the interpolated curvature.*

Algorithm 4 (Curvature interpolation) *Interpolation of a curvature field defined by nodal rotation tensors,* $\underline{\underline{\hat{R}}}_i$, $i = 1, 2, \ldots, N$. ***Step (1)****. Use algorithm 2 to obtain orthogonal rotation matrix* $\underline{\underline{R}}$. ***Step (2)****. Use interpolation formula, eq.* (5b)*, to obtain matrix* $\underline{\underline{T}}'$. ***Step (3)****. Use eq.* (10) *to find the interpolated curvature.*

It is easy to prove that algorithms 3 and 4 are both tensorial. Under a change of basis, algorithms 1 and 2 both yield $\underline{\underline{\bar{R}}} = \underline{\underline{R}}_R^T\underline{\underline{R}}\,\underline{\underline{R}}_R$, and under the same operation, eq. (5b) yields $\underline{\underline{\bar{T}}}' = \underline{\underline{R}}_R^T\underline{\underline{T}}'\underline{\underline{R}}_R$. Equation (10) then leads to $\bar{\tilde{\kappa}} = \text{skew}(\underline{\underline{\bar{T}}}'\underline{\underline{\bar{R}}}^T) = \text{skew}(\underline{\underline{R}}_R^T\underline{\underline{T}}'\underline{\underline{R}}_R\underline{\underline{R}}_R^T\underline{\underline{R}}^T\underline{\underline{R}}_R) = \underline{\underline{R}}_R^T\text{skew}(\underline{\underline{T}}'\underline{\underline{R}}^T)\underline{\underline{R}}_R = \underline{\underline{R}}_R^T\tilde{\kappa}\underline{\underline{R}}_R$, which proves that the algorithm is tensorial.

It is easy to prove that algorithms 3 is also objective. Under a composition of rotation, algorithm 1 yields $\underline{\underline{\bar{R}}} = \underline{\underline{R}}_R\underline{\underline{R}}$, and under the same operation, eq. (5b) yields $\underline{\underline{\bar{T}}}' = \underline{\underline{R}}_R\underline{\underline{T}}'$. Equation (10)

then leads to $\bar{\tilde{\kappa}} = \text{skew}(\underline{\underline{\bar{T}}}' \underline{\underline{\bar{R}}}^T) = \text{skew}(\underline{\underline{R}}_R \underline{\underline{T}}' \underline{\underline{R}}^T \underline{\underline{R}}_R^T) = \underline{\underline{R}}_R \tilde{\kappa} \underline{\underline{R}}_R^T$, or $\bar{\underline{\kappa}} = \underline{\underline{R}}_R \underline{\kappa}$. This last equation can be recast as $(\underline{\underline{R}}_R \underline{\underline{R}})^T \bar{\underline{\kappa}} = \underline{\underline{R}}^T \underline{\kappa}$, which proves that the algorithm is objective because the components of strain field resolved in the material basis, $\underline{\underline{R}}^T \underline{\kappa}$, are identical to their counterpart after the addition of the rigid body rotation, $(\underline{\underline{R}}_R \underline{\underline{R}})^T \bar{\underline{\kappa}}$. Note that algorithm 4 is not objective because algorithm 2, which it is based on, is not.

3.2 Interpolation of Euler parameters

Consider the following interpolation of the rotation field defined by discrete Euler parameters at N nodes

$$\hat{g}(x_1) = \sum_{i=1}^{N} h_i \hat{e}_i, \tag{11a}$$

$$\hat{g}'(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J} \hat{e}_i, \tag{11b}$$

where $\hat{e}_i$ is the array of Euler parameters at node i, and h_i the shape functions. While quaternions have been used in multibody dynamics simulations [21, 22], the computational cost of dealing with four parameters instead of a minimum set and the enforcement of the associated normality condition have limited their use. Although eqs. (11a) and (11b) give reasonable approximations for the interpolated rotation field and its derivative, respectively, they suffer a major drawback: the interpolated Euler parameters no longer satisfy the normality condition. In particular, except at the nodes, the interpolated Euler parameters do not form a unit quaternion, *i.e.*, $\hat{g}^T \hat{g} \neq 1$ and furthermore, $\hat{g}^T \hat{g}' \neq 0$.

3.2.1 Extraction of the Euler parameters through normalization

The following question can be asked: is it possible to extract a unit quaternion, $\hat{e}$, that is as close as possible to quaternion $\hat{g}$? To that effect, the following stationarity condition is imposed

$$\delta_{\hat{e},\lambda} \left(\|\hat{g} - \hat{e}\| - \frac{\lambda}{2}(\hat{e}^T \hat{e} - 1) \right) = 0, \tag{12}$$

where δ indicates the variation operation and λ is the Lagrange multiplier used to impose the normality condition. Taking a derivative of eq. (12) with respect to $\hat{e}$ yields $(\hat{e} - \hat{g})/\|\hat{g} - \hat{e}\| - \lambda \hat{e} = 0$. Multiplying this equation by $\hat{e}^T$ yield the Lagrange multiplier as $\lambda = \hat{e}^T(\hat{e} - \hat{g})/\|\hat{g} - \hat{e}\|$. Eliminating λ then leads to $(\underline{\underline{I}} - \hat{e}\hat{e}^T)(\hat{e} - \hat{g}) = 0$ and finally, $(\underline{\underline{I}} - \hat{e}\hat{e}^T)\hat{g} = 0$. Solution of this last equation yields $\hat{e} = \hat{g}/\sqrt{\hat{g}^T \hat{g}}$, written as

$$\hat{e} = \underline{\underline{H}}(\hat{g}) \hat{g}. \tag{13}$$

Tensor $\underline{\underline{H}}$, of size 4×4, is defined as

$$\underline{\underline{H}}(\hat{g}) = \frac{1}{\sqrt{\hat{g}^T \hat{g}}} \underline{\underline{I}}, \tag{14}$$

and the following properties are easily verified

$$\underline{\underline{H}}(\underline{\underline{D}}\, \hat{g}) = \underline{\underline{H}}(\hat{g}), \quad \underline{\underline{H}}(\hat{g}) = \underline{\underline{D}}\, \underline{\underline{H}}(\hat{g}) \underline{\underline{D}}^T, \tag{15}$$

where $\underline{\underline{D}}$ is an arbitrary rotation tensor, see eq. (52).

Algorithm 5 (Finite rotation interpolation) *Interpolation of a rotation field defined by nodal Euler parameters, $\hat{e}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use interpolation formula, eq.* (11a)*, to obtain quaternion $\hat{g}$.* ***Step (2)****. Use eq.* (13) *to obtain Euler parameters $\hat{e}$.*

Under a change of basis, nodal Euler parameters now become $\underline{\underline{D}}_R^T \hat{e}_i$, where matrix $\underline{\underline{D}}_R$ represents the rigid rotation. Interpolation equation (11a) then yields $\bar{\hat{g}} = \underline{\underline{D}}_R^T \hat{g}$, where $\bar{\hat{g}}$ is the interpolated rotation field after superposition in the rigid rotation field. Equation (13) now yields $\bar{\hat{e}} = \underline{\underline{H}}(\bar{\hat{g}})\bar{\hat{g}}$. Properties 15 then lead to $\bar{\hat{e}} = \underline{\underline{D}}_R^T \underline{\underline{H}}(\hat{g}) \underline{\underline{D}}_R \underline{\underline{D}}_R^T \hat{g} = \underline{\underline{D}}_R^T \underline{\underline{H}}(\hat{g})\hat{g} = \underline{\underline{D}}_R^T \hat{e}$, which proves that algorithm 5 is tensorial.

3.2.2 Computation of curvature

The curvature quaternion is defined as $\hat{\kappa} = 2\underline{\underline{B}}^T(\hat{e})\hat{e}'$, where matrix $\underline{\underline{B}}$ is defined by eq. (53). Equation (11b) yields an approximation to the spatial derivative of the rotation field, denoted $\hat{g}'$. For the exact rotation field, $\hat{e}^T\hat{e} = 1$, which implies that $\hat{e}^T\hat{e}' = 0$. Due to the approximation inherent to the interpolation process, $\hat{g}^T\hat{g}' \neq 0$. Even after extraction of a unit quaternion $\hat{e}$ using algorithm 5, $\hat{e}^T\hat{g}' \neq 0$.

To remedy this situation, it seems appropriate to find the spatial derivative of the Euler parameters rotation field as $\hat{e}' = \hat{g}' - \alpha\,\hat{e}$, where α is an unknown coefficient. Imposing the condition $\hat{e}^T\hat{e}' = 0$ yields $\alpha = \hat{e}^T\hat{g}'$ and finally,

$$\hat{e}' = (\underline{\underline{I}} - \hat{e}\,\hat{e}^T)\hat{g}'. \tag{16}$$

The curvature quaternion now becomes

$$\hat{\kappa} = 2\underline{\underline{B}}^T(\hat{e})(\underline{\underline{I}} - \hat{e}\,\hat{e}^T)\hat{g}'. \tag{17}$$

It is easily verified that the scalar part of this quaternion vanishes, and its vector part reduces to

$$\underline{\kappa} = 2\left(-g_0'\underline{e} + e_0\underline{g}' + \tilde{e}\underline{g}'\right). \tag{18}$$

Note that the corrected spatial derivatives of the Euler parameters given by eq. (16) do not appear in these expressions. Hence, curvatures can be computed without evaluating these corrections.

Algorithm 6 (Curvature interpolation) *Interpolation of a curvature field defined by nodal Euler parameters, $\hat{e}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use algorithm 5 to obtain unit quaternion $\hat{e}$.* ***Step (2)****. Use interpolation formula, eq.* (11b)*, to obtain quaternion $\hat{g}'$.* ***Step (3)****. Use eq.* (18) *to find the interpolated curvature.*

Under a change of basis, algorithm 5 yields $\bar{\hat{e}} = \underline{\underline{D}}_R^T \hat{e}$ and eq. (11b) yields $\bar{\hat{g}}' = \underline{\underline{D}}_R^T \hat{g}'$. Equation (17) leads to $\bar{\hat{\kappa}} = 2\underline{\underline{B}}^T(\bar{\hat{e}})(\underline{\underline{I}} - \bar{\hat{e}}\bar{\hat{e}}^T)\bar{\hat{g}}'$ and note that $\underline{\underline{B}}^T(\bar{\hat{e}}) = \underline{\underline{B}}^T(\underline{\underline{D}}_R^T\hat{e}) = \underline{\underline{D}}_R^T\underline{\underline{B}}^T(\hat{e})\underline{\underline{D}}_R$. It now follows that $\bar{\hat{\kappa}} = 2\underline{\underline{D}}_R^T\underline{\underline{B}}^T(\hat{e})\underline{\underline{D}}_R(\underline{\underline{I}} - \underline{\underline{D}}_R^T\hat{e}\hat{e}^T\underline{\underline{D}}_R)\underline{\underline{D}}_R^T\hat{g}' = \underline{\underline{D}}_R^T 2\underline{\underline{B}}^T(\hat{e})(\underline{\underline{I}} - \hat{e}\hat{e}^T)\hat{g}' = \underline{\underline{D}}_R^T\underline{\kappa}$, which proves that the algorithm is tensorial.

3.3 Interpolation of rotation parameter vectors

Consider the following interpolation of the rotation field defined by discrete rotation parameter vectors [23] at N nodes

$$\underline{p}(x_1) = \sum_{i=1}^{N} h_i \underline{\hat{p}}_i, \tag{19a}$$

$$\underline{p}'(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J} \underline{\hat{p}}_i, \tag{19b}$$

where $\underline{\hat{p}}_i$ is the rotation parameter vector at node i, and h_i the shape functions.

Once rotation parameter vector $\underline{p}$ is obtained from eq. (19a), the corresponding orthogonal rotation tensor is obtained

$$\underline{\underline{R}} = \underline{\underline{I}} + \zeta_1(\phi)\ \tilde{p} + \zeta_2(\phi)\ \tilde{p}\tilde{p}, \tag{20}$$

where $\zeta_1(\phi)$ and $\zeta_2(\phi)$ are even functions of the rotation angle, ϕ, defined as $\zeta_1(\phi) = (\sin\phi)/p$ and $\zeta_2(\phi) = (1-\cos\phi)/p^2$, respectively.

