Element-level mass matrix integration

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

Abstract

Many real-life engineering problems associated with solid mechanics are analyzed using the widely used Finite Element Method (FEM). An essential part of this numerical procedure is the evaluation of the element-level quantities, such as mass and stiffness matrices, internal and external forces, and other relevant functions. To this end, commercial packages adopt the Standard (ST) approach to numerical integration. Namely, a sufficiently accurate quadrature scheme (such as Gauss points) is employed. The higher the number of integration points, the higher the integration accuracy (integration order), yet the higher the complexity (i.e., the computational cost of the numerical procedure). Thus, practical guidance is to implement the least expensive quadrature (cubature) for which the global system converges. Therefore, deriving efficient and sufficiently accurate element-level integrators is an essential and practical goal.

In this contribution, we propose a novel approach to element-level mass matrix computation that does not rely on the notion of integration points. Instead, we exploit a special mathematical form of the integrand to derive a unique, highly efficient, case-specific integrator. Specifically, we rely on a systematic approximation of the Jacobian function in terms of its partial derivatives. Those partial derivative terms are efficiently expressed in terms of the systematic Jacobian matrix approximation. Finally, analytical integration is used to pre-compute the matrix coefficients (generalized weights). Our approach results in an easy-to-implement integration formula for the element mass matrix that is dramatically more efficient than the widely used ST method. Moreover, the suggested approach is roughly four times less computationally expensive than the recently proposed, state-of-the-art, Semi-Analytical (SA) approach for mass matrix evaluation. Finally, the recommended approach is general in the sense that it is suitable for all solid isoparametric elements regardless of their order (e.g., 8-node vs 20-node hexahedral, 6-node vs 15-node wedge) or dimension (e.g., quadrilateral & triangular vs. hexahedral & tetrahedral, etc). It allows for constant and varying initial density.

Finally, we specialize our model to a 20-node hexahedral element. We estimate the accuracy, the memory requirement, and the relative complexity of the resulting formula using the Computer Algebra System (CAS). In terms of relative 'polynomial' complexity, the resulting integrator is approximately equivalent to 1-point ST integration. In terms of accuracy, the proposed JD scheme is superior to the commercially used 14-point ST quadrature, which is sufficiently accurate for global convergence. In terms of memory storage, i.e., the number of constants which must be stored, the proposed model is several times less expensive than the 1-point ST. In conclusion, based on practical demonstration, our case-specific integration approach is a highly efficient and sufficiently accurate solution.

Key words: J-Derivatives (JD) approach; Semi-Analytical (SA); non-Gaussian; Finite Element Technology (FET); closed-form;

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

1 Introduction

Finite Element Technology (FET) is the practical implementation and advancement of the Finite Element Method (FEM) through software, algorithms, and computational tools to analyze and simulate physical systems. In particular, our study aims to develop sufficiently accurate and highly efficient element-level mass matrix integrators. Most commonly, the Standard (ST) approach to numerical integration (e.g. [3, 41]) is adopted. Achieving adequate accuracy (integration order) can become quite computationally expensive. As more accurate quadrature (cubature) is employed, it requires more integration points, increasing the computational cost of the integration. Thus, the practical rule of thumb is to use the least expensive quadrature for which the global system converges. Therefore, we must ensure that our integrator is both accurate enough and highly efficient.

Several approaches to non-standard integration exist. For example, symbolic computations with later code generation to raise the element-level integration efficiency can be found in [1, 37]. An additional technique successfully used to improve the efficiency of element-level stiffness matrices computation is the "Closed-Form" integration. Examples include plane elements [43, 32] and triangles [45], 3D elements such as hexahedral elements [2] or tetrahedral elements [34].

Recently, a highly efficient Semi-Analytical (SA) approach for a solid element mass matrix integration has been proposed [12, 13, 14, 15, 16, 20, 21, 18]. It identifies that the integrand is composed of two multiplicative parts. The first is merely a polynomial with respect to the local coordinate; it is mesh independent, while the other is the mesh-dependent part that must be treated differently from the first one. Then, the Jacobian function (the second part) is systematically approximated using sampling (integration) points and a polynomial ansatz function. Next, the resulting integrand undergoes analytical integration to precompute the generalized weights. Those generalized weight matrices are numbers that are straightforwardly implemented in the subroutine. The resulting formula for the SA mass matrix is a linear combination of weight matrices multiplied by the sampled Jacobian and initial density. It is easy for in-code implementation. Also, it has been shown that for 3D elements, such as an 8 & 20-node brick or a 10-node tetrahedral, the SA 4-point integrators are sufficiently accurate and require only four integration points, which is several times more efficient than the ST integration rules.

