J-Derivatives (JD) integrator for the 10-node tetrahedral element

Eli Hanukah
Mechanical Engineering Department, Braude College of Engineering, Karmiel.
P.O. Box 78, 2161002, Israel

Abstract

In the field of the Finite Element Method (FEM) for computational solid mechanics, there is a strong need to develop efficient integrators for element-level quantities, such as mass and stiffness matrices, internal and external forces, and so on. Currently, the Standard (ST) approach to numerical integration is employed. Thus, generally speaking, the more accurate (the higher order) quadrature (cubature) is chosen, the more integration points it includes, and the more computationally expensive it becomes. Therefore, the practical rule of thumb is to select the least costly scheme that is sufficiently accurate for global convergence.

This study is dedicated to the derivation of a highly efficient, yet sufficiently accurate, integrator for the element-level consistent mass matrix evaluation of the widely used 10-node tetrahedral element. We start by adopting the recently proposed J-Derivatives (JD) approach and, for the first time, specifying it for the quadratic tetrahedral element. We explicitly provide all the necessary details for the in-code implementation of the resulting formula. We account for both constant and varying initial density.

The accuracy of the suggested integrator is examined using five coarse-mesh geometries. Findings reveal that the suggested JD integrator outperforms the 11-point ST cubature by a wide margin and is practically equivalent to the 15-point ST scheme. Those results support the sufficient accuracy of our proposal. The relative 'polynomial' complexity is examined using a Computer Algebra System (CAS). Accordingly, the complexity of the complete, sufficiently accurate JD formula is 25% higher than that of the only 1-point ST integration. Also, we compare the CPU time of the in-code implementation. CPU time of the complete JD integration is 29% higher than that of only 1-point ST integration. All those outcomes convincingly support the proposed scheme as a practically desirable formula.

Key words: J-Derivatives; Semi-Analytical; Quadrature-free; non-Gaussian; closed-form; Finite Element Technology;

1 Introduction

A vast majority of practical real-life engineering problems associated with nonlinear solid mechanics are solved by means of the Finite Element Analysis (FEA). However, satisfactory solutions often require immense computational resources. Thus, progress in Finite Element Technology (FET), which aims to raise efficiency and overall performance of the method, is of great practical importance. Our contribution

Corresponding author(s): Eli Hanukah, Mechanical Engineering Department, Braude College of Engineering, Karmiel, eli.hanukah@braude.ac.il

provides a novel, highly efficient element-level integrator for the consistent mass matrix of the quadratic tetrahedral element.

Currently, to the best of our knowledge, for the vast majority of practical applications, the Standard (ST) numerical integration method is adopted, e.g. [48, 5]. Each scheme has its own number of integration points at which the integrand is evaluated and multiplied by the weight. The more integration points in the quadrature, the higher the scheme's accuracy (order); however, the higher the complexity. Thus, the practical rule of thumb is to use the least expensive while sufficiently accurate quadrature (cubature) for which global convergence occurs.

Several alternatives to conventional integration exist, for example, [1]. It is shown that symbolic/analytic work with high-level programming and explicit code generation can yield highly efficient code in many practical cases. Also, [52] demonstrates how Computer Algebra Systems (CAS) can be used to analytically treat and simplify stiffness matrix terms for an 8-node elastic element; the resulting expressions lead to significant time savings. Quadrature-free immersed isogeometric analysis can be found in [2]. The study proposes a novel method for solving partial differential equations on 3D CAD geometries using immersed isogeometric discretizations, without requiring quadrature schemes.

A code-generation strategy based on symbolic integration and polynomial common subexpression elimination (CSE) is explored [44]. A systematic evaluation of the approach, measuring operation count, execution time, and numerical error, is presented. Authors show that, with additional computational cost, it is possible to achieve much of, and sometimes substantially exceed, the performance of alternative approaches without compromising precision.

Closed-form terms for flat-faced tetrahedral elements' stiffness matrices show significant time savings compared to numerical integration [39]. Smoothed Finite Element Method (S-FEM) describes how smoothing changes the weak form and the element-level integrals (i.e., how stiffness assembly is modified) [35]. S-FEM variants, applications to dynamics/eigenvalue problems, and comments on mass/stiffness behavior and computational aspects, also elaborating on computational efficiency claims [57]. Presents a fully smoothed formulation where both stiffness and mass integrals are treated via smoothing/modified integrals — explicitly shows how domain integrals may be replaced by simpler boundary/smoothing cell integrals (necessary for element-level cost) [53]. These demonstrate how smoothing ideas are implemented at the element level (user-element subroutines), showing concrete element-level assembly strategies and their computational implications [43].

In this study, we follow and significantly improve on the Semi-Analytical (SA) approach for element-level mass matrix integration [13, 14, 15, 16, 17] and later [22, 23]. The SA approach exploited the integrand's mathematical structure. Specifically, it identified two types of multiplicative terms: coordinate-dependent and mesh-dependent ones. Then, it proposed a systematic approximation for the mixed-dependent function to achieve the desired separation between variables. Finally, it performs an analytical (symbolic) precomputation for the generalized weight matrices. The resulting formula took the form of a linear combination of the weight matrices multiplied by the sampled Jacobian function (metric). In the aftermath, the SA approach enabled a significant reduction in the number of sampling points required for a sufficiently

accurate element-level mass-matrix evaluation.

In our study, we follow the recently proposed J-Derivatives (JD) approach [20] for the evaluation of the element-level (element frame) mass matrix. Similar to the SA approach, the JD method leverages the space and mesh separation and proposes a systematic approximation of the mixed terms, followed by the analytical precomputation of the generalized weight matrices. However, the approximation is based on partial derivatives rather than the sampled values. This makes our integrator non-Gaussian or quadrature-free. Also, JD proposes an intelligent (efficient) way to express those partial derivatives such that the resulting scheme, while still sufficiently accurate, significantly overperforms the existing ones in efficiency.

Following the above, Section **2** recalls all the necessary details regarding the mathematical problem statement of the element-level consistent mass matrix. The Standard (ST) approach to numerical integration is also mentioned. Then, Subsection **2.1** recalls the recently proposed J-Derivatives (JD) method, allowing for the constant and varying initial density. Next, Section **3** specifies the JD approach to the 10-node tetrahedral element. All the necessary terms are derived and explicitly presented for in-code implementation. Section **4** explores the accuracy of the proposed formula based on a comparative accuracy study. A set of five coarse-mesh, automatically-meshed geometries is selected to compare the 11-point and 15-point ST integrations with the JD formula. Section **5** explores the complexity and the CPU time of the proposed approach. It is compared to the least expansive possible integration - the 1-point ST quadrature. Finally, Section **6** discusses the results and draws important practical conclusions. Appendix **A** keeps a record of important technical details of the practical JD scheme.

2 Mathematical background

In this section, we recall the mathematical problem statement of the consistent mass matrix. Then, we elaborate on the conventional (ST) approach to numerical integration. Finally, we explore the recently proposed J-Derivatives (JD) method.

The number of dimensions (either 2D or 3D) is denoted by n_{dim} and the number of element node is given by n_{nodes}. Shape functions $N_I(\boldsymbol{\xi})$, where $(I = 1, .., n_{nodes})$ are explicit polynomials of local coordinates $\boldsymbol{\xi}$, such that for the 2D case $n_{dim} = 2$, $\boldsymbol{\xi} = (\xi, \eta)$, while for the common 3D case $n_{dim} = 3$, $\boldsymbol{\xi} = (\xi, \eta, \zeta)$, see for example (23). Initial material density in the initial element domain is denoted by ρ_0; unless specifically mentioned, the density function is not constant.

Integration in the parent element domain is commonly denoted by $\int_{V_\square} (\cdot)\, d\square$, where for the $n_{dim} = 3$, the $d\square$ denotes $d\square = d\xi d\eta d\zeta$. Explicit integration bounds in the tetrahedral domain are given by $\int_0^1 \int_0^{1-\zeta} \int_0^{1-\eta-\zeta} (\cdot)\, d\xi d\eta d\zeta$. Elemental volume of the initial volume domain dV is given by $dV = J d\xi d\eta d\zeta = J d\square$, where J is the Jacobian function (metric), transforming from the parent element to the initial configuration element. Jacobian $J(\boldsymbol{\xi}; X_{Im})$, $(I = 1, .., n_{nodes}, m = 1, .., n_{dim})$ is a function of local coordinates $\boldsymbol{\xi}$ and of the mesh - nodal positions with respect to the global coordinate system $(\boldsymbol{e}_1, .., \boldsymbol{e}_{n_{dim}})$ namely $\boldsymbol{X}_I = \sum_{m=1}^{n_{dim}} X_{Im} \boldsymbol{e}_m$ where $(I = 1, .., n_{nodes})$.