Algorithm 7 (Finite rotation interpolation) *Interpolation of a rotation field defined by nodal rotation parameter vectors, $\underline{\hat{p}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use interpolation formula, eq.* (19a)*, to obtain rotation parameter vector $\underline{p}$.* ***Step (2)****. Use eq.* (20) *to obtain orthogonal tensor $\underline{\underline{R}}$.*

Under a change of basis, the nodal rotation parameter vectors now become $\underline{\bar{\hat{p}}}_i = \underline{\underline{R}}_R^T \underline{\hat{p}}_i = \underline{\hat{p}}_i^*$. The fact that the components of the rotation parameter vectors transform like first-order tensors under a change of basis operation is a fundamental property of all vectorial parameterizations of rotation [24]. It follows that $\underline{\bar{p}} = \underline{\underline{R}}_R^T \underline{p}$, and consequently, $\underline{\underline{\bar{R}}} = \underline{\underline{R}}_R^T \underline{\underline{R}}\, \underline{\underline{R}}_R$. Clearly, the interpolation strategy described by eq. (19a) is tensorial.

Because the interpolation strategy relies on a specific parameterization of rotation, it is not intrinsic. Indeed, if a given rotation field is parameterized using two different vectorial parameterizations of rotation, the two resulting interpolated rotation fields will differ slightly. In summary, interpolation strategy (19a) is tensorial for any vectorial parameterization of rotation, but is not intrinsic.

3.3.1 Computation of curvature

The curvature vector can be expressed in terms of the vectorial parameterization of rotation and of its spatial derivative as $\underline{\kappa} = \underline{\underline{H}}(\underline{p})\underline{p}'$, where $\underline{\underline{H}}(\underline{p})$ is the tangent tensor

$$\underline{\underline{H}}(\underline{p}) = \sigma_0(\phi)\ \underline{\underline{I}} + \sigma_1(\phi)\ \tilde{p} + \sigma_2(\phi)\ \tilde{p}\tilde{p}, \tag{21}$$

where $\sigma_0(\phi)$, $\sigma_1(\phi)$, and $\sigma_2(\phi)$ are even functions of the rotation angle, ϕ, defined as $\sigma_0(\phi) = 1/p'$, $\sigma_1(\phi) = (1-\cos\phi)/p^2$, and $\sigma_2(\phi) = (\sigma_0 - \zeta_1)/p^2$, respectively. Using eq. (19b) to find the spatial derivative of the rotation parameter vector field then yields the curvature field as

$$\underline{\kappa}(x_1) = \underline{\underline{H}}(\underline{p})\underline{p}'. \tag{22}$$

Algorithm 8 (Curvature interpolation) *Interpolation of a curvature field defined by nodal rotation parameter vectors, $\underline{\hat{p}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use interpolation formula* (19a) *to find the rotation parameter vector, $\underline{p}$.* ***Step (2)****. Use interpolation formula* (19b) *to obtain the spatial derivative of the rotation parameter vector, $\underline{p}'$.* ***Step (3)****. Use eq.* (22) *to find the interpolated curvature.*

Under a change of basis, algorithm 7 yields $\underline{\bar{p}} = \underline{\underline{R}}_R^T \underline{p}$ and eq. (19b) yields $\underline{\bar{p}}' = \underline{\underline{R}}_R^T \underline{p}'$. Equation (22) leads to $\underline{\bar{\kappa}} = \underline{\underline{H}}(\underline{\bar{p}})\underline{\bar{p}}'$. A fundamental property of the vectorial parameterization of rotation is that its tangent operator is a second-order tensor, *i.e.*, $\underline{\underline{H}}(\underline{\underline{R}}_R^T \underline{p}) = \underline{\underline{R}}_R^T \underline{\underline{H}}(\underline{p})\underline{\underline{R}}_R$. It now follows that $\underline{\bar{\kappa}} = \underline{\underline{R}}_R^T \underline{\underline{H}}(\underline{p})\underline{\underline{R}}_R \underline{\underline{R}}_R^T \underline{p}' = \underline{\underline{R}}_R^T \underline{\kappa}$, which proves that the algorithm is tensorial. Of course, the algorithm is not intrinsic because the tangent tensor appears explicitly in the expression for the curvature vector.

Crisfield and Jelenić [13] have underlined the lack of objectivity of curvature interpolation algorithms based on eqs. (19a) and (19b). The present developments, however, show that the approach is tensorial for any vectorial parameterization of rotation.

3.4 Interpolation of relative rotations

Crisfield and Jelenić [13] advocated the interpolation of the relative nodal rotations to achieve objective algorithms for strain interpolation. Let $\hat{\underline{\underline{R}}}_1$ and $\hat{\underline{\underline{R}}}_2$ denote the rotation tensors at nodes 1 and 2, respectively, resolved in inertial basis $\mathcal{B}$. The relative rotation tensor of node 2 with respect to node 1, denoted $\hat{\underline{\underline{R}}}_2^r$, is then $\hat{\underline{\underline{R}}}_2^r = \hat{\underline{\underline{R}}}_2 \hat{\underline{\underline{R}}}_1^T$. Resolving the components of this relative rotation tensor in the basis defined by the rotation at node 1 yields $\hat{\underline{\underline{R}}}_2^{r*} = \hat{\underline{\underline{R}}}_1^T \hat{\underline{\underline{R}}}_2$, where notation $(\cdot)^*$ indicates tensor components resolved in the basis defined by the rotation tensor at node 1, denoted basis $\mathcal{B}^*$. The relative nodal rotation tensors are now $\hat{\underline{\underline{R}}}_i^{r*} = \hat{\underline{\underline{R}}}_1^T \hat{\underline{\underline{R}}}_i$, $i = 1, 2, \ldots, N$, where by construction, $\hat{\underline{\underline{R}}}_1^{r*} = \underline{\underline{I}}$.

The algorithms presented in the previous sections could be reformulated starting from the relative nodal rotation tensors, relative nodal Euler parameters, or relative rotation parameter vectors. The latter choice is of particular interest and to facilitate the developments, it is convenient to introduce a compact notation for composition of rotation operations. To facilitate the writing of the algorithm, the following compact notation is adopted: $\underline{\underline{R}}(\underline{r}) = \underline{\underline{R}}(\underline{p})\underline{\underline{R}}(\underline{q}) \Leftrightarrow \underline{r} = \underline{p} \oplus \underline{q}$ and $\underline{\underline{R}}(\underline{r}) = \underline{\underline{R}}^T(\underline{p})\underline{\underline{R}}(\underline{q}) \Leftrightarrow \underline{r} = \underline{p}^- \oplus \underline{q}$.

Relative nodal rotation parameter vectors are interpolated to find

$$\underline{p}^{r*}(x_1) = \sum_{i=1}^{N} h_i \hat{\underline{p}}_i^{r*}, \tag{23a}$$

$$\underline{p}^{r*\prime}(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J} \hat{\underline{p}}_i^{r*}, \tag{23b}$$

where h_i are the shape functions.

Algorithm 9 (Finite rotation interpolation) *Interpolation of a rotation field defined by nodal rotation parameter vectors, $\hat{\underline{p}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1).*** *Compute the components of the relative nodal rotation parameter vectors resolved in basis $\mathcal{B}^*$, $\hat{\underline{p}}_i^{r*} = \hat{\underline{p}}_1^- \oplus \hat{\underline{p}}_i$, $i = 1, 2, \ldots, N$.* ***Step (2).*** *Use interpolation formula, eq.* (23a)*, to obtain the components of the relative rotation parameter vector, $\underline{p}^{r*}$, resolved in the same basis.* ***Step (3).*** *The components of the interpolated rotation parameter vector resolved in basis $\mathcal{B}$ are then $\underline{p}(x_1) = \hat{\underline{p}}_1 \oplus \underline{p}^{r*}(x_1)$.*

Note that the interpolation process takes place in the coordinate system defined by basis $\mathcal{B}^*$. Of course, any other local basis could be used such as, for instance, the basis defined by the rotation tensor at node 2. Algorithm 9 is otherwise identical to algorithm 7 based on the rotation parameter vectors themselves, and hence, algorithm 9 is objective and tensorial, but not intrinsic. Indeed, the superposition of an arbitrary rigid body motion would be purged by the interpolation process, leading to the same interpolated rotation.

3.4.1 Computation of curvature

The components of the curvature vector resolved in basis $\mathcal{B}^*$ are $\underline{\kappa}^* = \underline{\underline{H}}(\underline{p}^{r*})\underline{p}^{r*\prime}$, where $\underline{p}^{r*\prime}$ are the components of the spatial derivative of the rotation field given by eq. (23b). It then follows that $\underline{\kappa}^* = \underline{\underline{H}}(\hat{\underline{\underline{R}}}_1^T \underline{p}^r)\hat{\underline{\underline{R}}}_1^T \underline{p}^{r\prime} = \hat{\underline{\underline{R}}}_1^T \underline{\underline{H}}(\underline{p}^r)\underline{p}^{r\prime} = \hat{\underline{\underline{R}}}_1^T \underline{\kappa}$. The components of the curvature vector in basis $\mathcal{B}$ are then

$$\underline{\kappa}(x_1) = \hat{\underline{\underline{R}}}_1 \underline{\underline{H}}(\underline{p}^{r*})\underline{p}^{r*\prime}. \tag{24}$$

Algorithm 10 (Curvature interpolation) *Interpolation of a curvature field defined by nodal rotation parameter vectors, $\hat{\underline{p}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1).*** *Use algorithm 9 to obtain the relative*

rotation parameter vector $\underline{p}^{r*}$ *resolved in basis* $\mathcal{B}^*$. ***Step (2)***. *Use interpolation formula, eq.* (23b), *to obtain the spatial derivative of the relative rotation parameter vector* $\underline{p}^{r*\prime}$, *resolved in the same basis.* ***Step (3)***. *Use eq.* (24) *to find the interpolated curvature.*

This algorithm is objective and tensorial, but not intrinsic, because it is a particular case of algorithm 8.

3.5 Interpolation of rotation: numerical results

A rotation field is defined by the following Euler parameters, $e_0 = C_\phi$, $e_1 = S_\phi S_\theta C_\psi$, $e_2 = S_\phi S_\theta S_\psi$, and $e_3 = S_\phi C_\theta$. Note that the use of trigonometric functions guaranties the satisfaction of the normality condition. Angles ϕ, θ, and ψ are given functions of spatial variable $s \in [-1, +1]$, defined as $\phi = 0.8 \sin s$, $\theta = 2 \sin 0.8s + 0.6 \cos s$, and $\psi = \cos s - 1$. Figure 5 shows the rotation field versus spatial variable s as represented by its Euler parameters. Using the properties of Euler parameters [1], the components of the curvature vector are then found easily. The exact curvature field is depicted in fig. 6 that shows the components of the curvature vector, $\underline{\kappa}^T = \{\kappa_1, \kappa_2, \kappa_3\}$.

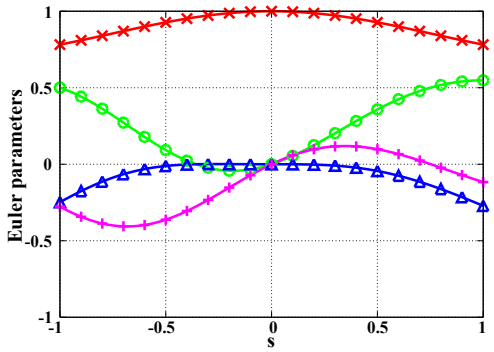

Figure 5: Rotation field versus s. Euler parameters e_0 ($\times$), e_1 ($\circ$), e_2 ($\triangle$), e_3 ($+$).

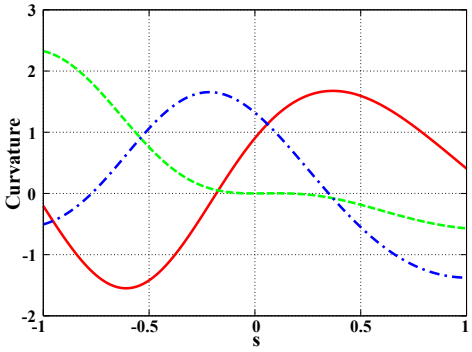

Figure 6: Curvature field versus s: κ_1 (solid line), κ_2 (dashed line), κ_3 (dashed-dotted line).