The described JD approach belongs to the SA family of mass matrix integrators, while demonstrating yet another substantial breakthrough in efficiency. Unlike the SA, the Jacobian function is systematically approximated in terms of its partial derivatives. Thus, there is no need for integration (sampling) points. The key is that a computationally efficient representation of those derivatives is given in terms of the systematically approximated Jacobian matrix and not in terms of the mesh (element nodal positions). Preliminary relative complexity study reveals that our linear model, which is as accurate as the 4-point SA model, is computationally equivalent to 1-point ST integration. Specifically, for the 20-node hexahedral element, JD replaces 14-point ST or 4-point SA schemes, maintaining accuracy equivalent to SA and surpassing the accuracy of 14-point schemes. Importantly, JD's computational complexity for a 20-node brick, as evaluated by CAS, is equivalent to a 1-point ST integrator. Also,

several times fewer constants must be kept in memory for the complete JD formula than for 1-point ST.

Following the above, the study is organized as follows: Section (2) recalls the necessary mathematical background of the consistent mass matrix formulation. Subsection (2.1) presents the commonly used approach to numerical integration. Section (3) presents the reader with the suggested J-Derivatives (JD) method, then subsection (3.1) generalizes the model for the varying (non-constant) initial density. Section (4) derives and records specific details for the 20-node hexahedral element mass matrix integration using the JD approach. Section (5) elaborates on the model's accuracy, based on a comparative study of several test-set geometries. Finally, section (6) studies the relative "polynomial" complexity, and section (7) summarizes the study, its main results, contribution, and practical importance. Appendix **A** keeps a record of technical details necessary for the specification of the JD to a 20-node brick element.

2 Mathematical problem statement

Let us consider the standard isoparametric displacement-based element (e.g. [46]). It is either a 2D namely $n_{dim} = 2$ or a 3D element $n_{dim} = 3$. Its number of nodes is denoted by n_{nodes} while its elemental number of degrees of freedom is given by n_{dof}. Clearly, the number of elemental degrees of freedom admits $n_{dof} = n_{nodes} \times n_{dim}$. Thus, the 6-node triangular element satisfies $n_{dof} = 6 \times 2 = 12$, while for the 8-node hexahedral $n_{dof} = 8 \times 3 = 24$. Elemental mass matrix $[M]$ is a square, nonsingular, positive definite matrix having the size of n_{dof}. However, it consists of $M_{IJ}, (I, J = 1, .., n_{nodes})$ independent values; moreover, M_{IJ} is also positive definite and symmetric $M_{IJ} = M_{JI}$. Hence, we are left with at most $(n_{nodes}^2 - n_{nodes})/2 + n_{nodes}$ independent values to be computed. Consequently, throughout the study, we'll work with the nodal (essential) mass matrix form $M_{I,J}$; however, for the sake of introductory completeness, the transition between the complete in-code n_{dof} form and the reduced nodal yet essential form is recalled

$$\underset{(i,j=1,..,n_{dof})}{[M]_{(i,j)}} = \left\{ \begin{array}{cc} & i = n_{dim}(I-1)+n \\ M_{IJ} & j = n_{dim}(J-1)+n \\ & (I,J = 1,..,n_{nodes}, n = 1,..,n_{dim}) \\ \\ \\ 0 & otherwise \end{array} \right\} \tag{1}$$

For a 2D case $n_{dim} = 2$, local coordinates $\boldsymbol{\xi}$ are denoted by $\boldsymbol{\xi} = (\xi, \eta)$, while for a 3D, namely, $n_{dim} = 3$, the local coordinates are given by $\boldsymbol{\xi} = (\xi, \eta, \zeta)$. Shape functions $N_I(\boldsymbol{\xi}), (I = 1, .., n_{nodes})$ are explicit polynomials in terms of local coordinates (e.g. 24). Initial elemental volume dV is given by $dV = J\ d\xi\ d\eta$, for 2D and $dV = J\ d\xi\ d\eta\ d\zeta$ for 3D case respectively. Here, J is the Jacobian function (or the metric), it is a function of the local coordinates and the nodal positions (the mesh), i.e., $J(\boldsymbol{\xi}; X_{kI})$. Where $X_{kI}, (k = 1, .., n_{dim}, I = 1, .., n_{nodes})$ stands for the position of element node I with respect to the global coordinate system, such as $(\boldsymbol{e_1}, .., \boldsymbol{e_m}), (m = 1, .., n_{dim})$ in the direction k. In other words, the position of an