Jacobian is the determinant of the Jacobian matrix. Let's denote an arbitrary ma-

terial point X occupying spatial point $\boldsymbol{X}$ in the initial element domain, following the isoparametric concept $\boldsymbol{X} = \sum_{I=1}^{n_{nodes}} N_I(\boldsymbol{\xi})\, \boldsymbol{X}_I$. The Jacobian matrix $[J]$ is defined by $[J] = Grad_{\boldsymbol{\xi}}(\boldsymbol{X})$. Thus, for the 3D case $n_{dim} = 3$, the Jacobian matrix is given by

$$[J] = \begin{bmatrix} X_{1;\xi} & X_{1;\eta} & X_{1;\zeta} \\ X_{2;\xi} & X_{2;\eta} & X_{2;\zeta} \\ X_{3;\xi} & X_{3;\eta} & X_{3;\zeta} \end{bmatrix}$$
$$J_{mk} = X_{m;k} = \sum_{I=1}^{n_{nodes}} N_{I;k}(\boldsymbol{\xi})\, X_{Im} \quad , \quad (m,k = 1,..,n_{dim}) \tag{1}$$

Where $X_m = \boldsymbol{X} \cdot \boldsymbol{e}_m = \sum_{I=1}^{n_{nodes}} N_I X_{Im}$, $(m = 1,..,n_{dim})$. Semicolon represents the partial differentiation with respect to local coordinates, e.g., $N_{I;2} = \frac{\partial N_I}{\partial \eta}$. Determinant admits the standard definition

$$J(\boldsymbol{\xi}, X_{Im}) = Det([J]) = J_{11}J_{22}J_{33} - J_{11}J_{23}J_{32} - J_{12}J_{21}J_{33} + J_{12}J_{23}J_{31} + J_{13}J_{21}J_{32} - J_{13}J_{22}J_{31} \tag{2}$$

Consistent element-level mass matrix M_{IJ}, $(I,J = 1,..,n_{nodes})$ is defined by

$$M_{IJ} = \int_{V_e} \rho_0 N_I N_J dV = \int_{V_\square} \rho_0 N_I(\boldsymbol{\xi})\, N_J(\boldsymbol{\xi})\, J(\boldsymbol{\xi}; X_{K,n})\, d\square$$
$$(I,J,K = 1,..,n_{nodes}),(n = 1,..,n_{ndim}) \tag{3}$$

It is noted that the above (3), provided constant initial density ρ_0, can be evaluated exactly, since all the integrand's multiplicative terms are polynomials with respect to the local coordinates $\boldsymbol{\xi}$, see (1)(2)(23). However, the resulting expression will be highly inefficient and redundantly accurate. The practical rule is to adopt the most effective and sufficiently accurate scheme. Therefore, to the best of our knowledge, all commercial packages employ the Standard (ST) approach to numerical integration for 3D solid elements, specifically for the 10-node tetrahedral element.

Standard (ST) numerical integration methods are based on integrand evaluation at a prescribed quadrature (cubature) points multiplied by weights associated with each point (e.g., Gauss points). Each quadrature has an integration order, which is the order of the polynomial that is integrated exactly using that scheme. Typically, the higher-order scheme, i.e., the more accurate quadrature, requires more sampling points and, as a consequence, complexity rises. ST approximation takes the next form

$$M_{IJ}^{ST} = \sum_{q=1}^{n_{ip}} w_q \rho_{0q} N_{Iq} N_{Jq} J_q(X_{K,n})$$
$$(I,J,K = 1,..,n_{nodes}),(n = 1,..,n_{ndim}) \tag{4}$$

Where n_{ip} stands for the number of integration points, w_q are weights associated with sampling point $(q = 1,..,n_{ip})$. Initial density evaluated at the integration point is denoted by ρ_{0q}. Shape functions and the Jacobian function evaluated at the integration points are denoted by N_{Iq} and J_q, respectively. Clearly, the higher the n_{ip}, the more terms the above ST approximation (4) contains, and the complexity increases

$$M_{IJ}^{ST} \approx w_1 N_{I1} N_{J1} \rho_{01} J_1 + w_2 N_{I2} N_{J2} \rho_{02} J_2 + ... + w_{n_{ip}} N_{In_{ip}} N_{Jn_{ip}} \rho_{0n_{ip}} J_{n_{ip}}$$
$$(I,J,K = 1,..,n_{nodes}),(n = 1,..,n_{ndim})$$

While the above (4) is implemented in a rather straightforward manner, we wish to include an additional representation that provides a generalizing perspective and serves as a connecting bridge to the latest developments. By adopting the next definition

$$M^{ST}_{IJq} = w_q N_{Iq} N_{Jq}, \quad (I, J = 1, .., n_{nodes}), (q = 1, .., n_{ip}) \tag{5}$$

The ST approximation (4) takes the form

$$\begin{gathered} M^{ST}_{IJ} = \sum_{q=1}^{n_{ip}} M^{ST}_{IJq} \ (\rho_{0q} J_q) \\ (I, J = 1, .., n_{nodes}) \end{gathered} \tag{6}$$

Clearly, the above (6) and (4) are identical. However, here, the ST approximation is represented as a linear combination of n_{ip} generalized weight matrices M^{ST}_{IJq} multiplied by $(\rho_0 J)$ function sampled at the integration point. One might ask - do better generalized weights exist such that for the same n_{ip} the resulting accuracy increases dramatically? This question has been answered positively by proposing the Semi-Analytical (SA) approach, e.g., [22, 23]. SA element-level mass matrix integrator $M^{SA}_{IJ} = \sum_{q=1}^{n_{ip}} M^{SA}_{IJq} \ (\rho_{0q} J_q)$ allowed to reduce several times (two to four times) the necessary amount if integration points by prescribing superior generalized weights matrices. Our contribution represents an additional step forward by achieving an equally dramatic increase in efficiency.

2.1 J-Derivatives (JD) approach

Herein, we recall the recently proposed J-Derivatives (JD) approach [20]. The JD method enables the derivation of highly efficient, case-specific integrators tailored for evaluating the element-level mass matrix. The tremendous efficiency increase is made possible by two cornerstones. The first is taking advantage of a specific integrand form in (3) by separating the coordinate-dependent and mesh-dependent terms. This was made possible due to an analytical approximation for the $(\rho_0 J)$ function. In addition, a computationally beneficial form of partial derivatives is found. Notably, we use CAS to derive all required expressions explicitly. Let us first present the approach based on the most practical case of constant initial density $\rho_0 = const$. In practice, for the majority of simulations, the mesh is fine enough to assume it is constant within each element. However, the generalization to varying density immediately follows. It introduces no computational or theoretical difficulties. For the constant density, the JD integrator is non-Gaussian; it does not require the notion of integration (or sampling) points.