Interval $s \in [-1, +1]$ was divided into N_e elements of equal length, and within each element, the rotation field was approximated using linear, quadratic, cubic, and quartic shape functions, corresponding to elements of order $o_e = 1$, 2, 3, and 4, respectively. Next, the curvature field was approximated using the various algorithms presented in the previous sections.

To assess the accuracy of the proposed algorithms, the interpolated curvature, denoted $\underline{\kappa}_a$, was compared to its exact counterpart, denoted $\underline{\kappa}_e$. The curvature error measure was selected as

$$e_\kappa = \frac{1}{N_{gp}} \sum_{k=1}^{N_{gp}} \frac{\|\underline{\kappa}_a(s_k) - \underline{\kappa}_e(s_k)\|}{\|\underline{\kappa}_e(s_k)\|}. \tag{25}$$

The exact and approximate curvatures were computed at N_{gp} sampling points denoted s_k. Within each element, $o_e + 1$ sampling points were selected to coincide with the location of the Gauss-Legendre quadrature points within the element.

Figure 7 shows the curvature error measure defined by eq. (25) versus the number of elements, $N_e \in [2, 256]$, on a logarithmic plot, when algorithm 3 is used to compute the curvature, *i.e.*, when the polar decomposition is used to extract an orthogonal rotation tensor from the interpolated matrix. Results are shown for linear, quadratic, cubic, and quartic shape functions; in each case, a linear regression was performed and is also shown on the figure. Figure 7 also shows the corresponding results when algorithm 4 is used to compute the curvature, *i.e.*, when Cayley's decomposition is used to extract an orthogonal rotation tensor from the interpolated rotation tensor. These results show that the two algorithms yield very similar results.

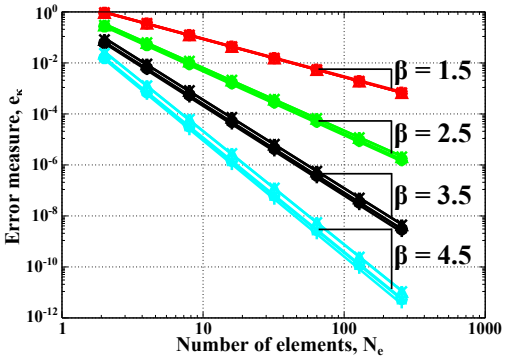

Figure 7: Curvature error measure (25) versus number of elements when using algorithm 3 ($\times$), 4 ($\triangle$), 8 (*), 10 (+), 6($\circ$).

Next, curvatures were computed using interpolation of the rotation parameter vector; the Wiener-Milenković parameters [25, 26, 1] were selected for this study. Figure 7 shows the numerical results for this interpolation strategy when using the rotation parameter vector and the *relative* rotation parameter vector, corresponding to algorithms 8 and 10, respectively. Here again, the two algorithms yield very similar results. Finally, curvatures were computed using interpolation of the Euler parameters, as described by algorithm 6 and the corresponding results are shown in fig. 7. Linear regressions of the numerical data were performed in all cases and are shown as dashed lines in fig. 7. For all the rotation interpolation algorithms presented here, relationship (4) applied, as indicated on the figure.

When using linear shape functions, it is remarkable to note that the results of all algorithms presented here are nearly indistinguishable on the logarithmic scale of fig. 7. As the order of the shape functions increases, a trend appears in the results. The interpolation based on the relative Wiener-Milenković parameters is the most accurate, that based on the rotation tensor (with use of either polar or Cayley's decomposition to extract the orthogonal rotation tensor) is the least accurate, while that based on the Euler parameters falls between the other two. Although the interpolation using rotation parameter vectors and relative rotation parameter vectors are very similar, interpolation of relative rotation parameter vectors is slightly more accurate.

4 Interpolation of motion fields

Section 3 has presented a review of the issues associated with the interpolation of rotation fields. This section addresses the issues that arise when interpolating general motion fields represented by a motion tensor,

$$\underline{\underline{\mathcal{C}}} = \begin{bmatrix} \underline{\underline{R}} & \tilde{u}\underline{\underline{R}} \\ \underline{\underline{0}} & \underline{\underline{R}} \end{bmatrix}, \tag{26}$$

were $\underline{\underline{R}}$ and $\underline{u}$ are the rotation of a frame and displacement of its origin, respectively. The approaches presented in section 3 can be extended to the case of motions fields, although additional issues must be resolved.

4.1 Interpolation of the motion tensor

Consider the following interpolation of the motion field defined by discrete motion tensors at N nodes

$$\underline{\underline{\mathcal{T}}}(x_1) = \begin{bmatrix} \underline{\underline{T}} & \underline{\underline{W}} \\ \underline{\underline{0}} & \underline{\underline{T}} \end{bmatrix} = \sum_{i=1}^{N} h_i \underline{\underline{\hat{\mathcal{C}}}}_i, \tag{27a}$$

$$\underline{\underline{\mathcal{T}}}'(x_1) = \begin{bmatrix} \underline{\underline{T}}' & \underline{\underline{W}}' \\ \underline{\underline{0}} & \underline{\underline{T}}' \end{bmatrix} = \sum_{i=1}^{N} \frac{h_i^+}{J} \underline{\underline{\hat{\mathcal{C}}}}_i, \tag{27b}$$

where $\underline{\underline{\hat{\mathcal{C}}}}_i$ is the motion tensor [1] at node i, and h_i the shape functions.

Although eqs. (27a) and (27b) give reasonable approximations for the interpolated motion field and its derivative, respectively, they suffer a major drawback: the interpolated quantities are not motion tensors. In particular, except at the nodes, the interpolated rotation matrix is not orthogonal, *i.e.*, $\underline{\underline{T}}\,\underline{\underline{T}}^T \neq \underline{\underline{I}}$. These issues are identical to those that arose in the interpolation of rotation fields, as discussed in section 3.1. Further issues arise with the present interpolation of motion. In particular, except at the nodes, matrix $\underline{\underline{W}}$ is no longer of the form $\tilde{u}\underline{\underline{R}}$, as required for a motion tensor.

4.1.1 Extraction of the motion tensor through generalized polar decomposition

In section 3.1.1, an orthogonal rotation tensor was extracted from its interpolated counterpart using the polar decomposition theorem or Cayley's decomposition. In the more general case under present scrutiny, the polar decomposition theorem must be generalized, see appendix B, to extract a motion tensor from its interpolated counterpart.

Following a reasoning similar to that used section 3.1.1, the interpolated motion tensor is decomposed as $\underline{\underline{\mathcal{T}}} = \underline{\underline{\mathcal{C}}}\,\underline{\underline{\mathcal{U}}}$ using the generalized polar decomposition theorem, eq (59), and minimization of the Frobenius norm then implies that motion tensor $\underline{\underline{\mathcal{C}}}$ closely approximates matrix $\underline{\underline{\mathcal{T}}}$ when $\underline{\underline{\mathcal{U}}} = \underline{\underline{\mathcal{I}}}$.

Algorithm 11 (Finite motion interpolation) *Interpolation of a motion field defined by nodal motion tensors,* $\underline{\underline{\hat{\mathcal{C}}}}_i$, $i = 1, 2, \ldots, N$. ***Step (1)****. Use interpolation formula, eq.* (27a)*, to obtain matrix* $\underline{\underline{\mathcal{T}}}$. ***Step (2)****. Use the generalized polar decomposition theorem to extract motion tensor* $\underline{\underline{\mathcal{C}}}$.

It is easy to prove that algorithm 11 is tensorial. Under a change of frame, the nodal motions become $\underline{\underline{\hat{\mathcal{C}}}}_i^* = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\hat{\mathcal{C}}}}_i\underline{\underline{\mathcal{C}}}_R$, where notation $(\cdot)^*$ indicates a change of frame operation. Interpolation equation (27a) then yields $\underline{\underline{\bar{\mathcal{T}}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{T}}}\,\underline{\underline{\mathcal{C}}}_R$, where notation $\bar{(\cdot)}$ indicates interpolated quantities after the change of frame. The motion tensors extracted from matrices $\underline{\underline{\mathcal{T}}}$ and $\underline{\underline{\bar{\mathcal{T}}}}$ using the generalized polar decomposition are $\underline{\underline{\mathcal{C}}}$ and $\underline{\underline{\bar{\mathcal{C}}}}$, respectively. Because the generalized polar decomposition is unique, $\underline{\underline{\bar{\mathcal{C}}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{C}}}\,\underline{\underline{\mathcal{C}}}_R = \underline{\underline{\mathcal{C}}}^*$. Clearly, under a change of frame, algorithm 11 yields an interpolated motion resolved in the new frame. Hence, algorithm 11 is tensorial.

It is easy to prove that algorithm 11 is also objective. Assume that a rigid motion characterized by rigid motion tensor $\underline{\underline{\mathcal{C}}}_R$ is superposed to the motion field. The nodal motions now become $\underline{\underline{\hat{\mathcal{C}}}}_i^+ = \underline{\underline{\mathcal{C}}}_R\underline{\underline{\hat{\mathcal{C}}}}_i$, where notation $(\cdot)^+$ indicates the composed motion. Interpolation equation (27a) then yields $\underline{\underline{\bar{\mathcal{T}}}} = \underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{T}}}$. The motion tensors extracted from matrices $\underline{\underline{\mathcal{T}}}$ and $\underline{\underline{\bar{\mathcal{T}}}}$ using the generalized polar decomposition are $\underline{\underline{\mathcal{C}}}$ and $\underline{\underline{\bar{\mathcal{C}}}}$, respectively. Because the generalized polar decomposition is unique,

$\underline{\underline{\bar{\mathcal{C}}}} = \underline{\underline{\mathcal{C}}}_R \underline{\underline{\mathcal{C}}}$. Clearly, algorithm 11 yields the correct composition of motion for the interpolated field. Hence, algorithm 11 is objective.

4.1.2 Extraction of the motion tensor through Cayley's decomposition

In this section, Cayley's decomposition will be used to extract a motion tensor from matrix $\underline{\underline{\mathcal{T}}}$ obtained from interpolation formula (27a). Let $\underline{\underline{\mathcal{C}}}$ be a motion tensor; Cayley's decomposition [1] states that $\tilde{\mathcal{A}} = (\underline{\underline{\mathcal{C}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\mathcal{C}}} + \underline{\underline{\mathcal{I}}})^{-1}$, where $\underline{\mathcal{A}}$ are Cayley's motion parameters of motion tensor $\underline{\underline{\mathcal{C}}}$ and $\tilde{\mathcal{A}}$ indicates a generalized skew-symmetric matrix as defined by eq. (56). On the other hand, if $\underline{\underline{\mathcal{T}}}$ is an arbitrary matrix, $\underline{\underline{\mathcal{W}}} = (\underline{\underline{\mathcal{T}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\mathcal{T}}} + \underline{\underline{\mathcal{I}}})^{-1}$, where matrix $\underline{\underline{\mathcal{W}}}$ is no longer a generalized skew-symmetric matrix.