arbitrary elemental node $(I, I = 1, .., n_{nodes})$ can be expressed as $\boldsymbol{X}_I = \sum_{k=1}^{ndim} X_{kI} \boldsymbol{e}_k$. Importantly, X_{kI} are numbers provided by the program (the mesher) to the integrator subroutine. For our purposes, those are part of an input. Here and throughout the study, lower indices run from 1 to n_{dim}, while upper indices change from 1 to n_{nodes}.

Additional convenient definition related to the parent element volume domain is $d\square$, such that for the 3D case $d\square = d\xi\ d\eta\ d\zeta$. Consequently, integration of an arbitrary function $A(\boldsymbol{\xi}; X_{kI})$ in the initial element domain V_e becomes $\int_{V_e} A(\boldsymbol{\xi}; X_{kI})\, dV$, and can be expressed as $\int_{V_\square} A(\boldsymbol{\xi}; X_{kI}) J(\boldsymbol{\xi}; X_{mJ})\, d\square$, here $(k, m = 1, .., n_{dim}, I, J = 1, .., n_{nodes})$. In particular, for the 3D hexahedral element, that will become important later, integration bounds are as follows $\int_{V_\square} (\cdot)\, d\square = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} (\cdot)\, d\xi d\eta d\zeta$. Throughout the study, we'll primarily refer to the 3D elements, as this is the most practical and important case. However, reducing to the 2D case is a straightforward dimension reduction, which is now clear. Entries of the consistent element mass matrix ([48]) are given by

$$\begin{gathered} M_{IJ} = \int_{V_e} \rho_0 N_I N_J\, dV = \int_{V_\square} \rho_0 N_I\left(\boldsymbol{\xi}\right) N_J\left(\boldsymbol{\xi}\right) J\left(\boldsymbol{\xi}; X_{mK}\right)\, d\square \\ M_{IJ} = M_{JI}, (m = 1, .., n_{dim}, I, J, K = 1, .., n_{nodes}) \end{gathered} \tag{2}$$

where ρ_0 is the initial element density. The matrix is symmetric and positive definite, which is quite natural given its physical meaning of mass.

While shape functions $N_I\left(\boldsymbol{\xi}\right)$ are merely polynomials (e.g. (24)), the Jacobian function $J\left(\boldsymbol{\xi}; X_{mK}\right)$ is mesh-dependent polynomial with respect to local coordinates $\boldsymbol{\xi}$, whose explicit representation includes a huge number of additive terms. For example, the Jacobian function of a 20-node hexahedral element consists of 443808 additive polynomial terms where monomials of local coordinates are multiplied by the mesh, e.g., $J = ... + 24\xi^4\eta\zeta(\frac{21}{16} X_{2,7} X_{3,13} X_{1,5} + ...) + ...)$. Thus, even though it is possible to integrate the above (2) analytically, the numerical cost associated with the resulting formula is unacceptable, and the accuracy is far beyond the minimal necessity. Thus, the Jucobian function will be systematically approximated; therefore, we had better recall how it is constructed and what it consists of.

Following the isoparametric formulation, an arbitrary material point X in the initial element domain, is given by $\boldsymbol{X} = \sum_{I=1}^{n_{nodes}} N_I\left(\boldsymbol{\xi}\right) \boldsymbol{X}_I$. Jacobian matrix $[J]$ having dimension $(n_{dim} \times n_{dim})$ is defined by $[J] = Grad_{\boldsymbol{\xi}} \boldsymbol{X} = \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}}$. Thus, a component of matrix $[J]$ becomes $X_{m;k} = \sum_{I=1}^{n_{nodes}} N_{I;k} X_{m,I}, (m, k = 1, .., n_{dim})$, where semicolon stands for partial differentiation for local coordinates $\boldsymbol{\xi}$. Accordingly, for the 3D case