An integrand in (3) is given by $\rho_0 N_I(\boldsymbol{\xi}) N_J(\boldsymbol{\xi}) J(\boldsymbol{\xi}; X_{KM})$, we distinguish between coordinate and mesh-dependent functions. However, Jacobian is both, thus it is analytically approximated by means of Taylor's multivariate expansion about the centroid $\boldsymbol{\xi}^* = (\xi^*, \eta^*, \zeta^*)$. With the help of the next definitions

$$\tilde{N}_0 = 1, \quad \tilde{N}_1 = (\xi - \xi^*), \quad \tilde{N}_2 = (\eta - \eta^*), \quad \tilde{N}_3 = (\zeta - \zeta^*) \tag{7}$$

$$\begin{aligned} \tilde{J}_0\left(X_{Km}\right) &= J\left(\boldsymbol{\xi}; X_{Km}\right)\big|_{\boldsymbol{\xi}^*} \qquad \tilde{J}_1\left(X_{Km}\right) = \frac{\partial J\left(\boldsymbol{\xi}; X_{Km}\right)}{\partial \xi}\big|_{\boldsymbol{\xi}^*} \\ \tilde{J}_2\left(X_{Km}\right) &= \frac{\partial J\left(\boldsymbol{\xi}; X_{Km}\right)}{\partial \eta}\big|_{\boldsymbol{\xi}^*} \qquad \tilde{J}_3\left(X_{Km}\right) = \frac{\partial J\left(\boldsymbol{\xi}; X_{Km}\right)}{\partial \zeta}\big|_{\boldsymbol{\xi}^*} \end{aligned} \tag{8}$$

Linearly approximated Jacobian function (metric) takes the convenient form

$$\begin{gathered} J\left(\boldsymbol{\xi}; X_{Jn}\right) \approx \sum_{q=0}^{3} \tilde{N}_q\left(\boldsymbol{\xi}\right) \tilde{J}_q\left(X_{Km}\right) \\ \left(J, K = 1, .., n_{nodes}\right), \quad \left(n, m = 1, .., n_{dim}\right) \end{gathered} \tag{9}$$

Now, the above (9) is installed in the original problem statement (3). The definition of the generalized weight matrices is as follows

$$\begin{gathered} M_{IJq}^{JD} = \int_{V_\square} N_I N_J \tilde{N}_q \; d\square \\ \left(I, J = 1, .., n_{nodes}\right), \quad \left(q = 0, .., 3\right) \end{gathered} \tag{10}$$

Therefore, the JD integrator for the constant density case is given by

$$M_{IJ}^{JD} = \sum_{q=0}^{3} M_{IJq}^{JD}\left(\rho_0 \tilde{J}_q\right), \quad \left(I, J = 1, .., n_{nodes}\right) \tag{11}$$

Importantly, generalized coefficient matrices M_{IJq}^{JD} given by (10) are precomputed numbers, similar to weights in ST numerical integration. For example, our study provides those values for the 10-node tetrahedral element. This enables in-code integrator implementation. Also, the above linear combination of generalized weight matrices resembles the previously mentioned ST representation and the SA integrators (6). However, the above JD uses the Jacobian's partial derivatives rather than the Jacobian evaluated at the sampling point. This point will be addressed later since it is an additional source of JD efficiency superiority. Also, we note that the above (11) is a non-Gaussian integrator (quadrature-free) in the sense that it does not require the notion of sampling points.

What is the order of the above JD formula? We can elaborate on the topic only in the non-rigorous sense. JD integrator (11) is exact for the linear Jacobian element. Namely, as long as the Jacobian is constant (regular mesh) or a linear function of coordinates $\boldsymbol{\xi}$, the formula is exact. Thus, the order is equal to the order of the integrand in (10). For example, for the 10-node tetrahedral element, shape functions N_I are quadratic (23); thus, $(N_I N_J \tilde{N}_q)$ is fifth order with respect to $\boldsymbol{\xi}$, see (7). In fact, the JD integrator approximates all coarse-mesh elements using a linear Jacobian. Based on previous SA studies, this is a sufficiently accurate assumption for a mass matrix integrator [22, 23]. We'll reproduce those findings in the comparative accuracy study.

The central question now is how to express partial derivatives $\tilde{J}_q$, $(q = 0, .., 3)$ such that the JD integrator (11) will be significantly less complex (more efficient) than all the alternatives? The secret is to express the partial derivatives in terms of partial derivatives of the Jacobian matrix and not in terms of the mesh! As such, the Jacobian matrix (1) is linearized with the help of Taylor's multivariate expansion about

the centroid $\boldsymbol{\xi}^* = (\xi^*, \eta^*, \zeta^*)$. To this end, we recall the definition (7) and use the next one

$$\left[J^0\right] = [J]\,\big|_{\boldsymbol{\xi}^*} \qquad \left[J^1\right] = \frac{\partial\,[J]}{\partial \xi}\big|_{\boldsymbol{\xi}^*} \qquad \left[J^2\right] = \frac{\partial\,[J]}{\partial \eta}\big|_{\boldsymbol{\xi}^*} \qquad \left[J^3\right] = \frac{\partial\,[J]}{\partial \zeta}\big|_{\boldsymbol{\xi}^*} \tag{12}$$

to write the linear approximation of $[J]$, given by (1) as

$$\begin{gathered}[J] \approx \sum_{q=0}^{3} \tilde{N}_q\,(\boldsymbol{\xi})\,\left[J^q\,(X_{Km})\right] \\ (K = 1, .., n_{nodes})\,, (m = 1, .., n_{dim})\end{gathered} \tag{13}$$

Partial derivative matrices (12) are evaluated using a CAS; for the 10-node tetrahedral, $[J^q]$, $(q = 0, .., 3)$ are given by (24,25,26,27). Partial derivatives of the Jacobian function $\tilde{J}_q$, $(q = 0, .., 3)$ follow (2), (8), (9) and the above (13) to take the explicit, computationally compact form

$$\begin{gathered}\tilde{J}_0 = J^0_{1,1}J^0_{2,2}J^0_{3,3} - J^0_{1,1}J^0_{2,3}J^0_{3,2} - J^0_{1,2}J^0_{2,1}J^0_{3,3} + \\ J^0_{1,2}J^0_{2,3}J^0_{3,1} + J^0_{1,3}J^0_{2,1}J^0_{3,2} - J^0_{1,3}J^0_{2,2}J^0_{3,1}\end{gathered} \tag{14}$$

$$\begin{gathered}\tilde{J}_m = J^0_{1,1}J^0_{2,2}J^m_{3,3} - J^0_{1,1}J^0_{2,3}J^m_{3,2} - J^0_{1,1}J^0_{3,2}J^m_{2,3} + J^0_{1,1}J^0_{3,3}J^m_{2,2} - J^0_{1,2}J^0_{2,1}J^m_{3,3} + \\ J^0_{1,2}J^0_{2,3}J^m_{3,1} + J^0_{1,2}J^0_{3,1}J^m_{2,3} - J^0_{1,2}J^0_{3,3}J^m_{2,1} + J^0_{1,3}J^0_{2,1}J^m_{3,2} - J^0_{1,3}J^0_{2,2}J^m_{3,1} - \\ J^0_{1,3}J^0_{3,1}J^m_{2,2} + J^0_{1,3}J^0_{3,2}J^m_{2,1} + J^0_{2,1}J^0_{3,2}J^m_{1,3} - J^0_{2,1}J^0_{3,3}J^m_{1,2} - J^0_{2,2}J^0_{3,1}J^m_{1,3} + \\ J^0_{2,2}J^0_{3,3}J^m_{1,1} + J^0_{2,3}J^0_{3,1}J^m_{1,2} - J^0_{2,3}J^0_{3,2}J^m_{1,1} \\ (m = 1, 2, 3)\end{gathered} \tag{15}$$

Using the above (14),(15), the proposed JD integrator (11) becomes not only sufficiently accurate (see Section 4), but also highly efficient (see Section 5).

Finally, we'll generalize (11) for the varying initial density $\rho_0 \neq const$. Derivation follows the same theoretical guidelines and poses no computational or theoretical difficulty. Previously, we've kept the density constant and linearly approximated the Jacobian using its partial derivatives. Now, we'll linearly approximate the multiplicative function $(\rho_0 J)$, and use its partial derivatives in the resulting integrator. As such, let us assume a linear density distribution within the 10-node tetrahedral element domain (while a careful reader notes that the proposed model can be used as is, for any other domain as well). Initial density values at the four vertex nodes are known and denoted by ρ_{0m}, $(m = 1, 2, 3, 4)$, thus

$$\begin{gathered}\rho_0 = (1 - \xi - \eta - \zeta)\,\rho_{01} + \xi\rho_{02} + \eta\rho_{03} + \zeta\rho_{04} \\ \rho_{0m} = \rho_0\big|_{\boldsymbol{\xi} = NODE\ m} \qquad (m = 1, 2, 3, 4)\end{gathered} \tag{16}$$