The following minimization problem is defined: $\min_{\underline{\mathcal{A}}} \|\tilde{\mathcal{A}} - \underline{\underline{\mathcal{W}}}\|_F$, where matrices $\tilde{\mathcal{A}}$ and $\underline{\underline{\mathcal{W}}}$ are of the following form

$$\tilde{\mathcal{A}} = \begin{bmatrix} \tilde{b} & \tilde{a} \\ \underline{0} & \tilde{b} \end{bmatrix}, \quad \underline{\underline{\mathcal{W}}} = \begin{bmatrix} \underline{\underline{B}} & \underline{A} \\ \underline{0} & \underline{\underline{B}} \end{bmatrix}. \tag{28}$$

Matrices $\underline{A}$ and $\underline{\underline{B}}$ are arbitrary, non-skew-symmetric matrices. Here again, the Frobenius norm is selected and tedious algebra yields the solution of the minimization problem as

$$\tilde{\mathcal{A}} = \begin{bmatrix} \mathrm{skew}(\underline{\underline{B}}) & \mathrm{skew}(\underline{A}) \\ \underline{0} & \mathrm{skew}(\underline{\underline{B}}) \end{bmatrix} = \mathcal{S}\mathrm{kew}(\underline{\underline{\mathcal{W}}}), \tag{29}$$

where the second equality defines operator $\mathcal{S}$kew, which can be shown to enjoy the following property

$$\mathcal{S}\mathrm{kew}(\underline{\underline{\mathcal{C}}}\,\underline{\underline{\mathcal{W}}}\,\underline{\underline{\mathcal{C}}}^{-1}) = \underline{\underline{\mathcal{C}}}\;\mathcal{S}\mathrm{kew}(\underline{\underline{\mathcal{W}}})\;\underline{\underline{\mathcal{C}}}^{-1}, \tag{30}$$

where $\underline{\underline{\mathcal{C}}}$ is an arbitrary motion tensor.

The above result is now applied to the problem at hand. Matrix $\underline{\underline{\mathcal{W}}} = (\underline{\underline{\mathcal{T}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\mathcal{T}}} + \underline{\underline{\mathcal{I}}})^{-1}$ is not a generalized skew-symmetric matrix but it is closely approximated by generalized skew-symmetric matrix $\tilde{\mathcal{A}} = \mathcal{S}\mathrm{kew}[(\underline{\underline{\mathcal{T}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\mathcal{T}}} + \underline{\underline{\mathcal{I}}})^{-1}]$, leading the following expression for Cayley's motion parameters

$$\underline{\mathcal{A}} = \mathcal{A}\mathrm{xial}[(\underline{\underline{\mathcal{T}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\mathcal{T}}} + \underline{\underline{\mathcal{I}}})^{-1}]. \tag{31}$$

Motion tensor $\underline{\underline{\mathcal{C}}}$ is then obtained through the following expression

$$\underline{\underline{\mathcal{C}}} = (\underline{\underline{\mathcal{I}}} - \tilde{\mathcal{A}})^{-1}(\underline{\underline{\mathcal{I}}} + \tilde{\mathcal{A}}). \tag{32}$$

Algorithm 12 (Finite motion interpolation) *Interpolation of a motion field defined by nodal motion tensors,* $\underline{\underline{\hat{\mathcal{C}}}}_i$, $i = 1, 2, \ldots, N$. ***Step (1)***. *Use interpolation formula, eq.* (27a)*, to obtain matrix* $\underline{\underline{\mathcal{T}}}$. ***Step (2)***. *Use eq.* (31) *to find Cayley's motion parameters,* $\underline{\mathcal{A}}$. ***Step (3)***. *Use eq.* (32) *to find the interpolated motion tensor,* $\underline{\underline{\mathcal{C}}}$.

Under a change of frame, the nodal motion tensors now become $\underline{\underline{\hat{\mathcal{C}}}}_i^* = \underline{\underline{\mathcal{C}}}_R^{-1} \underline{\underline{\hat{\mathcal{C}}}}_i \underline{\underline{\mathcal{C}}}_R$. Let $\underline{\underline{\bar{\mathcal{T}}}}$ be the interpolated motion field after superposition in the rigid motion field and $\underline{\bar{\mathcal{A}}}$ Cayley's motion parameters obtained from eq. (31). Note that eq. (31) is tensorial in nature, *i.e.*, $\underline{\bar{\mathcal{A}}} = \mathcal{A}\mathrm{xial}[(\underline{\underline{\bar{\mathcal{T}}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\bar{\mathcal{T}}}} + \underline{\underline{\mathcal{I}}})^{-1}] = \mathcal{A}\mathrm{xial}[\underline{\underline{\mathcal{C}}}_R^{-1}(\underline{\underline{\mathcal{T}}} - \underline{\underline{\mathcal{I}}})\underline{\underline{\mathcal{C}}}_R \underline{\underline{\mathcal{C}}}_R^{-1}(\underline{\underline{\mathcal{T}}} + \underline{\underline{\mathcal{I}}})^{-1}\underline{\underline{\mathcal{C}}}_R] = \mathcal{A}\mathrm{xial}[\underline{\underline{\mathcal{C}}}_R^{-1}(\underline{\underline{\mathcal{T}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\mathcal{T}}} + \underline{\underline{\mathcal{I}}})^{-1}\underline{\underline{\mathcal{C}}}_R] = \underline{\underline{\mathcal{C}}}_R^{-1}\mathcal{A}\mathrm{xial}[(\underline{\underline{\mathcal{T}}} - \underline{\underline{\mathcal{I}}})(\underline{\underline{\mathcal{T}}} + \underline{\underline{\mathcal{I}}})^{-1}] = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{A}}$. Because Cayley's motion parameters form a vectorial parameterization of motion, eq. (32) yields $\underline{\underline{\bar{\mathcal{C}}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{C}}}\,\underline{\underline{\mathcal{C}}}_R = \underline{\underline{\mathcal{C}}}^*$, which proves that algorithm 12 is tensorial. This algorithm is based on the identification of Cayley's motion parameters, a specific parameterization of motion, and hence, is not intrinsic.

4.1.3 Computation of curvature

The curvature vector is defined as $\tilde{\mathcal{E}} = \underline{\underline{\mathcal{C}}}'\underline{\underline{\mathcal{C}}}^{-1}$ and its evaluation requires the knowledge of the spatial derivative of the motion field. Equation (27b) yields an approximation to the spatial derivative of the motion field, denoted $\underline{\underline{\mathcal{T}}}'$. For the exact motion field, the product $\underline{\underline{\mathcal{C}}}'\underline{\underline{\mathcal{C}}}^{-1}$ yields a generalized skew-symmetric matrix, $\tilde{\mathcal{E}}$. Due to the approximation inherent to the interpolation process, $\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{T}}}^{-1} \neq \tilde{\mathcal{E}}$. Even after extraction of a motion tensor using algorithm 12, $\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}^{-1} \neq \tilde{\mathcal{E}}$.

Matrix $\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}^{-1}$ is not a generalized skew-symmetric matrix, but in view of eq. (29), is approximated by generalized skew-symmetric matrix $\tilde{\mathcal{E}} = \mathcal{S}\text{kew}(\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}^{-1})$ closely, leading to the following expression for the curvature vector

$$\tilde{\mathcal{E}} = \mathcal{S}\text{kew}(\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}^{-1}). \tag{33}$$

Algorithm 13 (Curvature interpolation) *Interpolation of a curvature field defined by nodal motion tensors, $\underline{\underline{\hat{\mathcal{C}}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use algorithm 11 to obtain motion tensor $\underline{\underline{\mathcal{C}}}$.* ***Step (2)****. Use interpolation formula, eq.* (27b)*, to obtain matrix $\underline{\underline{\mathcal{T}}}'$.* ***Step (3)****. Use eq.* (33) *to find the interpolated curvature.*

Algorithm 14 (Curvature interpolation) *Interpolation of a curvature field defined by nodal motion tensors, $\underline{\underline{\hat{\mathcal{C}}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use algorithm 12 to obtain motion tensor $\underline{\underline{\mathcal{C}}}$.* ***Step (2)****. Use interpolation formula, eq.* (27b)*, to obtain matrix $\underline{\underline{\mathcal{T}}}'$.* ***Step (3)****. Use eq.* (33) *to find the interpolated curvature.*

Under a change of frame, algorithms 13 and 14 both yield $\underline{\underline{\bar{\mathcal{C}}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{C}}}\,\underline{\underline{\mathcal{C}}}_R$, and under the same operation, eq. (27b) yields $\underline{\underline{\bar{\mathcal{T}}}}' = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}_R$. Equation (33) then leads to $\bar{\tilde{\mathcal{E}}} = \mathcal{S}\text{kew}(\underline{\underline{\bar{\mathcal{T}}}}'\underline{\underline{\bar{\mathcal{C}}}}^{-1}) = \mathcal{S}\text{kew}(\underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{C}}}^{-1}\underline{\underline{\mathcal{C}}}_R) = \underline{\underline{\mathcal{C}}}_R^{-1}\mathcal{S}\text{kew}(\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}^{-1})\underline{\underline{\mathcal{C}}}_R$, where the last equality follows from property (30). The algorithm is tensorial because $\bar{\tilde{\mathcal{E}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\tilde{\mathcal{E}}\underline{\underline{\mathcal{C}}}_R$, or $\underline{\bar{\mathcal{E}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{E}}$.

It is easy to prove that algorithms 13 is also objective. Under a composition of motion, algorithm 11 yields $\underline{\underline{\bar{\mathcal{C}}}} = \underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{C}}}$, and under the same operation, eq. (27b) yields $\underline{\underline{\bar{\mathcal{T}}}}' = \underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{T}}}'$. Equation (33) then leads to $\bar{\tilde{\mathcal{E}}} = \mathcal{S}\text{kew}(\underline{\underline{\bar{\mathcal{T}}}}'\underline{\underline{\bar{\mathcal{C}}}}^{-1}) = \mathcal{S}\text{kew}(\underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{T}}}'\underline{\underline{\mathcal{C}}}^{-1}\underline{\underline{\mathcal{C}}}_R^{-1}) = \underline{\underline{\mathcal{C}}}_R\tilde{\mathcal{E}}\underline{\underline{\mathcal{C}}}_R^{-1}$, or $\underline{\bar{\mathcal{E}}} = \underline{\underline{\mathcal{C}}}_R\underline{\mathcal{E}}$. This last equation can be recast as $(\underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{C}}})^{-1}\underline{\bar{\mathcal{E}}} = \underline{\underline{\mathcal{C}}}^{-1}\underline{\mathcal{E}}$, which proves that the algorithm is objective because the components of strain field resolved in the material frame, $\underline{\underline{\mathcal{C}}}^{-1}\underline{\mathcal{E}}$, are identical to their counterpart after the addition of the rigid body motion, $(\underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{C}}})^{-1}\underline{\bar{\mathcal{E}}}$. Note that algorithm 14 is not objective because algorithm 12, which it is based on, is not.

4.2 Interpolation Euler motion parameters

Consider the following interpolation of the motion field defined by discrete Euler motion parameters at N nodes

$$\check{h}(x_1) = \sum_{i=1}^{N} h_i \check{b}_i, \tag{34a}$$

$$\check{h}'(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J} \check{b}_i, \tag{34b}$$

where $\check{b}_i$ are the Euler motions parameters at node i, and h_i the shape functions.

The Euler motion parameters form bi-quaternions [1], denoted

$$\check{b}_i = \begin{Bmatrix} \hat{q}_i \\ \hat{e}_i \end{Bmatrix}, \quad \check{h} = \begin{Bmatrix} \hat{p} \\ \hat{g} \end{Bmatrix}. \tag{35}$$

The nodal Euler parameters, $\hat{e}_i$, are unit quaternions and nodal quaternion $\hat{q}_i$ are orthogonal to the Euler parameters, *i.e.*, $\hat{q}_i^T\hat{e}_i = 0$. The corresponding interpolated quantities are denoted $\hat{p}$ and $\hat{g}$, but due to the interpolation process, $\hat{g}$ is not a unit quaternion and $\hat{p}^T\hat{g} \neq 0$. Although eqs. (34a) and (34b) give reasonable approximations for the interpolated motion field and its derivative, respectively, they suffer a major drawback: the interpolated Euler motion parameters no longer satisfy the orthonormality conditions.