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

$$\begin{gathered} J\left(\boldsymbol{\xi}, X_{m,I}\right) = Det\left([J]\right) = J_{1,1}J_{2,2}J_{3,3} - J_{1,1}J_{2,3}J_{3,2} - J_{1,2}J_{2,1}J_{3,3} + J_{1,2}J_{2,3}J_{3,1} \\ + J_{1,3}J_{2,1}J_{3,2} - J_{1,3}J_{2,2}J_{3,1} \end{gathered}$$

where $Det\left(\cdot\right)$ stands for the standard determinant definition. What is the order of the Jacobian matrix with respect to local coordinates? Clearly, it matches the order

of $N_{I;k}, (k = 1, .., n_{dim}, I = 1, .., n_{nodes})$. For example, for the 10-node tetrahedral element, shape functions are complete second-order polynomials. Thus, the Jacobian matrix is linear. However, for the 20-node hexahedral shape functions include even fourth order terms (24), so that the Jacobian matrix is third order with respect to $\boldsymbol{\xi}$. The order of the Jacobian function is 3 times the order of $N_{I;k}$, due to the determinant operation of a 3×3 matrix. Thus, even for a linear Jacobian matrix, the Jacobian function is third order with respect to local coordinates.

Given the above (3), it becomes clear that the integrand in (2) consists of polynomial terms with respect to local coordinates, hence, an exact integration is possible. However, the numerical cost associated with this operation is far beyond reasonable. In addition, achieving such accuracy for element-level mass matrix evaluation provides no practical benefits in the overall global scheme. Thus, commercial packages adopt the standard approach to numerical integration in the assessment of (2).

2.1 Standard (ST) approach to numerical integration

The ST method is based on a specific integration quadrature (cubature). Each scheme has its number of integration points n_{ip} and its rank (order). The quadrature of order k will exactly integrate a polynomial of rank k. Location of integration point is denoted by $\boldsymbol{\xi}_q = (\xi_q, \eta_q, \zeta_q)$, $(q = 1, .., n_{ip})$ while w_q are the associated weights. The ST approximation takes the form

$$\begin{gathered} M_{IJ} \approx M_{IJ}^{ST} = \sum_{q=1}^{n_{ip}} w_q \rho_{0q} N_{Iq} N_{Jq} J_q \left(X_{mK}\right) \\ (m = 1, .., n_{dim}, I, J, K = 1, .., n_{nodes}) \end{gathered} \tag{4}$$

where ρ_{0q} is the evaluation of the initial density at point $\boldsymbol{\xi}_q$, shape functions calculated at integration points are denoted by $N_{Iq} = N_I\left(\boldsymbol{\xi}_q\right)$, the same applies to the Jacobian function at an integration point $J_q = J\left(\boldsymbol{\xi}_q, X_{mK}\right)$, $(q = 1, .., n_{ip})$. The higher the n_{ip} in the above (4), the more multiplications and additions must be performed, which increases the computational cost of the procedure.

$$M_{IJ}^{ST} = w_1 \rho_{01} N_{I1} N_{J1} J_1 + w_2 \rho_{02} N_{I2} N_{J2} J_2 + ... + w_{n_{ip}} \rho_{0n_{ip}} N_{In_{ip}} N_{Jn_{ip}} J_{n_{ip}}$$

Reducing the computational costs associated with element-level integration is so essential that the widely accepted rule-of-thumb guides practitioners to employ the lowest n_{ip} possible as long as the global system converges. Namely, our alternative integrator must be sufficiently accurate and highly efficient.

For later convenience, one can view the above integrator (4) from an alternative perspective. By adopting the next definition, the ST approach (4) can be represented as a linear combination of weight matrices multiplied by sampled function $\rho_{0q} J_q, (q = 1, .., n_{ip})$.

$$\begin{gathered} M_{IJq}^{ST} = w_q N_{Iq} N_{Jq} \\ M_{IJ}^{ST} = \sum_{q=1}^{n_{ip}} M_{IJq}^{ST} \, \rho_{0q} \, J_q \quad (I, J = 1, .., n_{nodes}) \end{gathered} \tag{5}$$

Here, M_{IJq}^{ST} can be viewed as weight matrices. Moreover, given the above formalism (5), one might wonder whether better weights exist. Namely, can one propose different generalized weight matrices, such that the above summation requires fewer terms (integration points), while having better accuracy? Recently, significantly better coefficients have been found using two distinct novel approaches - the SA and the Optimization-Based (OB) approach [19, 21].