Then, with the help of linearized Jacobian given by (9), the above density approximation (16), and definitions (7-8), the linear multivariate approximation of the (now quadratic function) $(\rho_0 J)$ about the centroid $\boldsymbol{\xi}^* = (\xi = \xi^*, \eta = \eta^*, \zeta = \zeta^*)$, takes the form

$$\begin{gathered}(\rho_0 J) \approx \sum_{q=0}^{3} \tilde{N}_q \tilde{J}^{\rho_0}_q \qquad \tilde{J}^{\rho_0}_0 = (\rho_0 J)\,\big|_{\boldsymbol{\xi}^*} \\ \tilde{J}^{\rho_0}_1 = \frac{\partial\,(\rho_0 J)}{\partial \xi}\big|_{\boldsymbol{\xi}^*} \qquad \tilde{J}^{\rho_0}_2 = \frac{\partial\,(\rho_0 J)}{\partial \eta}\big|_{\boldsymbol{\xi}^*} \qquad \tilde{J}^{\rho_0}_3 = \frac{\partial\,(\rho_0 J)}{\partial \zeta}\big|_{\boldsymbol{\xi}^*}\end{gathered} \tag{17}$$

where partial derivatives $\tilde{J}_q^{\rho_0}$, $(q = 0, 1, 2, 3)$ are given by

$$\begin{aligned} \tilde{J}_0^{\rho_0} &= \tilde{\rho}_{00}\tilde{J}_0 \qquad \tilde{J}_m^{\rho_0} = \tilde{\rho}_{0m}\tilde{J}_0 + \tilde{\rho}_{00}\tilde{J}_m \qquad (m = 1, 2, 3) \\ \tilde{\rho}_{00} &= \frac{1}{4}\left(\rho_{01} + \rho_{02} + \rho_{03} + \rho_{04}\right) \qquad \tilde{\rho}_{0m} = \rho_{0\,m+1} - \rho_{01} \end{aligned} \tag{18}$$

Then, substitution of (17) to the original problem statement (3) followed by the consequential adoption of the generalized weights matrices (10), results in the JD integrator

$$M_{IJ}^{JD} = \sum_{q=0}^{3} M_{IJq}^{JD}\tilde{J}_0^{\rho_0} \tag{19}$$

Clearly, the above (19) and the previously developed (11) agree with each other when the density is constant, as it can be seen in (18). Importantly, in terms of additional complexity, all the extra required computations are given by (18), and are practically negligible.

Theoretically speaking, the above integrator is less accurate than (11), since it is exact for linear $(\rho_0 J)$ and not for the linear Jacobian function. However, this phenomenon is true for all the other integrators as well. If one decides to vary the initial density rather than lower the element size until the constant-density assumption is sufficient for practical purposes, they'll experience a decrease in accuracy. It is unclear which approach suffers more, since the exact solution for comparison does not exist; one does not have "the true" density distribution function, but merely its sampled values. Still, we provide the generalized version to avoid appearing less capable than other integrators that do allow non-constant density. Finally, we wish to remark that for a linearly distributed density with a constant Jacobian (regular mesh), the above JD is exact ((17) truly linear). In that sense, JD admits 'full integration', which is a well-known practical requirement for global system conversion. This fact once again supports the practical claim of sufficient accuracy.

3 Specification for the 10-node tetrahedral element

In this section, we specify, for the first time, the JD integrator to the widely-used 10-nodes quadratic tetrahedral element, e.g., [55]. As such, we provide all the details for the practical in-code implementation. Based on those expressions, comparisons of accuracy, complexity, and CPU time are performed. Shape functions are recalled in (23).

Firstly, partial Jacobian matrices $[J^q]$, $(q = 0, 1, 2, 3)$, must be evaluated. To this end, Jacobian matrix definition (1) together with the standard shape functions (23) is used in Taylor's multivariate expansion (13) about the tetrahedral centroid given by $\boldsymbol{\xi}^* = (\xi = 1/4, \eta = 1/4, \zeta = 1/4)$. The resulting matrices are detailed in (24,25,26,27). Unsurprisingly, those are mesh-dependent matrices, yet the only two necessary constants to be remembered are given by

$$c_1 = 4 \qquad c_2 = 2 \tag{20}$$

Notably, for our specific case of quadratic tetrahedral, the linear approximation of the Jacobian matrix is an exact representation. The order in (1) follows the order of

partial derivatives of the shape functions (23), which are quadratic. Thus $N_{I;m}$ where $(I = 1, .., n_{nodes})$ and $(m = 1, .., n_{ndim})$ are linear with respect to $\boldsymbol{\xi}$.

Secondly, partial derivatives $\tilde{J}_q$ for $(q = 0, 1, 2, 3)$ must be evaluated using the explicit formula (14) and (15). Those hold for every $n_{dim} = 3$ element. Moreover, this holds even for higher-order JD models, where an integrator developer wishes to leverage higher-order approximations of the Jacobian matrix (and/or the Jacobian function).

Thirdly, generalized weight matrices M^{JD}_{IJq} where $(I, J = 1, .., n_{nodes})$ and $(q = 0, 1, 2, 3)$ must be evaluated. To this end, definitions (10),(23), and (7) are recalled together with explicit integration bounds in the tetrahedral element domain

$$M^{JD}_{IJq} = \int_{V_\square} N_I N_J \tilde{N}_q \; d\square = \int_0^1 \int_0^{1-\zeta} \int_0^{1-\eta-\zeta} N_I N_J \tilde{N}_q \; d\xi d\eta d\zeta \\ (I, J = 1, .., n_{nodes}), (q = 0, 1, 2, 3) \tag{21}$$

The above weight matrices are symmetric $M^{JD}_{IJq} = M^{JD}_{JIq}$, thus, at most, one can expect 55 independent components in each matrix. However, after careful examination of the results, there are only 10 independent entries for all the weight matrices. Let us define

$$m_1 = \frac{1}{10080}, \quad m_2 = \frac{2}{10080}, \quad m_3 = \frac{4}{10080}, \quad m_4 = \frac{6}{10080}, \quad m_5 = \frac{8}{10080} \\ m_6 = \frac{16}{10080}, \quad m_7 = \frac{24}{10080}, \quad m_8 = \frac{32}{10080}, \quad m_9 = \frac{64}{10080}, \quad m_{10} = \frac{128}{10080} \tag{22}$$

With the help of the above definition (22), weight matrices (21) are given by (28)-(31). Now, computation (11) can be realized in code.

Finally, several remarks regarding in-code implementation. (i) We do use CAS for automatic code generation; however, as it was shown throughout the section, there is no extremely lengthy code often associated with the 'closed-form' approach. (ii) There are only twelve constants to be remembered, given by (20) and (22). This is a dramatically smaller number of constants than necessary for the ST approach. As such, following the ST integration (4), one needs weights w_q, $(q = 1, .., n_{ip})$, plus N_{Iq}, $(I = 1, .., n_{nodes}, q = 1, .., n_{ip})$, and partial derivatives of the shape functions for (1), denoted by $N^q_{I;m} = N_{I;m}(\boldsymbol{\xi} = \boldsymbol{\xi}^q)$, $(I = 1, .., n_{nodes}, m = 1, .., n_{dim}, q = 1, .., n_{ip})$. Overall, it is at most (n_{ip}) plus $(10 \times n_{ip})$ plus $(10 \times 3 \times n_{ip})$, which is many times higher than twelve, even for $(n_{ip} = 1)$. (iii) Generalized weight matrices (28)-(31) consist of many zeros, which can be accounted for by explicit code generation, for example, both of those are interchangeable

$$M^{JD}_{17} = \rho_0 (M^{JD}_{170} \tilde{J}_0 + M^{JD}_{171} \tilde{J}_1 + M^{JD}_{172} \tilde{J}_2 + M^{JD}_{173} \tilde{J}_3) \\ M^{JD}_{17} = \rho_0 (-m_6 \tilde{J}_0 - m_3 \tilde{J}_2)$$

Clearly, one is more economical than the other.

4 Comparative accuracy study

Now that the proposed JD integrator has been detailed, we must establish its accuracy and computational cost. Specifically, it is essential to demonstrate that the

scheme is sufficiently accurate. Then, it is crucial to establish efficiency superiority. Both of those, if true, would indicate that the suggested JD formula is highly desirable for practical implementation.