4.2.1 Extraction of Euler motion parameters through orthonormalization

To remedy this situation, Euler motion parameters are extracted from the interpolated quantities using the following relationship

$$\check{b} = \begin{Bmatrix} \hat{q} \\ \hat{e} \end{Bmatrix} = \begin{bmatrix} \beta\underline{\underline{I}} & \alpha\underline{\underline{I}} \\ \underline{\underline{0}} & \beta\underline{\underline{I}} \end{bmatrix} \begin{Bmatrix} \hat{p} \\ \hat{g} \end{Bmatrix}, \tag{36}$$

where coefficient α and β are unknown scalars. Imposing the condition that $\hat{e}$ is a unit quaternion yields $\beta = 1/\sqrt{\hat{g}^T\hat{g}}$ and $\hat{e} = \hat{g}/\sqrt{\hat{g}^T\hat{g}}$. This solution is identical to that found when dealing with the interpolation of Euler parameters, see eq. (13), which was derived from minimization problem (12). Next, orthogonality condition $\hat{e}^T\hat{q} = 0$ is enforced, leading to $\alpha = -(\hat{g}^T\hat{p})/(\sqrt{\hat{g}^T\hat{g}})^3$. Introducing these results in eq. (36) then yields

$$\check{b} = \underline{\underline{\mathbb{H}}}(\check{h})\check{h}. \tag{37}$$

Tensor $\underline{\underline{\mathbb{H}}}$, of size 8×8, is defined as

$$\underline{\underline{\mathbb{H}}}(\check{h}) = \frac{1}{\sqrt{\hat{g}^T\hat{g}}} \begin{bmatrix} \underline{\underline{I}} & -\dfrac{\hat{g}^T\hat{p}}{\hat{g}^T\hat{g}}\underline{\underline{I}} \\ \underline{\underline{0}} & \underline{\underline{I}} \end{bmatrix}, \tag{38}$$

and the following properties are easily verified

$$\underline{\underline{\mathbb{H}}}(\underline{\underline{\mathbb{C}}}\,\check{h}) = \underline{\underline{\mathbb{H}}}(\check{h}), \quad \underline{\underline{\mathbb{H}}}(\check{h}) = \underline{\underline{\mathbb{C}}}^{-1}\underline{\underline{\mathbb{H}}}(\check{h})\underline{\underline{\mathbb{C}}}, \tag{39}$$

where $\underline{\underline{\mathbb{C}}}$ is an arbitrary motion tensor expressed by eq. (54).

Algorithm 15 (Finite motion interpolation) *Interpolation of a motion field defined by nodal Euler motion parameters, $\check{b}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use interpolation formula, eq.* (34a)*, to obtain bi-quaternion $\check{h}$.* ***Step (2)****. Use eq.* (37) *to extract Euler motion parameters $\hat{b}$ from bi-quaternion $\check{h}$.*

Under a change of frame, the nodal Euler motion parameters become $\bar{\check{b}}_i = \underline{\underline{\mathbb{C}}}_R^{-1}\check{b}_i$. A cursory look at eq. (34a) reveals that interpolation of these nodal Euler motion parameters yields $\bar{\check{h}} = \underline{\underline{\mathbb{C}}}_R^{-1}\check{h}$, where $\bar{\check{h}}$ is the bi-quaternion resulting from the interpolation of the motion field resolved in the new frame. Equation (37) now becomes $\bar{\check{b}} = \underline{\underline{\mathbb{H}}}(\bar{\check{h}})\bar{\check{h}}$ and properties (39) lead to $\bar{\check{b}} = \underline{\underline{\mathbb{C}}}_R^{-1}\underline{\underline{\mathbb{H}}}(\check{h})\underline{\underline{\mathbb{C}}}_R\underline{\underline{\mathbb{C}}}_R^{-1}\check{h} = \underline{\underline{\mathbb{C}}}_R^{-1}\underline{\underline{\mathbb{H}}}(\check{h})\check{h} = \underline{\underline{\mathbb{C}}}_R^{-1}\check{b}$, which proves that algorithm 15 is tensorial.

4.2.2 Computation of curvature

The curvature bi-quaternion is defined as $\check{\kappa} = 2\underline{\underline{\bar{\mathbb{B}}}}^T(\check{b})\check{b}'$, where matrix $\underline{\underline{\bar{\mathbb{B}}}}$ is defined by eq. (55). Its evaluation requires the knowledge of the spatial derivative of the motion field. Equation (34b) yields an approximation to the spatial derivative of the motion field, denoted $\check{h}'$. For the exact motion field, $\hat{e}^T\hat{e} = 1$, which implies that $\hat{e}^T\hat{e}' = 0$, and $\hat{e}^T\hat{q} = 0$, which implies that $\hat{e}^T\hat{q}' + \hat{q}^T\hat{e}' = 0$. Due

to the approximation inherent to the interpolation process, these conditions are not satisfied by the interpolated quantities.

To remedy this situation, it seems appropriate to find the spatial derivative of the Euler motion parameters rotation field as $\hat{q}' = \hat{p}' - \beta\hat{q} - \alpha\hat{e}$ and $\hat{e}' = \hat{g}' - \beta\hat{e}$, where α and β are unknown coefficients. Imposing the condition $\hat{e}^T\hat{e}' = 0$ yields $\beta = \hat{e}^T\hat{g}'$ and finally, $\hat{e}' = (\underline{\underline{I}} - \hat{e}\,\hat{e}^T)\hat{g}'$. This correction is identical to that found for Euler parameters, see eq. (16). Next, imposing condition $\hat{e}^T\hat{q}' + \hat{q}^T\hat{e}' = 0$ yields $\alpha = \hat{e}^T\hat{p}' + \hat{q}^T\hat{g}'$, and finally, $\hat{q}' = (\underline{\underline{I}} - \hat{e}\,\hat{e}^T)\hat{p}' - (\hat{q}\,\hat{e}^T + \hat{e}\,\hat{q}^T)\hat{g}'$. Collecting these results yields

$$\check{b}' = \underline{\underline{\mathbb{G}}}(\check{b})\check{h}', \tag{40}$$

where

$$\underline{\underline{\mathbb{G}}}(\check{b}) = \begin{bmatrix} \underline{\underline{I}} - \hat{e}\,\hat{e}^T & -(\hat{q}\,\hat{e}^T + \hat{e}\,\hat{q}^T) \\ \underline{\underline{0}} & \underline{\underline{I}} - \hat{e}\,\hat{e}^T \end{bmatrix}. \tag{41}$$

This matrix can be shown to satisfy the following identity

$$\underline{\underline{\mathbb{G}}}(\underline{\underline{\mathbb{C}}}\,\check{b}) = \underline{\underline{\mathbb{C}}}^{-1}\underline{\underline{\mathbb{G}}}(\check{b})\underline{\underline{\mathbb{C}}}, \tag{42}$$

for any motion tensor $\underline{\underline{\mathbb{C}}}$.

The curvature bi-quaternion now becomes

$$\check{\kappa} = 2\underline{\underline{\mathbb{B}}}^T(\check{b})\underline{\underline{\mathbb{G}}}(\check{b})\check{h}'. \tag{43}$$

It is easily verified that the scalar parts of both stretch and curvature quaternions vanish and the vector parts of these two quaternions reduce to

$$\underline{\epsilon} = 2\left(-p_0'\underline{e} + e_0\underline{p}' + \tilde{e}\underline{p}'\right) - 2\left(-g_0'\underline{q} + q_0\underline{g}' + \tilde{q}\underline{g}'\right), \tag{44a}$$

$$\underline{\kappa} = 2\left(-g_0'\underline{e} + e_0\underline{g}' + \tilde{e}\underline{g}'\right). \tag{44b}$$

Note that the corrected spatial derivatives of the Euler motion parameters given by eq. (40) do not appear in these expressions. Hence, curvatures can be computed without evaluating these corrections.

Algorithm 16 (Curvature interpolation) *Interpolation of a curvature field defined by nodal Euler motion parameters,* $\check{b}_i$, $i = 1, 2, \ldots, N$. ***Step (1)****. Use algorithm 15 to obtain Euler motion parameter* $\check{b}$. ***Step (2)****. Use interpolation formula, eq.* (34b)*, to obtain bi-quaternion* $\check{h}'$. ***Step (3)****. Use eqs.* (44) *to find the interpolated stretch and curvature vectors.*

Under a change of frame, algorithm 15 yields $\bar{\check{b}} = \underline{\underline{\mathbb{C}}}_R^{-1}\check{b}$ and eq. (34b) yields $\bar{\check{h}}' = \underline{\underline{\mathbb{C}}}_R^{-1}\check{h}'$. Equation (43) leads to $\bar{\check{\kappa}} = 2\underline{\underline{\mathbb{B}}}^T(\bar{\check{b}})\underline{\underline{\mathbb{G}}}(\bar{\check{b}})\bar{\check{h}}'$; note that $\underline{\underline{\mathbb{B}}}^T(\bar{\check{b}}) = \underline{\underline{\mathbb{C}}}_R^{-1}\underline{\underline{\mathbb{B}}}^T(\check{b})\underline{\underline{\mathbb{C}}}_R$ and in view of eq. (42), $\underline{\underline{\mathbb{G}}}(\bar{\check{b}}) = \underline{\underline{\mathbb{C}}}_R^{-1}\underline{\underline{\mathbb{G}}}(\check{b})\underline{\underline{\mathbb{C}}}_R$. It now follows that $\bar{\check{\kappa}} = 2\underline{\underline{\mathbb{C}}}_R^{-1}\underline{\underline{\mathbb{B}}}^T(\check{b})\underline{\underline{\mathbb{C}}}_R\underline{\underline{\mathbb{C}}}_R^{-1}\underline{\underline{\mathbb{G}}}(\check{b})\underline{\underline{\mathbb{C}}}_R\underline{\underline{\mathbb{C}}}_R^{-1}\check{h}' = 2\underline{\underline{\mathbb{C}}}_R^{-1}\underline{\underline{\mathbb{B}}}^T(\check{b})\underline{\underline{\mathbb{G}}}(\check{b})\check{h}' = \underline{\underline{\mathbb{C}}}_R^{-1}\check{\kappa}$, which proves that the algorithm is tensorial.

4.3 Interpolation of motion parameter vectors

Consider the following interpolation of the motion field defined by discrete motion parameter vectors [27] at N nodes

$$\underline{\mathcal{P}}(x_1) = \sum_{i=1}^{N} h_i\underline{\hat{\mathcal{P}}}_i, \tag{45a}$$

$$\underline{\mathcal{P}}'(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J}\underline{\hat{\mathcal{P}}}_i, \tag{45b}$$

where $\underline{\hat{\mathcal{P}}}_i$ is the motion parameter vector at node i, and h_i the shape functions.

Once motion parameter vector $\underline{\mathcal{P}}$ is obtained from eq. (45a), the corresponding orthogonal rotation tensor is obtained

$$\underline{\underline{\mathcal{C}}}(\underline{\mathcal{P}}) = \underline{\underline{\mathcal{I}}} + \underline{\underline{\mathcal{Z}}}(\bar{\zeta}_1, \zeta_1)\tilde{\mathcal{P}} + \underline{\underline{\mathcal{Z}}}(\bar{\zeta}_2, \zeta_2)\tilde{\mathcal{P}}\tilde{\mathcal{P}}, \tag{46}$$

where $\bar{\zeta}_1(\phi)$ and $\bar{\zeta}_2(\phi)$ are even functions of the rotation angle, ϕ, defined as $\bar{\zeta}_1 = \varrho(\varphi_2 - \varphi_0\zeta_2)$ and $\bar{\zeta}_2 = \varrho(\varphi_2\zeta_1 - \zeta_2^2)$, respectively. Matrix $\underline{\underline{\mathcal{Z}}}$ is defined by eq. (57).

Algorithm 17 (Finite motion interpolation) *Interpolation of a motion field defined by nodal motion parameter vectors, $\underline{\hat{\mathcal{P}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use interpolation formula, eq.* (45a)*, to obtain motion parameter vector $\underline{\mathcal{P}}$.* ***Step (2)****. Use eq.* (46) *to obtain motion tensor $\underline{\underline{\mathcal{C}}}$.*

Under a change of frame, the nodal motion parameter vectors become $\underline{\bar{\hat{\mathcal{P}}}}_i = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\hat{\mathcal{P}}}_i = \underline{\hat{\mathcal{P}}}_i^*$. The fact that the components of the motion parameter vectors transform like first-order tensors under a change of frame operation is a fundamental property of all vectorial parameterizations of motion [24]. It follows that $\underline{\bar{\mathcal{P}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{P}}$, and consequently, $\underline{\underline{\bar{\mathcal{C}}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{C}}}\,\underline{\underline{\mathcal{C}}}_R$. Clearly, the interpolation strategy described by eq. (45a) is tensorial.