3 J-Derivatives (JD) approach

Generally speaking, the proposed JD method can be viewed as belonging to the Semi-Analytical (SA) family of element-level mass matrix integrators developed recently [12, 13, 14, 15, 16, 20, 21, 18]. It combines a systematic analytical approximation for the Jacobian with later analytical integration for the generalized coefficients to derive a case-specific numerical integration scheme. We say 'case-specific' to stress that those integrators are suitable for solid element mass matrix only; as such, they can not be used for internal forces or a stiffness matrix evaluation. However, a dramatic step forward in efficiency is possible due to a significantly more efficient approximation that does not require sampling points.

First, for the sake of simplicity, let us assume that the initial density in the element domain is constant $\rho_0 = const$. This assumption agrees with most practical analyses (since the mesh is fine). However, we'll close the section by accounting for varying initial density $\rho_0 \neq const$, see subsection (3.1). We will show that it introduces no theoretical or computational difficulties, does not change the mathematical structure of the resulting integrator, and adds no notable computational costs.

Let us focus on the integrand in (2). Clearly, it consists of two multiplicative parts

$$\rho_0 \quad [\, N_I(\boldsymbol{\xi})\, N_J(\boldsymbol{\xi}) \,] \quad [\, J(\boldsymbol{\xi}; X_{mK}) \,], \qquad (m = 1, .., n_{dim}, I, J, K = 1, .., n_{nodes})$$

while the first one is simple mesh-independent polynomial $\rho_0 N_I(\boldsymbol{\xi})\, N_J(\boldsymbol{\xi})$, (e.g. 24), the second one is lengthy mesh-dependent polynomial function $J(\boldsymbol{\xi}; X_{mK})$, see (3). The second term is systematically approximated by means of Taylor's multivariable expansion about the element's centroid $\boldsymbol{\xi}^* = (\xi = \xi^*, \eta = \eta^*, \zeta = \zeta^*)$

$$\begin{gathered} J(\boldsymbol{\xi}; X_{mK}) \approx J\big|_{\boldsymbol{\xi}^*} + \\ (\xi - \xi^*)\, \frac{\partial J}{\partial \xi}\Big|_{\boldsymbol{\xi}^*} + (\eta - \eta^*)\, \frac{\partial J}{\partial \eta}\Big|_{\boldsymbol{\xi}^*} + (\zeta - \zeta^*)\, \frac{\partial J}{\partial \zeta}\Big|_{\boldsymbol{\xi}^*} \end{gathered} \tag{6}$$

In the above (6), we've explicitly accounted for the zeroth and the first order. Clearly, the higher the approximation order, the better the accuracy; however, the more terms included, the more computationally complex the problem becomes. Here, linear approximation is enough to demonstrate the general idea behind the method. In addition, it appears that the linear approximation of J yields a sufficiently accurate

integration formula. Thus, with the help of the following definitions

$$\begin{aligned} &\tilde{N}_0 = 1, \quad \tilde{N}_1 = (\xi - \xi^*), \quad \tilde{N}_2 = (\eta - \eta^*), \quad \tilde{N}_3 = (\zeta - \zeta^*) \\ &\tilde{J}_0 = J\big|_{\boldsymbol{\xi}^*} \qquad \tilde{J}_1 = \frac{\partial J}{\partial \xi}\Big|_{\boldsymbol{\xi}^*} \qquad \tilde{J}_2 = \frac{\partial J}{\partial \eta}\Big|_{\boldsymbol{\xi}^*} \qquad \tilde{J}_3 = \frac{\partial J}{\partial \zeta}\Big|_{\boldsymbol{\xi}^*} \end{aligned} \tag{7}$$

approximation (6) takes the form

$$\begin{gathered} J \approx \tilde{N}_0\tilde{J}_0 + \tilde{N}_1\tilde{J}_1 + \tilde{N}_2\tilde{J}_2 + \tilde{N}_3\tilde{J}_3 = \sum_{q=0}^{3} \tilde{N}_q\left(\boldsymbol{\xi}\right)\tilde{J}_q\left(X_{mI}\right) \\ (m = 1, .., n_{dim}, I = 1, .., n_{nodes}) \end{gathered} \tag{8}$$