A Comparative accuracy study is performed based on five test geometries, which were generated using ABAQUS and meshed with an automatic mesh generator [22]. The first geometry is a regular mesh (constant Jacobian/flat-faced); the second is a ball; and the last three bodies have been made with especially poor meshing. Moreover, in the fifth geometry, several elements are singular, namely, the Jacobian crosses zero at some point in the domain. We kept it to test it as the worst-case scenario (Figure **1**).

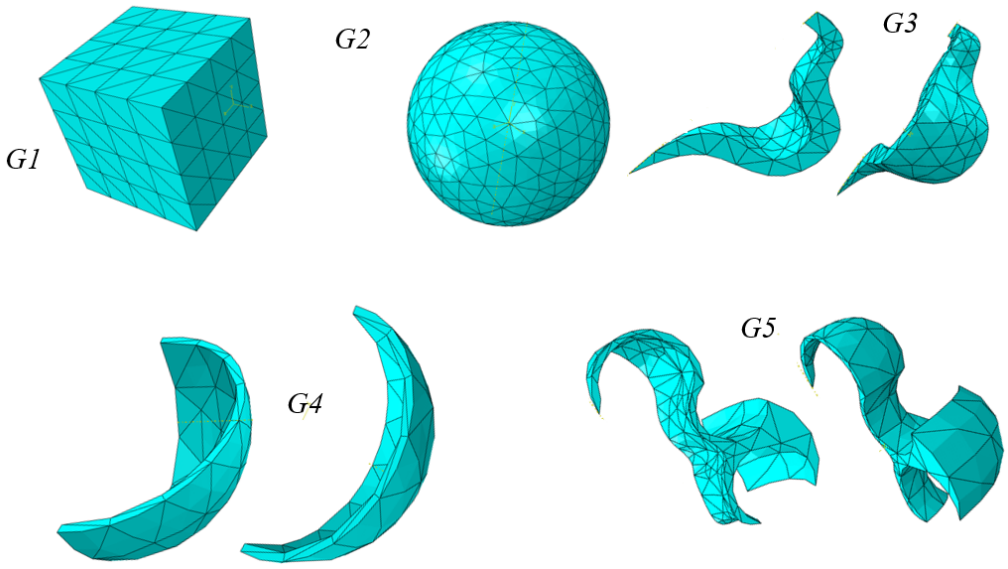

Figure 1: Test geometries G1 trough G5. G1 is a regular-mesh (constant Jacobian) box geometry. G2 - ball geometry. G3-G5 curved bodies, coarse-mesh geometries. All meshes have been generated using an automatic mesh generator.

Table **1** keeps record of the number of elements and nodes for each geometry in the test set

	G1	G2	G3	G4	G5
# Elements	583	3358	332	136	302
# Nodes	994	5147	682	310	772

Table 1: - records the number of elements and nodes for each test geometry G1,..,G5

In our comparative accuracy study, we've chosen two practical quadratures to compare to, the 11-point and the 15-point quadratures [48, 5]. For each element in every test geometry, we evaluated the exact mass matrices. Then, for every element, we evaluate all the mass matrices using 11-point ST, 15-point ST, and the suggested JD integrator. All errors are reported in percentiles %. We report the average error, the minimum average error (for the most accurate matrix in the geometry), and the maximum average error (for the least accurate matrix in the geometry), see Figure **2**.

%	Geom 1			Geom 2			Geom 3			Geom 4			Geom 5		
	11p	15p	JD	11p	15p	JD	11p	15p	JD	11p	15p	JD	11p	15p	JD
avg	41.9	0	0	41.9	0	0.00	42.4	0.01	0.03	49.4	0.03	0.12	46.2	0.05	0.25
max	41.9	0	0	42.3	0.01	0.03	50.3	0.05	0.27	95.3	0.22	0.75	100	1.68	9.14
min	41.9	0	0	41.9	0	0	41.9	0	0	41.9	0	0	41.9	0	0

Figure 2: Summary of the accuracy comparison in % between the 11-point and 15-point ST quadrature vs the proposed JD integrator, for all the test-set geometries

In accordance with our expectation, the JD is exact for the regular mesh (Geom1); moreover, it is practically exact for the second geometry - probably the Jacobian function there is close to linear. While the 11-point ST does not admit even the regular mesh, the 15-point ST is practically exact. Errors are small relative to the associated errors in the stiffness matrix, especially with an extremely coarse mesh. Our JD, although slightly less accurate than the 15-point ST, is practically equivalent to it. The only geometry for which maximum errors cross the single-digit threshold is the fifth one. Clearly, no engineer would tolerate such a poor (and even ill- or singular-) mesh; it was created as a worst-case scenario. And still, even for the fifth geometry, maximum errors are contained in a rather reasonable range. Yes, a higher-order JD integrator can indeed be proposed by using a quadratic (or mixed) approximation for the Jacobian function (9). However, the above results convince us that higher-order pursuit will have academic but no practical meaning - the proposed JD formula is sufficiently accurate. Our goal is purely pragmatic: the least expensive yet accurate enough integrator. Finally, we recall that the JD is formulated to be exact for the regular mesh (constant metric/flat-faced elements) but also for the linear Jacobian; this is an even stricter demand than the well-known 'full' integration requirement.

5 Complexity and CPU time

In this section, our goal is to demonstrate quantitatively that the suggested JD integrator (11) is highly desirable in terms of its computational costs. To this end, we've studied the (relative) 'polynomial' complexity of the JD integrator vs the 1-point ST integration. Also, we've examined the subroutine running CPU time vs the 1-point ST subroutine. In both cases, according to our findings, the complete and sufficiently accurate JD formula is of the same order of magnitude in complexity. It has a comparable runtime to 1-point ST quadrature.

Complexity estimation is performed using CAS, specifically MAPLE. We use the built-in function named 'Complexity', which evaluates the (relative) complexity of a formula (expression/polynomial). We use this function in 'polynomial' mode, which treats all variables equally. The returned numerical values bear no absolute meaning; they are used to compare one value to another. Thus, we compute the complexity of 1-point ST and the complexity of a complete JD formula, and compare the results for rough efficiency estimation.

For the conventional numerical integration (4), we first must evaluate the complex-

ity of each component of the Jacobian matrix (1), i.e.,

$$J_{11} = N_{1;1}X_{11} + N_{2;1}X_{21} + N_{3;1}X_{31} + ... + N_{9;1}X_{91} + N_{10;1}X_{10\,1}$$
$$...$$
$$...$$
$$J_{33} = N_{1;3}X_{13} + N_{2;3}X_{23} + N_{3;3}X_{33} + ... + N_{9;3}X_{93} + N_{10;3}X_{10\,3}$$

All the variables are kept symbolic, so that the built-in function is not affected by the sudden vanishing of a particular number or mantissa length; it treats all variables equally. Then, the complexity associated with the determinant computation (2) is evaluated. Finally, the complexity associated with (4) is computed. While we do account for symmetry, e.g., we compute $M_{12}^{ST} = w_1\rho_0 N_1 N_2 J$, we assign $M_{21}^{ST} = M_{12}^{ST}$. All the complexities are added, and the total number is recorded.

For the JD approach, we add up all the complexities associated with entries of the partial derivatives of the Jacobian matrix (24) - (27). Then, the complexities associated with the Jacobian's partial derivatives (14) and (15). Eventually, complexities associated with the JD formula (11). Here too, we account for mass matrix symmetry. Also, we account for the zeros in the generalized weight matrices (28) - (31). After all, we have two numbers to compare. C_{1p}^{ST} denotes the complexity of 1-point ST numerical integration while C^{JD} stands for the complexity of the complete JD scheme. The ratio is given by $C^{JD}/C_{1p}^{ST} = 1.253$, which means that our sufficiently accurate JD formula is about 25% more complex than 1-point ST integration, which is meaningless in terms of accuracy (see Figure **2**). It is fair to say that for the 10-node tetrahedral element, both are roughly similar in terms of complexity. The same has been observed for the 20-node hexahedral element [20].

How many independent numbers must be remembered for the subroutine? For the JD integrator, we require only twelve numbers to be stored (20),(22). For each ST point, there is 1 weight, 10 shape functions at the integration point, and 10×3 shape functions' partial derivatives at the integration point. About 41 numbers for the 1-point ST vs 12 for the complete JD formula!