4.3.1 Computation of curvature

The curvature vector can be expressed in terms of the vectorial parameterization of motion and of its spatial derivative as $\underline{\mathcal{E}} = \underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}})\underline{\mathcal{P}}'$, where $\underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}})$ is the tangent tensor

$$\underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}}) = \underline{\underline{\mathcal{Z}}}(\bar{\sigma}_0, \sigma_0) + \underline{\underline{\mathcal{Z}}}(\bar{\zeta}_2, \zeta_2)\tilde{\mathcal{P}} + \underline{\underline{\mathcal{Z}}}(\bar{\sigma}_2, \sigma_2)\tilde{\mathcal{P}}\tilde{\mathcal{P}}, \tag{47}$$

where $\bar{\sigma}_0 = \varrho\varphi_0'/(pp')$, $p^2\bar{\sigma}_2 = \bar{\sigma}_0 - 2\varrho\varphi_2 - \bar{\zeta}_1$, $\varrho = pd/\varphi_0$, and d the intrinsic displacement. Using eq. (45b) to find the spatial derivative of the motion parameter vector field then yields the curvature field as

$$\underline{\mathcal{E}}(x_1) = \underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}})\underline{\mathcal{P}}'. \tag{48}$$

Algorithm 18 (Curvature interpolation) *Interpolation of a curvature field defined by nodal motion parameter vectors, $\underline{\hat{\mathcal{P}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use interpolation formula* (45a) *to find the motion parameter vector, $\underline{\mathcal{P}}$.* ***Step (2)****. Use interpolation formula* (45b) *to obtain the spatial derivative of the motion parameter vector, $\underline{\mathcal{P}}'$.* ***Step (3)****. Use eq.* (48) *to find the interpolated curvature.*

Under a change of frame, algorithm 17 yields $\underline{\bar{\mathcal{P}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{P}}$ and eq. (45b) yields $\underline{\bar{\mathcal{P}}}' = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{P}}'$. Equation (48) leads to $\underline{\bar{\mathcal{E}}} = \underline{\underline{\mathcal{H}}}(\bar{\mathcal{P}})\bar{\mathcal{P}}'$. A fundamental property of the vectorial parameterization of motion is that its tangent operator is a second-order tensor [24], *i.e.*, $\underline{\underline{\mathcal{H}}}(\underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{P}}) = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}})\underline{\underline{\mathcal{C}}}_R$. It now follows that $\underline{\bar{\mathcal{E}}} = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}})\underline{\underline{\mathcal{C}}}_R\underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{P}}' = \underline{\underline{\mathcal{C}}}_R^{-1}\underline{\mathcal{E}}$, which proves that the algorithm is tensorial. Of course, the algorithm is not intrinsic because the tangent tensor appears explicitly in the expression for the curvature vector.

4.4 Interpolation of relative motion parameter vectors

To facilitate the writing of the present algorithm, the following compact notation is adopted: $\underline{\underline{\mathcal{C}}}(\underline{\mathcal{R}}) = \underline{\underline{\mathcal{C}}}(\underline{\mathcal{P}})\underline{\underline{\mathcal{C}}}(\underline{\mathcal{Q}}) \Leftrightarrow \underline{\mathcal{R}} = \underline{\mathcal{P}} \oplus \underline{\mathcal{Q}}$ and $\underline{\underline{\mathcal{C}}}(\underline{\mathcal{R}}) = \underline{\underline{\mathcal{C}}}^{-1}(\underline{\mathcal{P}})\underline{\underline{\mathcal{C}}}(\underline{\mathcal{Q}}) \Leftrightarrow \underline{\mathcal{R}} = \underline{\mathcal{P}}^- \oplus \underline{\mathcal{Q}}$. Following the developments presented in section 3.4, let $\underline{\underline{\hat{\mathcal{C}}}}_1$ and $\underline{\underline{\hat{\mathcal{C}}}}_2$ denote the motion tensors at nodes 1 and 2, respectively, resolved in frame $\mathcal{F}$. The relative motion of node 2 with respect to node 1, denoted $\underline{\underline{\hat{\mathcal{C}}}}_2^r$, is then

$\underline{\underline{\hat{\mathcal{C}}}}_2^r = \underline{\underline{\hat{\mathcal{C}}}}_2 \underline{\underline{\hat{\mathcal{C}}}}_1^{-1}$. Finally, resolving the components of this relative motion tensor in the frame defined by the motion at node 1 yields $\underline{\underline{\hat{\mathcal{C}}}}_2^{r*} = \underline{\underline{\hat{\mathcal{C}}}}_1^{-1} \underline{\underline{\hat{\mathcal{C}}}}_2$, where notation $(\cdot)^*$ indicates tensor components resolved in the frame defined by the motion tensor at node 1, denoted frame $\mathcal{F}^*$. Using the compact notation defined above, the relative nodal motion resolved in frame $\mathcal{F}^*$ become $\hat{\underline{\mathcal{P}}}_i^{r*} = \hat{\underline{\mathcal{P}}}_1^- \oplus \hat{\underline{\mathcal{P}}}_i$. These relative nodal motion parameter vectors are now interpolated to find

$$\underline{\mathcal{P}}^{r*}(x_1) = \sum_{i=1}^{N} h_i \hat{\underline{\mathcal{P}}}_i^{r*}, \tag{49a}$$

$$\underline{\mathcal{P}}^{r*\prime}(x_1) = \sum_{i=1}^{N} \frac{h_i^+}{J} \hat{\underline{\mathcal{P}}}_i^{r*}, \tag{49b}$$

where h_i are the shape functions.

Algorithm 19 (Finite motion interpolation) *Interpolation of a motion field defined by nodal motion parameter vectors, $\hat{\underline{\mathcal{P}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Compute the components of the relative nodal motion parameter vectors resolved in frame $\mathcal{F}^*$, $\hat{\underline{\mathcal{P}}}_i^{r*} = \hat{\underline{\mathcal{P}}}_1^- \oplus \hat{\underline{\mathcal{P}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (2)****. Use interpolation formula, eq.* (49a)*, to obtain the components of the relative motion parameter vector, $\underline{\mathcal{P}}^{r*}$, resolved in the same frame.* ***Step (3)****. The components of the interpolated motion parameter vector resolved in frame $\mathcal{F}$ are then $\underline{\mathcal{P}}(x_1) = \hat{\underline{\mathcal{P}}}_1 \oplus \underline{\mathcal{P}}^{r*}(x_1)$.*

Note that the interpolation process takes place in the coordinate system defined by frame $\mathcal{F}^*$. Of course, any other local frame could be used such as, for instance, the frame defined by the motion tensor at node 2. Algorithm 19 is otherwise identical to algorithm 17 based on the motion parameter vectors themselves, and hence, algorithm 19 is objective and tensorial, but not intrinsic. Indeed, the superposition of an arbitrary rigid body motion would be purged by the interpolation process, leading to the same interpolated motion.

4.4.1 Computation of curvature

The components of the curvature vector resolved in frame $\mathcal{F}^*$ are $\underline{\mathcal{E}}^* = \underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}}^{r*}) \underline{\mathcal{P}}^{r*\prime}$, where $\underline{\mathcal{P}}^{r*\prime}$ are the components of the spatial derivative of the motion field given by eq. (49b). It then follows that $\underline{\mathcal{E}}^* = \underline{\underline{\mathcal{H}}}(\underline{\underline{\hat{\mathcal{C}}}}_1^{-1} \underline{\mathcal{P}}^r) \underline{\underline{\hat{\mathcal{C}}}}_1^{-1} \underline{\mathcal{P}}^{r\prime} = \underline{\underline{\hat{\mathcal{C}}}}_1^{-1} \underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}}^r) \underline{\mathcal{P}}^{r\prime} = \underline{\underline{\hat{\mathcal{C}}}}_1^{-1} \underline{\mathcal{E}}$. The components of the curvature vector in frame $\mathcal{F}$ are then

$$\underline{\mathcal{E}}(x_1) = \underline{\underline{\hat{\mathcal{C}}}}_1 \underline{\underline{\mathcal{H}}}(\underline{\mathcal{P}}^{r*}) \underline{\mathcal{P}}^{r*\prime}. \tag{50}$$

Algorithm 20 (Curvature interpolation) *Interpolation of a curvature field defined by nodal motion parameter vectors, $\hat{\underline{\mathcal{P}}}_i$, $i = 1, 2, \ldots, N$.* ***Step (1)****. Use algorithm 19 to obtain the relative motion parameter vector $\underline{\mathcal{P}}^{r*}$ resolved in frame $\mathcal{F}^*$.* ***Step (2)****. Use interpolation formula, eq.* (49b)*, to obtain the spatial derivative of the relative motion parameter vector $\underline{\mathcal{P}}^{r*\prime}$, resolved in the same frame.* ***Step (3)****. Use eq.* (50) *to find the interpolated curvature.*

This algorithm is objective and tensorial, but not intrinsic, because it is a particular case of algorithm 18.

4.5 Interpolation of motion: numerical results

The numerical example presented in section 3.5 is expanded to deal with a general motion. The rotation field is that defined in the previous example, see section 3.5. The displacement field is

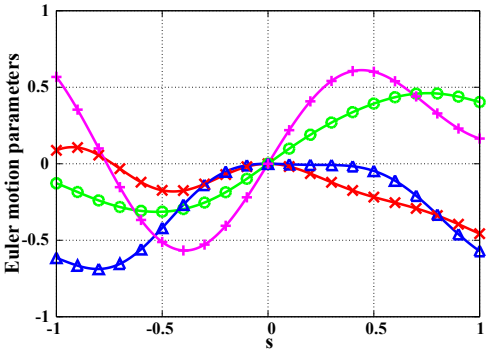

Figure 8: Motion field versus s. Euler motion parameters q_0 ($\times$), q_1 ($\circ$), q_2 ($\triangle$), and q_3 ($+$).

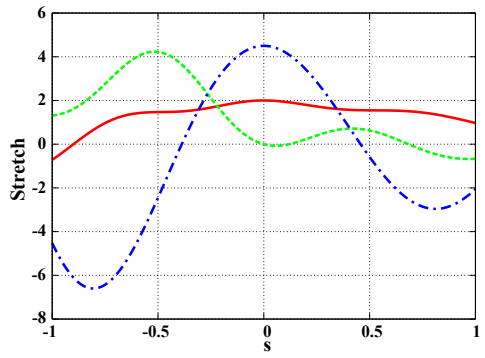

Figure 9: Stretch field versus s: ϵ_1 (solid line), ϵ_2 (dashed line), and ϵ_3 (dashed-dotted line).

defined in section 2.1. Using the properties of Euler motion parameters [1], the components of the stretch and curvature vectors are then found easily.

Figures 8 and 5 show the displacement and rotation related quaternion fields versus spatial variable s. The exact stretch and curvature fields are depicted in figs. 9 and 6 that show the components of the stretch and curvature vectors, respectively, denoted $\underline{\epsilon}^T = \{\epsilon_1, \epsilon_2, \epsilon_3\}$ and $\underline{\kappa}^T = \{\kappa_1, \kappa_2, \kappa_3\}$, respectively.

To assess the accuracy of the proposed algorithms, the interpolated curvature, denoted $\underline{\mathcal{E}}_a^T = \{\underline{\epsilon}_a^T, \underline{\kappa}_a^T\}$, was compared to its exact counterpart, denoted $\underline{\mathcal{E}}_e^T = \{\underline{\epsilon}_e^T, \underline{\kappa}_e^T\}$. The curvature error measure was selected as

$$e_{\mathcal{E}} = \frac{1}{N_{gp}} \sum_{k=1}^{N_{gp}} \frac{\|\underline{\mathcal{E}}_a(s_k) - \underline{\mathcal{E}}_e(s_k)\|}{\|\underline{\mathcal{E}}_e(s_k)\|}. \tag{51}$$

The exact and approximate curvatures were computed at N_{gp} sampling points denoted s_k. Within each element, $o_e + 1$ sampling points were selected to coincide with the location of the Gauss-Legendre quadrature points within the element.