The above approximation (8) achieves clear separation between a simple dependency on the local coordinates $\boldsymbol{\xi}$ and dependency on the element's mesh (X_{mI}). Now, we install (8) in the original mass matrix definition (2). Consequently, the JD approximation to the element mass matrix is written as

$$\begin{aligned} M_{IJ}^{JD} &= \rho_0 \sum_{q=0}^{3} M_{IJq}^{JD}\, \tilde{J}_q \\ M_{IJq}^{JD} &= \int_{V_\square} N_I N_J \tilde{N}_q \, d\square \qquad (I, J = 1, .., n_{nodes}) \end{aligned} \tag{9}$$

It is important to emphasize that M_{IJq}^{JD} are generalized weights, those are four symmetric matrices $(q = 0, .., 3)$ full of real numbers. In that sense, the JD model is a linear combination of weight matrices multiplied by Jacobian derivatives. For each element type, those numbers are precomputed and then incorporated into the subroutine. For example, for the 10-node tetrahedral element, integration bounds are given by $\int_0^1 \int_0^{1-\zeta} \int_0^{1-\eta-\zeta} (\cdot)\, d\xi d\eta d\zeta$. In addition, most values repeat themselves (e.g., Section (4)); thus, efficient implementation is possible.

Notably, the above (9) admits exactly the regular mesh (constant Jacobian function/full integration) and the linear J elements, while it is accurate for all the other cases. Then, one might ask what the order of the above integrator is? This question can be answered only in a loose sense, and it depends on the elements. For example, for the 10-node tetrahedral element (5), the shape functions are quadratic with respect to $\boldsymbol{\xi}$; thus, their multiplication is a fourth order, and an additional order follows from the linear J. In fact, all the coarse mesh elements are approximated to linear J elements. The higher the order of the shape functions, the higher the 'order' of the scheme. In a case of a 20-node brick, the shape functions (24) are fourth order, thus, loosely speaking, the integrator is ninth order. Finally, we note that based on previous findings (20, 21) admitting linear J exactly is sufficiently accurate for global system convergence.

The key question now is how to express partial derivatives $\tilde{J}_q\left(X_{mK}\right)$, $(q = 0, .., 3)$, $(m = 1, .., n_{dim}, K = 1, .., n_{nodes})$ such that the overall computational cost of (9) will be superior? Its accuracy is designed to be adequate, yet the computational cost must also be superior. The principal remedy is to express $\tilde{J}_q$ as a function of the entries of partial derivatives of the Jacobian matrix and not as an explicit function of the mesh

X_{mK}. Importantly, it is significantly more beneficial to evaluate partial derivatives of the Jacobian matrix as a function of mesh (X_{mK}), and then to evaluate partial derivatives $\tilde{J}_q$. Next, we'll illustrate this point rigorously.

Let's consider a Taylor's multivariable expansion of the Jacobian matrix (3) about the centroid $\boldsymbol{\xi}^* = (\xi = \xi^*, \eta = \eta^*, \zeta = \zeta^*)$

$$[J(\boldsymbol{\xi}; X_{mK})] \approx [J]\big|_{\boldsymbol{\xi}^*} + (\xi - \xi^*)\frac{\partial [J]}{\partial \xi}\Big|_{\boldsymbol{\xi}^*} + (\eta - \eta^*)\frac{\partial [J]}{\partial \eta}\Big|_{\boldsymbol{\xi}^*} + (\zeta - \zeta^*)\frac{\partial [J]}{\partial \zeta}\Big|_{\boldsymbol{\xi}^*}$$

Practically, it is sufficiently accurate to consider only the linear part. This study goes no further than linear approximation to J, see (6,7), thus no higher than linear terms in the above are necessary. Clearly, for higher-order integrators, one can consider additional terms (3). The more terms one considers, the more accurate the resulting model becomes, but also the more expensive it is. Importantly, partial derivatives $\tilde{J}_q, (q = 0, 1, 2, 3)$, which will be derived based on the linear Jacobian matrix, are exact. With the help of definition (7)a the following definition

$$\left[J^0\right] = [J]\big|_{\boldsymbol{\xi}^*} \qquad \left[J^1\right] = \frac{\partial [J]}{\partial \xi}\Big|_{\boldsymbol{\xi}^*} \qquad \left[J^2\right] = \frac{\partial [J]}{\partial \eta}\Big|_{\boldsymbol{\xi}^*} \qquad \left[J^3\right] = \frac{\partial [J]}{\partial \zeta}\Big|_{\boldsymbol{\xi}^*} \tag{10}$$