At last, we wished to ensure that our complexity computations translate to CPU time. To this end, we created two functions (subroutines). The implementation used automatic code generation via CAS, the same explicit expressions we used for complexity. There were no loops or matrix multiplications, which could distort the results, since one is faster than the other; only explicit closed-form expressions generated by a CAS were used. Obviously, we had to run both functions many times (100×10^6) to obtain sufficient CPU time for averaging and comparison. T_{avg}^{1pST} denotes the averaged time for 10^6 runs and T_{avg}^{JD} stands for the averaged time for 10^6 runs of the complete JD formula. Importantly, $T_{avg}^{JD}/T_{avg}^{1pST} = 1.293$, namely, the suggested, complete, sufficiently accurate JD model runs about 29% longer than 1-point ST integration. This result agrees with the previously reported complexity ratio.

Authors emphasize that they cannot conduct a comprehensive complexity study for the JD method. Also, it would be more convincing to use well-established codes for CPU time study. However, all comparisons (accuracy, complexity, and CPU time) were conducted fairly to both methods and to the best of our programming abilities.

6 Discussion and Conclusions

The main goal of the study is to derive a novel case-specific integrator tailored to the element-level consistent mass matrix of the 10-node tetrahedral element. To this end, the recently proposed J-Derivatives (JD) approach is adopted [20]. The report provides, for the first time, all the necessary technical terms for in-code implementation. In addition, we examine the accuracy, complexity, and CPU time of the suggested scheme.

The JD method exploits the mathematical structure of the integrand (3) and distinguishes between coordinate-dependent and mesh-dependent terms. However, the Jacobian function includes both; thus, it is systematically approximated using Taylor's multivariate expansion about the centroid. Then, the coordinate-dependent functions are analytically integrated to produce the generalized weight matrices (10). This makes the JD formula a linear combination of four matrices multiplied by the partial derivatives of the Jacobian function and the initial density (11). Importantly, the JD scheme is exact for the regular mesh (constant Jacobian/flat faced/constant metric/full integration) and for the linear Jacobian function. This establishes the suggested integrator as sufficiently accurate based on purely theoretical considerations. Also, it is non-Gaussian in the sense that no integration or sampling points are needed. Partial derivatives of the Jacobian function $\tilde{J}_q$ where $(q = 0, 1, 2, 3)$, are expressed in terms of the partial derivatives of the linearized Jacobian matrix (14), (15), and not in terms of the mesh. This intelligent choice ensures high efficiency of the proposed approach.

Accuracy of the derived integrator is examined based on five test geometries meshed using Abaqus automatic mesh generator [22], see Figure **1**. We compared the accuracy of the JD integrator with that of 11-point and 15-point conventional quadrature (cubature) numerical integration. Our conclusion supports the JD as an accurate enough formula, as its precision is practically equivalent to that of the well-established and widely used 15-point ST.

The complexity of the proposed JD integrator has been examined using the CAS built-in function and compared to the 1-point ST integration. We've chosen the 1-point conventional integration as the base for comparison, since we couldn't think of any less expensive options. Keep in mind that for practical applications, 15-point quadratures are used. According to our findings, our complete, sufficiently accurate JD formula is 25% more complex than the 1-point ST, making them roughly equivalent in complexity. In addition, we've implemented 1-point ST computation and the JD integrator to study CPU time. Accordingly, the complete JD required 29% more CPU time than the 1-point ST quadrature. Once again, it makes both of them "on the same league". In terms of memory - numbers to be remembered for the subroutine, the ST requires 41 numbers for each point while the complete JD formula only 12 given by (20),(22).

In conclusion, the proposed JD-based mass matrix integrator is found by us to be sufficiently accurate based on both theoretical considerations and practical comparison to the 15-point ST integration. Its complexity and CPU time were compared to the 1-point ST scheme and found to be of comparable magnitude (about 25-30% higher). At the same time, its memory requirements are much lower than those of the ST approach. Thus, the proposed formula is highly desirable and ready for im-

plementation in code.

7 Acknowledgments

I would like to express my deepest love and gratitude to those who have provided endless support and understanding, peace of mind, and belief in my work - Ruth, little Avital, and a tiny Orly.

8 Appendix A

Shape functions of the standard 10-node quadratic tetrahedral (e.g.[6]) are given by

$$\begin{array}{ccc} N_1 = (1-\xi-\eta-\zeta)\,(1-2\xi-2\eta-2\zeta) & & \\ N_2 = \xi\,(2\xi-1) & N_3 = \eta\,(2\eta-1) & N_4 = \zeta\,(2\zeta-1) \\ N_5 = 4\xi\,(1-\xi-\eta-\zeta) & N_6 = 4\xi\eta & N_7 = 4\eta\,(1-\xi-\eta-\zeta) \\ N_8 = 4\zeta\,(1-\xi-\eta-\zeta) & N_9 = 4\xi\zeta & N_{10} = 4\eta\zeta \end{array} \tag{23}$$

Partial derivatives of the Jacobian matrix $[J^q]$, $(q = 0, .., 3)$, given by (12), are explicitly evaluated using CAS

$$[J^0] = \begin{bmatrix} -X_{71}+X_{61}-X_{81}+X_{91} & -X_{51}+X_{61}-X_{81}+X_{10\,1} & X_{91}-X_{51}-X_{71}+X_{10\,1} \\ -X_{72}+X_{62}-X_{82}+X_{92} & -X_{52}+X_{62}-X_{82}+X_{10\,2} & X_{92}-X_{52}-X_{72}+X_{10\,2} \\ -X_{73}+X_{63}-X_{83}+X_{93} & -X_{53}+X_{63}-X_{83}+X_{10\,3} & X_{93}-X_{53}-X_{73}+X_{10\,3} \end{bmatrix} \tag{24}$$

$$[J^1] = c_1 \begin{bmatrix} -c_2X_{51}+X_{11}+X_{21} & -X_{71}+X_{11}-X_{51}+X_{61} & -X_{81}+X_{91}+X_{11}-X_{51} \\ -c_2X_{52}+X_{12}+X_{22} & -X_{72}+X_{12}-X_{52}+X_{62} & -X_{82}+X_{92}+X_{12}-X_{52} \\ -c_2X_{53}+X_{13}+X_{23} & -X_{73}+X_{13}-X_{53}+X_{63} & -X_{83}+X_{93}+X_{13}-X_{53} \end{bmatrix} \tag{25}$$

$$[J^2] = c_1 \begin{bmatrix} -X_{71}+X_{11}-X_{51}+X_{61} & -c_2X_{71}+X_{11}+X_{31} & -X_{71}-X_{81}+X_{10\,1}+X_{11} \\ -X_{72}+X_{12}-X_{52}+X_{62} & -c_2X_{72}+X_{12}+X_{32} & -X_{72}-X_{82}+X_{10\,2}+X_{12} \\ -X_{73}+X_{13}-X_{53}+X_{63} & -c_2X_{73}+X_{13}+X_{33} & -X_{73}-X_{83}+X_{10\,3}+X_{13} \end{bmatrix} \tag{26}$$

$$[J^3] = c_1 \begin{bmatrix} -X_{81}+X_{91}+X_{11}-X_{51} & -X_{71}-X_{81}+X_{10\,1}+X_{11} & -c_2X_{81}+X_{11}+X_{41} \\ -X_{82}+X_{92}+X_{12}-X_{52} & -X_{72}-X_{82}+X_{10\,2}+X_{12} & -c_2X_{82}+X_{12}+X_{42} \\ -X_{83}+X_{93}+X_{13}-X_{53} & -X_{73}-X_{83}+X_{10\,3}+X_{13} & -c_2X_{83}+X_{13}+X_{43} \end{bmatrix} \tag{27}$$