Figure 10 shows the curvature error measure defined by eq. (51) versus the number of elements, $N_e \in [2, 256]$, on a logarithmic plot, when algorithm 13 is used to compute the curvature, *i.e.*, when the generalized polar decomposition theorem is used to extract a motion tensor from the interpolated motion tensor. Results are shown for linear, quadratic, cubic, and quartic shape functions; in each case, a linear regression was performed and is also shown on the figure. Figure 10 also shows the corresponding results when algorithm 14 is used to compute the curvature, *i.e.*, when Cayley's decomposition is used to extract a motion tensor from the interpolated motion tensor. These results show that the two algorithms yield very similar results.

Next, curvatures were computed using interpolation of the motion parameter vector; the motion Wiener-Milenković parameters [1] were selected for this study. Figure 10 shows the numerical results for this interpolation strategy when using the motion parameter vector and the *relative*

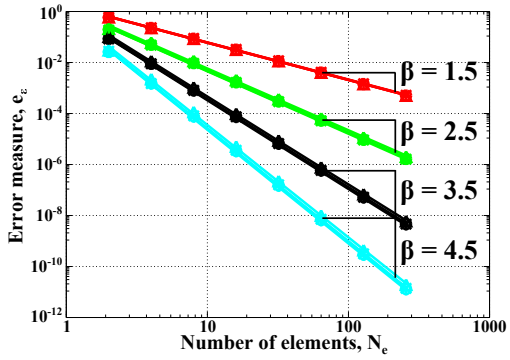

Figure 10: Curvature error measure (51) versus number of elements when using algorithm 13(×), 14(△), 18 (*), 20 (+), 16 (∘).

motion parameter vector, corresponding to algorithms 18 and 20, respectively. Here again, the two algorithms yield very similar results. Finally, curvatures were computed using interpolation of the Euler motion parameters, as described by algorithm 16 and the corresponding results are shown in fig. 10.

Linear regressions of the numerical data were performed in all cases and are shown as dashed lines in fig. 10. As the number of elements increases, the accuracy of the proposed curvature interpolation algorithms increases, yielding a positive coefficient β. For all the motion interpolation algorithms presented here, relationship (4) applied, as indicated on the figure.

When using linear shape functions, it is remarkable to note that the results of all algorithms presented here are nearly indistinguishable on the logarithmic scale of fig. 10. As the order of the shape functions increases, a trend appears in the results. The interpolation based on the relative Wiener-Milenković motion parameters is the most accurate, that based on the motion tensor is the least accurate, while that based on the Euler motion parameters falls between the other two. Although the interpolation using motion parameter vectors and relative motion parameter vectors are very similar, interpolation of relative motion parameter vectors is slightly more accurate.

5 Discussion

Table 1 presents a summary of all the algorithms presented in this study. The first five algorithms deal with the interpolation of rotation fields and the last five with the interpolation of motion. The approach and resulting properties of the motion interpolation algorithms are mirror images of their counterparts for rotation interpolation. The second column of the table provides references to the detailed algorithms.

The next three columns of the table list properties of each of algorithm and the last column gives a qualitative estimate of the computational cost required by each algorithm. This data calls for the following observations. First, the rotation (motion) interpolation algorithm based on polar decomposition possesses many desirable properties, namely, they are tensorial, objective, and intrinsic, but unfortunately, are also the most costly. The algorithms based on the interpolation of the relative rotation (motion) parameter vector are tensorial, objective, and low cost, but unfortunately not intrinsic, because they are based on a specific parameterization of rotation (motion). The lowest cost algorithms are based on the interpolation of the rotation (motion) parameter vector; these are tensorial, but neither intrinsic nor objective. The algorithm based on interpolation of Euler rotation (motion) is also low cost, but neither intrinsic nor objective. The least attractive algorithm is that based on Cayley's decomposition: it presents few desirable characteristics and is burdened by high cost.

While each algorithm predicts different interpolated rotation (motion) and curvature (stretch)

Table 1: Summary of the proposed algorithms. Rotation interpolation algorithm $(\cdot)^*$; curvature interpolation algorithm $(\cdot)^\dagger$; motion interpolation algorithm $(\cdot)^+$; curvature-stretch interpolation algorithm $(\cdot)^\ddagger$.

Nodal values	**Algorithm**	**Tensorial**	**Objective**	**Intrinsic**	**Cost**
Rotation tensor (Polar decomp.)	1^*, $3^\dagger$	✓	✓	✓	high
Rotation tensor (Cayley's decomp.)	2^*, $4^\dagger$	✓	×	×	high
Euler rotation parameters	5^*, $6^\dagger$	✓	×	×	low
Rotation parameter vector	7^*, $8^\dagger$	✓	×	×	low
Relative rotation parameter vector	9^*, $10^\dagger$	✓	✓	×	low
Motion tensor (Polar decomp.)	11^+, $13^\ddagger$	✓	✓	✓	high
Motion tensor (Cayley's decomp.)	12^+, $14^\ddagger$	✓	×	×	high
Euler motion parameters	15^+, $16^\ddagger$	✓	×	×	low
Motion parameter vector	17^+, $18^\ddagger$	✓	×	×	low
Relative motion parameter vector	19^+, $20^\ddagger$	✓	✓	×	low

fields, all converge to the exact solution at the same rate. Indeed, despite the observations made in the previous paragraph, it must be emphasized that all algorithms presented in this paper present the same $o_e + 1/2$ convergence rate, which is identical to the convergence rate observed for the interpolation of the displacement field. Clearly, interpolation of rotation (motion) fields is as accurate and converges as fast as the interpolation of displacement fields, a well established practice at the heart of the finite element method.

5.1 Computational cost of the algorithms

The representation of rotation (motion) can be categorized into redundant representations, which use more than three (six) parameters, and minimum set representations, which use three (six) parameters exactly. When algorithms are based on redundant representations, constraints among the redundant parameters are implied. These constraints are not respected by the interpolation process. Hence, suitable algorithms must be developed to post-process the interpolated quantities and ensure their compliance with the required constraints. This post-processing step is responsible for the increased computational cost. The higher the level of redundancy, the more expensive the algorithm becomes.

This explains why algorithms based on interpolation of the rotation (motion) tensor are more expensive than those based on Euler rotation (motion) parameters. The algorithms based on the rotation (motion) tensor, a redundant representation, are the only algorithms to be intrinsic because they do not call for a parameterization of rotation (motion), but are computationally expensive. The algorithms based on Euler rotation (motion) parameters, a redundant representation also, are cheaper.

The most effective algorithms are based on minimum set representations of rotation and motion.

5.2 Vectorial parameterization of rotation (motion)

It does not appear possible to interpolate representations of rotation that are not tensorial in nature, such as Euler angles, for instance. Bauchau *et al.* [23, 24, 1] have shown that parameterizations of

rotation (motion) are tensorial if and only if the rotation (motion) parameter vectors are parallel to the eigenvector of the rotation (motion) tensor associated with its unit eigenvalue. These representations are called "vectorial parameterization" of rotation (motion). Because of its tensorial nature, the rotation (motion) parameter vector is ideally suited for the interpolation of rotation (motion) fields, and this work also shows that this approach is computationally the most effective.

Clearly, this approach is not intrinsic. For instance, Crisfield and Jelenić [13, 14] used the Cartesian rotation vector for their developments, whereas the present work is based on the Wiener-Milenković parameterization. Other vectorial parameterizations [23, 24, 1], such as Cayley's, Gibbs', or Rodrigues', among others, could be used and yield results similar to those observed for the Wiener-Milenković parameters presented here. In particular, all have the same $o_e + 1/2$ convergence rate. Algorithms based on the interpolation of rotation (motion) parameter vectors appear to be the most attractive. They are not intrinsic, but achieve the highest accuracy with the lowest computational cost. They can be made objective by applying the interpolation technique to the relative rotation (motion) field; the incremental computational cost is small.

The vectorial parameterization of rotation exhibits desirable features, but singularities will occur for specific values of the rotation angle, as proved by Stuelpnagel [28]. These problems, however, can be eliminated by using a rescaling operation [23, 1]. Of course, when interpolating relative rotations, these problems disappear because relative rotations will remain small within each element, enabling the use of any vectorial parameterization of rotation without risking singularities.

5.3 Convergence rate

All the rotation and motion interpolation algorithms presented in this paper have the same $o_e + 1/2$ convergence rate. Textbooks [6, 7] focusing on the analysis of the finite element method establish the rate of convergence of the approach. For instance, a simple bar element based on the displacement formulation of the finite element method will exhibit an o_e convergence rate; *i.e.*, two-noded elements using linear shape functions converge at a linear rate, etc. This observation implies that the leading source of error in the FEM is not the interpolation of the displacement field ($o_e + 1/2$ convergence rate) but the approximation of equilibrium equations. Indeed, when using the principle of virtual work, the strain-displacement equations and constitutive laws are enforced exactly, whereas the equilibrium equations are satisfied in an integral sense only.

These observations help explain the statement of Jelenić and Crisfield [14], who mentioned that "The non-invariance and path-dependence in these formulations decrease with both p-refinement and h-refinement and in practical applications cannot always be easily spotted." This is simply due to the fact that rotation (motion) interpolation errors are not the leading cause of inaccuracy of the FEM.

6 Conclusions

From the numerical results presented in this paper, the following conclusions can be drawn. *(1)* All the algorithms converge at the same rate, $\beta \approx o_e + 1/2$, where o_e is the order of the element. Note that the same convergence rate is observed for the interpolation of displacement, rotation, and motion. *(2)* Although objective and intrinsic, algorithms based on the interpolation of the rotation (motion) tensor tend to be less accurate than their counterpart based on the Euler (motion) parameters or rotation (motion) parameter vectors; furthermore, they are also computationally more expensive. *(3)* Algorithms based on the interpolation of the nodal rotation (motion) parameter vectors tend to be the most accurate and most computationally efficient.

Certain properties of interpolation algorithms have received much attention in the literature. For instance Crisfield, Jelenić, and many others have questioned the objectivity of specific algorithms.

A cursory look at table 1 reveals that objective algorithms are no more accurate and do not converge at a faster rate than other algorithms. One could argue that intrinsic algorithms are superior to their extrinsic counterparts. Indeed, the predictions of the latter depend on the arbitrary selection of a specific parameterization of rotation or motion whereas the former do not. While such argument is pleasing from a theoretical viewpoint, it does not translate in improved accuracy or convergence rate; in fact, table 1 shows that intrinsic algorithms are less efficient from a computational standpoint.

Clearly, rotation (motion) interpolation algorithms are mathematical processes aimed at approximating the rotation (motion) field within one finite element as accurately as possible. Consequently, the selection of the most appropriate algorithm should be based on its accuracy and convergence characteristics only. The results presented here indicate that the algorithms based on the interpolation of rotation (motion) parameter vectors are the most attractive. They are not intrinsic, but achieve the highest accuracy with the lowest computational cost. Interpolation of the relative rotation (motion) parameter vectors also achieves objectivity at a marginally higher cost.

Interpolation of rotation (motion) is physically meaningful. It is as accurate as the interpolation of displacement, a widely accepted tool in the finite element method. The argument stating that "it is incorrect to interpolate rotations (motions) because rotations (motions) do not form a linear space" is groundless. It is valid to interpolate rotation and motion as long as the algorithm respects the tensorial nature of these quantities.