A linear approximation to the Jacobian matrix is represented by

$$[J(\boldsymbol{\xi}; X_{mI})] \approx \tilde{N}_0\left[J^0\right] + \tilde{N}_1\left[J^1\right] + \tilde{N}_2\left[J^2\right] + \tilde{N}_3\left[J^3\right] = \sum_{q=0}^{3} \tilde{N}_q(\boldsymbol{\xi})\left[\tilde{J}_q(X_{mI})\right]$$
$$(m = 1, .., n_{dim}, I = 1, .., n_{nodes}) \tag{11}$$

For the 3D case, determinant of the above approximation (11), together with partial derivatives definition (7),(10) yield

$$\begin{aligned}\tilde{J}_0 = Det\left(\left[J^0\right]\right) = J^0_{1,1}J^0_{2,2}J^0_{3,3} - J^0_{1,1}J^0_{2,3}J^0_{3,2} - J^0_{1,2}J^0_{2,1}J^0_{3,3} + \\ J^0_{1,2}J^0_{2,3}J^0_{3,1} + J^0_{1,3}J^0_{2,1}J^0_{3,2} - J^0_{1,3}J^0_{2,2}J^0_{3,1}\end{aligned} \tag{12}$$

Determinant of the linear approximation of the Jacobian matrix (11), together with partial derivatives definition (7),(10) results in

$$\begin{aligned}\tilde{J}_m = J^0_{1,1}J^0_{2,2}J^m_{3,3} - J^0_{1,1}J^0_{2,3}J^m_{3,2} - J^0_{1,1}J^0_{3,2}J^m_{2,3} + J^0_{1,1}J^0_{3,3}J^m_{2,2} - J^0_{1,2}J^0_{2,1}J^m_{3,3} + \\ J^0_{1,2}J^0_{2,3}J^m_{3,1} + J^0_{1,2}J^0_{3,1}J^m_{2,3} - J^0_{1,2}J^0_{3,3}J^m_{2,1} + J^0_{1,3}J^0_{2,1}J^m_{3,2} - J^0_{1,3}J^0_{2,2}J^m_{3,1} - \\ J^0_{1,3}J^0_{3,1}J^m_{2,2} + J^0_{1,3}J^0_{3,2}J^m_{2,1} + J^0_{2,1}J^0_{3,2}J^m_{1,3} - J^0_{2,1}J^0_{3,3}J^m_{1,2} - J^0_{2,2}J^0_{3,1}J^m_{1,3} + \\ J^0_{2,2}J^0_{3,3}J^m_{1,1} + J^0_{2,3}J^0_{3,1}J^m_{1,2} - J^0_{2,3}J^0_{3,2}J^m_{1,1}\end{aligned}$$
$$(m = 1, 2, 3) \tag{13}$$

Then, the least expensive and least accurate JD integrator possible is based on the assumption of a constant Jacobian (6)(7)(8), i.e., (9) takes the form

$$M^{JD}_{IJ} = \rho_0\, M^{JD}_{IJ0}\, \tilde{J}_0 \qquad M^{JD}_{IJ0} = \int_{V_\square} N_I N_J\, d\square \qquad (I, J = 1, .., n_{nodes}) \tag{14}$$

Then, for the linearly varying J, using (7)a and (8), the JD (9) is given by

$$\begin{aligned} M^{JD}_{IJ0} &= \int_{V_\square} N_I N_J \, d\square \qquad & M^{JD}_{IJ1} &= \int_{V_\square} N_I N_J \left(\xi - \xi^*\right) \, d\square \\ M^{JD}_{IJ2} &= \int_{V_\square} N_I N_J \left(\eta - \eta^*\right) \, d\square \qquad & M^{JD}_{IJ3} &= \int_{V_\square} N_I N_J \left(\zeta - \zeta^*\right) \, d\square \\ M^{JD}_{IJ} &= \rho_0 \sum_{q=0}^{3} M^{JD}_{IJq} \, \tilde{J}_q \qquad & (I, J &= 1, .., n_{nodes}) \end{aligned} \tag{15}$$

Importantly, partial derivatives $\tilde{J}_q, (q = 0, 1, 2, 3)$, given by (12) and (13) are exact; they are not affected by the fact that we used linear approximation for the Jacobian matrix (11), they would be the same for higher order $[J]$ approximation, thus, for our practical purposes higher orders of $[J]$ are redundant. The above integrator (15) is exact for the linear Jacobian; for all other configurations, it assumes the Jacobian to be linear. According to our previous findings [20, 21], this level of accuracy is sufficient, and practically speaking, there is no need to waste additional computational effort. In 3D, partial derivatives $\tilde{J}_q, (q = 1, 2, 3)$ given by (13) include 18 additive terms, while J_0 given by (3) consist of only 6 terms. However, it will become apparent in the next Section (4) that the components of the Jacobian matrix are significantly more expensive than those of the partial derivatives matrices, which are essential for the proposed JD. That'll make the above extremely efficient, see Section (6).