With the help of mass coefficients (22), the generalized weight matrices (21), take the form

$$M^{JD}_{IJ0} = \begin{bmatrix} m_7 & m_3 & m_3 & m_3 & -m_6 & -m_7 & -m_6 & -m_6 & -m_7 & -m_7 \\ m_3 & m_7 & m_3 & m_3 & -m_6 & -m_6 & -m_7 & -m_7 & -m_6 & -m_7 \\ m_3 & m_3 & m_7 & m_3 & -m_7 & -m_6 & -m_6 & -m_7 & -m_7 & -m_6 \\ m_3 & m_3 & m_3 & m_7 & -m_7 & -m_7 & -m_7 & -m_6 & -m_6 & -m_6 \\ -m_6 & -m_6 & -m_7 & -m_7 & m_{10} & m_9 & m_9 & m_9 & m_9 & m_8 \\ -m_7 & -m_6 & -m_6 & -m_7 & m_9 & m_{10} & m_9 & m_8 & m_9 & m_9 \\ -m_6 & -m_7 & -m_6 & -m_7 & m_9 & m_9 & m_{10} & m_9 & m_8 & m_9 \\ -m_6 & -m_7 & -m_7 & -m_6 & m_9 & m_8 & m_9 & m_{10} & m_9 & m_9 \\ -m_7 & -m_6 & -m_7 & -m_6 & m_9 & m_9 & m_8 & m_9 & m_{10} & m_9 \\ -m_7 & -m_7 & -m_6 & -m_6 & m_8 & m_9 & m_9 & m_9 & m_9 & m_{10} \end{bmatrix} \tag{28}$$

$$M^{JD}_{IJ1} = \begin{bmatrix} -m_2 & -m_1 & m_1 & m_1 & -m_3 & -m_2 & 0 & 0 & -m_2 & m_2 \\ -m_1 & m_4 & -m_1 & -m_1 & m_3 & m_3 & m_2 & m_2 & m_3 & m_2 \\ m_1 & -m_1 & -m_2 & m_1 & -m_2 & -m_3 & 0 & m_2 & -m_2 & 0 \\ m_1 & -m_1 & m_1 & -m_2 & -m_2 & -m_2 & m_2 & 0 & -m_3 & 0 \\ -m_3 & m_3 & -m_2 & -m_2 & m_6 & m_5 & 0 & 0 & m_5 & 0 \\ -m_2 & m_3 & -m_3 & -m_2 & m_5 & m_6 & 0 & 0 & m_5 & 0 \\ 0 & m_2 & 0 & m_2 & 0 & 0 & -m_6 & -m_5 & 0 & -m_5 \\ 0 & m_2 & m_2 & 0 & 0 & 0 & -m_5 & -m_6 & 0 & -m_5 \\ -m_2 & m_3 & -m_2 & -m_3 & m_5 & m_5 & 0 & 0 & m_6 & 0 \\ m_2 & m_2 & 0 & 0 & 0 & 0 & -m_5 & -m_5 & 0 & -m_6 \end{bmatrix} \tag{29}$$

$$M^{JD}_{IJ2} = \begin{bmatrix} -m_2 & m_1 & -m_1 & m_1 & 0 & -m_2 & -m_3 & 0 & m_2 & -m_2 \\ m_1 & -m_2 & -m_1 & m_1 & 0 & -m_3 & -m_2 & m_2 & 0 & -m_2 \\ -m_1 & -m_1 & m_4 & -m_1 & m_2 & m_3 & m_3 & m_2 & m_2 & m_3 \\ m_1 & m_1 & -m_1 & -m_2 & m_2 & -m_2 & -m_2 & 0 & 0 & -m_3 \\ 0 & 0 & m_2 & m_2 & -m_6 & 0 & 0 & -m_5 & -m_5 & 0 \\ -m_2 & -m_3 & m_3 & -m_2 & 0 & m_6 & m_5 & 0 & 0 & m_5 \\ -m_3 & -m_2 & m_3 & -m_2 & 0 & m_5 & m_6 & 0 & 0 & m_5 \\ 0 & m_2 & m_2 & 0 & -m_5 & 0 & 0 & -m_6 & -m_5 & 0 \\ m_2 & 0 & m_2 & 0 & -m_5 & 0 & 0 & -m_5 & -m_6 & 0 \\ -m_2 & -m_2 & m_3 & -m_3 & 0 & m_5 & m_5 & 0 & 0 & m_6 \end{bmatrix} \tag{30}$$

$$M^{JD}_{IJ3} = \begin{bmatrix} -m_2 & m_1 & m_1 & -m_1 & 0 & m_2 & 0 & -m_3 & -m_2 & -m_2 \\ m_1 & -m_2 & m_1 & -m_1 & 0 & 0 & m_2 & -m_2 & -m_3 & -m_2 \\ m_1 & m_1 & -m_2 & -m_1 & m_2 & 0 & 0 & -m_2 & -m_2 & -m_3 \\ -m_1 & -m_1 & -m_1 & m_4 & m_2 & m_2 & m_2 & m_3 & m_3 & m_3 \\ 0 & 0 & m_2 & m_2 & -m_6 & -m_5 & -m_5 & 0 & 0 & 0 \\ m_2 & 0 & 0 & m_2 & -m_5 & -m_6 & -m_5 & 0 & 0 & 0 \\ 0 & m_2 & 0 & m_2 & -m_5 & -m_5 & -m_6 & 0 & 0 & 0 \\ -m_3 & -m_2 & -m_2 & m_3 & 0 & 0 & 0 & m_6 & m_5 & m_5 \\ -m_2 & -m_3 & -m_2 & m_3 & 0 & 0 & 0 & m_5 & m_6 & m_5 \\ -m_2 & -m_2 & -m_3 & m_3 & 0 & 0 & 0 & m_5 & m_5 & m_6 \end{bmatrix} \tag{31}$$

References

[1] M. S. Alnæs and K.-A. Mardal. On the efficiency of symbolic computations combined with code generation for finite element methods. *ACM Transactions on Mathematical Software (TOMS)*, 37(1):1–26, 2010.

[2] P. Antolin and T. Hirschler. Quadrature-free immersed isogeometric analysis. *Engineering with Computers*, 38(5):4475–4499, 2022.

[3] M. Cerrolaza and J. Osorio. Relations among stiffness coefficients of hexahedral 8-noded finite elements: A simple and efficient way to reduce the integration time. *Finite elements in analysis and design*, 55:1–6, 2012.

[4] R. Cools. An encyclopaedia of cubature formulas. *Journal of Complexity*, 19(3):445–453, 2003. ISSN 0885-064X. doi: https://doi.org/10.1016/S0885-064X(03)00011-6. URL https://www.sciencedirect.com/science/article/pii/S0885064X03000116. Oberwolfach Special Issue.

[5] R. Cools. An encyclopaedia of cubature formulas. *Journal of complexity*, 19(3): 445–453, 2003.

[6] G. Dhondt. *The finite element method for three-dimensional thermomechanical applications*. John Wiley & Sons, 2004.

[7] C. A. Felippa. A compendium of fem integration formulas for symbolic work. *Engineering computations*, 21(8):867–890, 2004.

[8] J. A. González, R. Kolman, S. Cho, C. A. Felippa, and K. Park. Inverse mass matrix via the method of localized lagrange multipliers. *International Journal for Numerical Methods in Engineering*, 113(2):277–295, 2018.

[9] A. Gravouil, T. Elguedj, and H. Maigre. An explicit dynamics extended finite element method. part 2: Element-by-element stable-explicit/explicit dynamic scheme. *Computer Methods in Applied Mechanics and Engineering*, 198(30-32): 2318–2328, 2009.

[10] E. Hanukah. Development of a higher order closed-form model for isotropic hyperelastic cylindrical body, including small vibration problem. *arXiv preprint arXiv:1312.0083*, 2013.

[11] E. Hanukah. Higher order closed-form model for isotropic hyperelastic spherical shell (3d solid). *arXiv preprint arXiv:1401.0204*, 2013.

[12] E. Hanukah. A new closed-form model for isotropic elastic sphere including new solutions for the free vibrations problem. *arXiv preprint arXiv:1311.0741*, 2013.

[13] E. Hanukah. Exact integration scheme for six-node wedge element mass matrix. *arXiv preprint arXiv:1412.6538*, 2014.

[14] E. Hanukah. Consistent mass matrix of ten nodes tetrahedral element based on analytical integration. *arXiv preprint arXiv:1411.1341*, 2014.

[15] E. Hanukah. Mass matrix integration scheme for fifteen-node wedge element. *arXiv preprint arXiv:1501.00175*, 2014.

[16] E. Hanukah. Semi-analytical mass matrix for 8-node brick element. *arXiv preprint arXiv:1410.3195*, 2014.