A Notational conventions

When dealing with unit quaternions, $\hat{e}$, the rotation tensor, $\underline{\underline{R}}$, is expressed by the following matrix of size 4×4,

$$\underline{\underline{D}}(\hat{e}) = \begin{bmatrix} 1 & \underline{0}^T \\ \underline{0} & \underline{\underline{R}}(\hat{e}) \end{bmatrix}. \tag{52}$$

The following matrices are used to manipulate quaternions

$$\underline{\underline{B}}(\hat{e}) = \begin{bmatrix} e_0 & -\underline{e}^T \\ \underline{e} & e_0\underline{\underline{I}} - \tilde{e} \end{bmatrix}, \quad \underline{\underline{S}}(\hat{p}) = \begin{bmatrix} 0 & \underline{0}^T \\ \underline{0} & \widetilde{p} \end{bmatrix}. \tag{53}$$

When dealing with bi-quaternions, $\check{g}^T = \left\{\hat{q}^T, \hat{e}^T\right\}$, the motion tensor, $\underline{\underline{\mathcal{C}}}$, is expressed by the following matrix of size 8×8,

$$\underline{\underline{\mathbb{C}}} = \begin{bmatrix} \underline{\underline{D}}(\hat{e}) & \underline{\underline{S}}(\hat{u})\underline{\underline{D}}(\hat{e}) \\ \underline{\underline{0}} & \underline{\underline{D}}(\hat{e}) \end{bmatrix}, \tag{54}$$

The following matrix is used to manipulate bi-quaternions

$$\underline{\underline{\mathbb{B}}}(\check{g}) = \begin{bmatrix} \underline{\underline{B}}^T(\hat{e}) & \underline{\underline{B}}^T(\hat{q}) \\ \underline{\underline{0}} & \underline{\underline{B}}^T(\hat{e}) \end{bmatrix}. \tag{55}$$

When dealing with motion, generalized skew-symmetric matrices are of the following form

$$\tilde{\mathcal{A}} = \begin{bmatrix} \tilde{b} & \tilde{a} \\ \underline{\underline{0}} & \tilde{b} \end{bmatrix}, \tag{56}$$

where $\tilde{a}$ and $\tilde{b}$ are 3×3 skew-symmetric matrices. Matrix $\underline{\underline{\mathcal{Z}}}$ is defined as follows

$$\underline{\underline{\mathcal{Z}}}(\alpha, \beta) = \begin{bmatrix} \beta\underline{\underline{I}} & \alpha\underline{\underline{I}} \\ \underline{\underline{0}} & \beta\underline{\underline{I}} \end{bmatrix}, \tag{57}$$

where α and β are two arbitrary scalars.

B Polar decomposition theorem

The *polar decomposition theorem* states that an invertible matrix, $\underline{\underline{A}}$, can be decomposed into the product of an orthogonal matrix, $\underline{\underline{R}}$, by a positive-definite, symmetric matrix, $\underline{\underline{U}}$,

$$\underline{\underline{A}} = \underline{\underline{R}}\,\underline{\underline{U}} = \underline{\underline{V}}\,\underline{\underline{R}}. \tag{58}$$

The second equality provides an alternative statement of the polar decomposition theorem, where matrix $\underline{\underline{A}}$ is now decomposed as the product a positive-definite, symmetric matrix, $\underline{\underline{V}}$, by the same orthogonal matrix, $\underline{\underline{R}}$. Matrices $\underline{\underline{R}}$, $\underline{\underline{U}}$, and $\underline{\underline{V}}$ are uniquely defined.

The polar decomposition theorem will be generalized by the following statement

$$\underline{\underline{\mathcal{T}}} = \underline{\underline{\mathcal{C}}}\,\underline{\underline{\mathcal{U}}} = \underline{\underline{\mathcal{V}}}\,\underline{\underline{\mathcal{C}}}, \tag{59}$$

where $\underline{\underline{\mathcal{C}}}$ is the motion tensor defined by eq. (26). Matrix $\underline{\underline{\mathcal{T}}}$ is an arbitrary matrix of the following form

$$\underline{\underline{\mathcal{T}}} = \begin{bmatrix} \underline{\underline{T}} & \underline{\underline{W}} \\ \underline{\underline{0}} & \underline{\underline{T}} \end{bmatrix}, \tag{60}$$

where $\underline{\underline{T}}$ is an invertible matrix and matrices $\underline{\underline{\mathcal{U}}}$ and $\underline{\underline{\mathcal{V}}}$ are

$$\underline{\underline{\mathcal{U}}} = \begin{bmatrix} \underline{\underline{U}} & \underline{\underline{F}} \\ \underline{\underline{0}} & \underline{\underline{U}} \end{bmatrix}, \tag{61a}$$

$$\underline{\underline{\mathcal{V}}} = \begin{bmatrix} \underline{\underline{V}} & \underline{\underline{G}} \\ \underline{\underline{0}} & \underline{\underline{V}} \end{bmatrix}. \tag{61b}$$

Matrices $\underline{\underline{U}}$ and $\underline{\underline{V}}$ are positive-definite and symmetric matrices; matrices $\underline{\underline{F}}$ and $\underline{\underline{G}}$ are symmetric.

The generalization of the polar decomposition theorem expressed by eq. (59) is proved easily. The first equality of eq. (59) implies $\underline{\underline{T}} = \underline{\underline{R}}\,\underline{\underline{U}}$; this means that matrices $\underline{\underline{R}}$ and $\underline{\underline{U}}$ correspond to the polar decomposition of $\underline{\underline{T}}$, see eq. (58), and therefore, matrix $\underline{\underline{R}}$ is orthogonal, matrix $\underline{\underline{U}}$ positive-definite and symmetric, and both are defined uniquely. Next, the first equality of eq. (59) also implies $\underline{\underline{W}} = \underline{\underline{R}}\,\underline{\underline{F}} + \tilde{u}\underline{\underline{R}}\,\underline{\underline{U}}$, which can be recast as $\tilde{u}^*\underline{\underline{U}} + \underline{\underline{F}} = \underline{\underline{R}}^T\underline{\underline{W}}$, where $\underline{u}^* = \underline{\underline{R}}^T\underline{u}$. Because matrix $\underline{\underline{F}}$ is symmetric, $(\tilde{u}^*\underline{\underline{U}}) - (\tilde{u}^*\underline{\underline{U}})^T = (\underline{\underline{R}}^T\underline{\underline{W}}) - (\underline{\underline{R}}^T\underline{\underline{W}})^T$, which leads to a linear system of equations for $\underline{u}^*$,

$$\left[\mathrm{tr}(\underline{\underline{U}})\underline{\underline{I}} - \underline{\underline{U}}\right]\underline{u}^* = 2\,\mathrm{axial}(\underline{\underline{R}}^T\underline{\underline{W}}). \tag{62}$$

Because matrices $\underline{\underline{U}}$, $\underline{\underline{R}}$, and $\underline{\underline{W}}$ are known, a unique solution exists $\underline{u}^*$, leading to $\underline{u} = \underline{\underline{R}}\underline{u}^*$, provided that the determinant of the system does not vanish. It is shown easily that $\det[\mathrm{tr}(\underline{\underline{U}})\underline{\underline{I}} - \underline{\underline{U}}] = (\lambda_2^2+\lambda_3^2)(\lambda_1^2+\lambda_3^2)(\lambda_1^2+\lambda_2^2)$, where λ_i^2, $i = 1,2,3$ are the positive eigenvalues of positive-definite matrix $\underline{\underline{U}}$, and hence, a solution of system (62) always exists. Finally, matrix $\underline{\underline{F}} = \underline{\underline{R}}^T(\underline{\underline{W}} - \tilde{u}\underline{\underline{T}})$, which is symmetric because $\underline{u}$ was determined by the solution of system (62). This proves that all the quantities appearing in decomposition (59) can be found unequivocally, hence proving the stated generalization of the polar decomposition theorem. The second equality in eq. (59) can be proved in a similar manner.

References

[1] O.A. Bauchau. *Flexible Multibody Dynamics*. Springer, Dordrecht, Heidelberg, London, New-York, 2011.

[2] A.A. Shabana and R.A. Wehage. A coordinate reduction technique for dynamic analysis of spatial substructures with large angular rotations. *Journal of Structural Mechanics*, 11(3):401–431, March 1983.

[3] O.P. Agrawal and A.A. Shabana. Application of deformable-body mean axis to flexible multibody system dynamics. *Computer Methods in Applied Mechanics and Engineering*, 56(2):217–245, 1986.

[4] A.A. Shabana. Flexible multibody dynamics: Review of past and recent developments. *Multibody System Dynamics*, 1(2):189–222, June 1997.

[5] M. Géradin and A. Cardona. *Flexible Multibody System: A Finite Element Approach*. John Wiley & Sons, New York, 2001.

[6] T.J.R. Hughes. *The Finite Element Method*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[7] K.J. Bathe. *Finite Element Procedures*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1996.

[8] O.A. Bauchau and J.I. Craig. *Structural Analysis with Application to Aerospace Structures*. Springer, Dordrecht, Heidelberg, London, New-York, 2009.

[9] L.E. Malvern. *Introduction to the Mechanics of a Continuous Medium*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1969.

[10] A.A. Shabana. Uniqueness of the geometric representation in large rotation finite element formulations. *Journal of Computational and Nonlinear Dynamics*, 5(4):044501 1–5, October 2010.

[11] A.A. Shabana and A.M. Mikkola. Use of the finite element absolute nodal coordinate formulation in modeling slope discontinuity. *ASME Journal of Mechanical Design*, 125(2):342–350, 2003.

[12] A.A. Shabana. General method for modeling slope discontinuity and T-sections using ANCF gradient deficient finite elements. *Journal of Computational and Nonlinear Dynamics*, 6(2):024502 1–6, April 2011.

[13] M.A. Crisfield and G. Jelenić. Objectivity of strain measures in the geometrically exact three-dimensional beam theory and its finite-element implementation. *Proceedings of the Royal Society, London: Mathematical, Physical and Engineering Sciences*, 455(1983):1125–1147, 1999.

[14] G. Jelenić and M.A. Crisfield. Geometrically exact 3D beam theory: Implementation of a strain-invariant finite element for static and dynamics. *Computer Methods in Applied Mechanics and Engineering*, 171:141–171, 1999.

[15] A. Ibrahimbegović, F. Frey, and I. Kozar. Computational aspects of vector-like parameterization of three-dimensional finite rotations. *International Journal for Numerical Methods in Engineering*, 38(21):3653–3673, 1995.

[16] A. Cardona and M. Géradin. A beam finite element non-linear theory with finite rotation. *International Journal for Numerical Methods in Engineering*, 26:2403–2438, 1988.

[17] J.C. Simo and L. Vu-Quoc. A three-dimensional finite strain rod model. Part II: Computational aspects. *Computer Methods in Applied Mechanics and Engineering*, 58(1):79–116, 1986.

[18] P. Betsch and P. Steinmann. Frame-indifferent beam element based upon the geometrically exact beam theory. *International Journal for Numerical Methods in Engineering*, 54:1775–1788, 2002.

[19] I. Romero and F. Armero. An objective finite element approximation of the kinematics of geometrically exact rods and its use in the formulation of an energy-momentum conserving scheme in dynamics. *International Journal for Numerical Methods in Engineering*, 54:1683–1716, 2002.

[20] I. Romero. The interpolation of rotations and its application to finite element models of geometrically exact rods. *Computational Mechanics*, 34(2):121–133, 2004.

[21] R.A. Wehage. Quaternions and Euler parameters. A brief exposition. In E.J. Haug, editor, *Computer Aided Analysis and Optimization of Mechanical Systems Dynamics*, pages 147–180. Springer-Verlag, Berlin, Heidelberg, 1984.

[22] M. Géradin and A. Cardona. Kinematics and dynamics of rigid and flexible mechanisms using finite elements and quaternion algebra. *Computational Mechanics*, 4:115–135, 1989.

[23] O.A. Bauchau and L. Trainelli. The vectorial parameterization of rotation. *Nonlinear Dynamics*, 32(1):71–92, 2003.

[24] O.A. Bauchau and L.H. Li. Tensorial parameterization of rotation and motion. *Journal of Computational and Nonlinear Dynamics*, 6:031007 1–8, 2011.

[25] T.F. Wiener. *Theoretical Analysis of Gimballess Inertial Reference Equipment Using Delta-Modulated Instruments*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1962. Department of Aeronautical and Astronautical Engineering.

[26] V. Milenković. Coordinates suitable for angular motion synthesis in robots. In *Proceedings of the Robot VI Conference, Detroit MI, March 2-4, 1982*, 1982. Paper MS82-217.

[27] O.A. Bauchau and J.Y. Choi. The vector parameterization of motion. *Nonlinear Dynamics*, 33(2):165–188, 2003.

[28] J. Stuelpnagel. On the parameterization of the three-dimensional rotation group. *SIAM Review*, 6(4):422–430, 1964.