[17] E. Hanukah. On semi-analytical integration specified for mass matrix of finite elements. *arXiv preprint arXiv:1506.02168*, 2015.

[18] E. Hanukah. *Analytical and Semi-Analytical Approaches Applied to the Nonlinear*

Mechanical Behavior of 3-D Structure. PhD thesis, Technion - Israel Institute of Technology, 2019.

[19] E. Hanukah. Semi-analytical approach to element-level integration for the solid nonlinear finite element. *engrXiv preprint engrXiv:10.31224/3231*, 2023.

[20] E. Hanukah. Element-level mass matrix integration. *engrXiv preprint engrXiv:10.31224/5624*, 2025.

[21] E. Hanukah and S. Givli. Free vibration of an isotropic elastic skewed parallelepiped–a closed form study. *European Journal of Mechanics-A/Solids*, 53: 131–142, 2015.

[22] E. Hanukah and S. Givli. A new approach to reduce the number of integration points in mass-matrix computations. *Finite Elements in Analysis and Design*, 116: 21–31, 2016.

[23] E. Hanukah and S. Givli. Improving mass matrix and inverse mass matrix computations of hexahedral elements. *Finite Elements in Analysis and Design*, 144: 1–14, 2018.

[24] E. Hanukah and B. Goldshtein. A structural theory for a 3d isotropic linear-elastic finite body. *arXiv preprint arXiv:1207.6767*, 2012.

[25] Z. He, E. Li, G. Liu, G. Li, and A. Cheng. A mass-redistributed finite element method (mr-fem) for acoustic problems using triangular mesh. *Journal of Computational Physics*, 323:149–170, 2016.

[26] H. E. Hinnant. A fast method of numerical quadrature for p-version finite element matrices. *International Journal for Numerical Methods in Engineering*, 37 (21):3723–3750, 1994.

[27] M. Jabareen. Keynote: The cosserat point element (cpe)–a new approach for finite element formulation. In *The 7th International Conference on Computational Methods (ICCM2016)*, 2016.

[28] M. Jabareen, E. Hanukah, and M. Rubin. A ten node tetrahedral cosserat point element (cpe) for nonlinear isotropic elastic materials. *Computational Mechanics*, 52:257–285, 2013.

[29] S. Johnson and T. Jeyapoovan. Evaluation of stiffness matrix in finite element analysis using element edge method for the 8-node brick element. *International Journal of Computational Materials Science and Engineering*, 8(04):1950018, 2019.

[30] S. Johnson, S. Sivakumar, and D. Nagarajan. A 13-point sampling point scheme for 20-node brick elements. *International Journal of Computational Materials Science and Engineering*, 10(04):2150019, 2021.

[31] S. Johnson, S. Sivakumar, and D. Nagarajan. A nine-point sampling point scheme for twenty node brick element. In *Journal of Physics: Conference Series*, volume 1913, page 012143. IOP Publishing, 2021.

[32] A. Korncoff and S. J. Fenves. Symbolic generation of finite element stiffness matrices. *Computers & structures*, 10(1-2):119–124, 1979.

[33] K. L. Lawrence, R. V. Nambiar, and B. Bergmann. Closed form stiffness matrices and error estimators for plane hierarchic triangular elements. *International journal for numerical methods in engineering*, 31(5):879–894, 1991.

[34] Y. Liang and J. McPhee. Symbolic integration of multibody system dynamics with the finite element method. *Multibody System Dynamics*, 43(4):387–405, 2018.

[35] G.-R. Liu and N. Trung. *Smoothed finite element methods*. CRC press, 2016.

[36] I. Lozada, J. Osorio, D. Griffiths, and M. Cerrolaza. Semi-analytical integration of the 8-node plane element stiffness matrix using symbolic computation. *Numerical Methods for Partial Differential Equations: An International Journal*, 22(2): 296–316, 2006.

[37] I. Lozada, D. Griffiths, and M. Cerrolaza. Semi-analytical integration of the elastic stiffness matrix of an axisymmetric eight-noded finite element. *Numerical Methods for Partial Differential Equations*, 26(6):1624–1635, 2010.

[38] S. E. McCaslin, P. S. Shiakolas, B. H. Dennis, and K. L. Lawrence. A new approach to obtaining closed-form solutions for higher order tetrahedral finite elements using modern computer algebra systems. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 54792, pages 225–231, 2011.

[39] S. E. McCaslin, P. S. Shiakolas, B. H. Dennis, and K. L. Lawrence. Closed-form stiffness matrices for higher order tetrahedral finite elements. *Advances in Engineering Software*, 44(1):75–79, 2012.

[40] M. A. Moetakef, K. L. Lawrence, S. P. Joshi, and P. S. Shiakolas. Closed-form expressions for higher order electroelastic tetrahedral elements. *AIAA Journal*, 33(1):136–142, 1995.

[41] J. C. Osorio, M. Cerrolaza, and M. Perez. Optimising the stiffness matrix integration of n-noded 3d finite elements. *International Journal of Computational Science and Engineering*, 16(2):173–180, 2018.

[42] M. Pavlović. Symbolic computation in structural engineering. *Computers & structures*, 81(22-23):2121–2136, 2003.

[43] Z. Pei, W. Xie, T. Suo, and Z. Xu. An inner-element edge-based smoothed finite element method. *Acta Mechanica Solida Sinica*, pages 1–10, 2025.

[44] F. P. Russell and P. H. Kelly. Optimized code generation for finite element local assembly using symbolic manipulation. *ACM Transactions on Mathematical Software (TOMS)*, 39(4):1–29, 2013.

[45] P. Shiakolas, R. Nambiar, K. Lawrence, and W. Rogers. Closed-form stiffness matrices for the linear strain and quadratic strain tetrahedron finite elements. *Computers & structures*, 45(2):237–242, 1992.

[46] S. Shun-Cheng and D. Zhu-Ping. Iterative method using consistent mass matrix in axisymmetrical finite element analysis of hypervelocity impact. *International journal of impact engineering*, 21(10):817–825, 1998.

[47] H. Songyang, W. Dongdong, W. Zhenyu, and L. Zhiwei. Precise mid-node lumped mass matrices for 3d 20-node hexahedral and 10-node tetrahedral finite elements. *Chinese Journal of Theoretical and Applied Mechanics*, pages 1–13, 2023.

[48] A. H. Stroud. *Approximate calculation of multiple integrals*. Prentice Hall, 1971.

[49] A. Tkachuk and M. Bischoff. Direct and sparse construction of consistent inverse mass matrices: general variational formulation and application to selective mass scaling. *International Journal for Numerical Methods in Engineering*, 101(6):435–469, 2015.

[50] J. D. Trotter. *High-performance finite element computations*. PhD thesis, Simula Research Laboratory, 2020.

[51] L. Videla, M. Cerrolaza, and N. Aparicio. Explicit integration of the stiffness ma-

trix of a four-noded-plane-elasticity finite element. *Communications in numerical methods in engineering*, 12(11):731–743, 1996.

[52] L. Videla, T. Baloa, D. Griffiths, and M. Cerrolaza. Exact integration of the stiffness matrix of an 8-node plane elastic finite element by symbolic computation. *Numerical Methods for Partial Differential Equations: An International Journal*, 24(1):249–261, 2008.

[53] D. Wan, D. Hu, G. Yang, and T. Long. A fully smoothed finite element method for analysis of axisymmetric problems. *Engineering Analysis with Boundary Elements*, 72:78–88, 2016.

[54] B. Wang, D. Babu, R. Nambiar, and K. Lawrence. Shape design sensitivity analysis using closed-form stiffness matrix for hierarchic triangular elements. *Computers & structures*, 43(1):69–75, 1992.

[55] P. Wriggers. *Nonlinear finite element methods*. Springer Science & Business Media, 2008.

[56] S. R. Wu and W. Qiu. Nonlinear transient dynamic analysis by explicit finite element with iterative consistent mass matrix. *Communications in numerical methods in engineering*, 25(3):201–217, 2009.

[57] W. Zeng and G. Liu. Smoothed finite element methods (s-fem): an overview and recent developments. *Archives of Computational Methods in Engineering*, 25(2): 397–435, 2018.

[58] O. Zienkiewicz, R. Taylor, and D. Fox. The finite element method for solid and structural mechanics elsevier, 2005.