3.1 Generalization for the varying initial density

Extension of the JD model to allow for the non-constant initial density $\rho_0 \neq const$ poses no theoretical difficulty and is not associated with meaningful computational costs. We'll demonstrate it based on the most practical linear model promoted here. First, we allow for linearly varying initial density. To this end, we construct an elemental linear model for ρ_0. Then, we approximate the multiplication $\rho_0 J$ rather than only J.

Consider, for instance, a 10-node tetrahedral element, with four of its first nodes at the element vertices (5) and its centroid $\left(\xi^* = \frac{1}{4}, \eta^* = \frac{1}{4}, \zeta^* = \frac{1}{4}\right)$. Let us denote the initial density at node m by $\rho_{0m}, (m = 1, .., 4)$; those values are known input to the subroutine. Then, let us assume a linear density distribution in the element domain, i.e.

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

Now, we can use an analytical approximation for the multiplication $\rho_0 J$ and not merely for J

$$\begin{gathered} \rho_0\left(\boldsymbol{\xi}\right) J\left(\boldsymbol{\xi}; X_{mK}\right) \approx \left(\rho_0 J\right)_{\boldsymbol{\xi}^*} + \\ \left(\xi - \xi^*\right) \frac{\partial\left(\rho_0 J\right)}{\partial \xi}\Big|_{\boldsymbol{\xi}^*} + \left(\eta - \eta^*\right) \frac{\partial\left(\rho_0 J\right)}{\partial \eta}\Big|_{\boldsymbol{\xi}^*} + \left(\zeta - \zeta^*\right) \frac{\partial\left(\rho_0 J\right)}{\partial \zeta}\Big|_{\boldsymbol{\xi}^*} \\ \left(m = 1, .., n_{dim}, K = 1, .., n_{nodes}\right) \end{gathered} \tag{17}$$

Now, with the help of the following definitions

$$\begin{gathered}\tilde{\rho}_1 = \frac{1}{4}\left(\rho_{01} + \rho_{02} + \rho_{03} + \rho_{04}\right), \quad \tilde{\rho}_m = \rho_{0m} - \rho_{01}, \quad (m = 2, 3, 4) \\ \tilde{J}_0^{\rho} = \tilde{\rho}_1 J_0 \\ \tilde{J}_1^{\rho} = \tilde{\rho}_2 J_0 + \tilde{\rho}_1 J_1, \qquad \tilde{J}_2^{\rho} = \tilde{\rho}_3 J_0 + \tilde{\rho}_1 J_2, \qquad \tilde{J}_3^{\rho} = \tilde{\rho}_4 J_0 + \tilde{\rho}_1 J_3\end{gathered} \tag{18}$$

The linear JD model (15) takes the form

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

The above (19) possesses the same mathematical structure as the original (15). Generalized weight matrices M_{IJq}^{JD} are identical. All additional computational costs are associated with definitions (18), which add only several multiplications and additions. Nevertheless, this formulation does include the notion of sampling points, since $\rho_{0m}, (m = 1, .., 4)$ are values sampled at the nodes (specific points). Yet, the mathematical structure, coefficient matrices, and computational costs remain; no major changes to the subroutine except very minor (18).

4 Practical example: 20-node hexahedral element

Let's examine the standard widely used 20-node brick element (5). Its centroid is given by $\boldsymbol{\xi}^* = (\xi = 0, \eta = 0, \zeta = 0)$. Its shape functions are detailed in (24). Explicit integration bounds are as follows $\int_{V_\square}(\cdot)\, d\square = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1}(\cdot)\, d\xi d\eta d\zeta$. First, one must derive the partial derivatives matrices $[J^m], (m = 0, 1, 2, 3)$ of the Jacobian matrix (3), given by (10). Here, we present only the typical components, while Appendix **A** records all the details; see (25), (26), and (27). With the help of the next constant definition

$$c_1 = \frac{1}{8}, \quad c_2 = 2, \quad c_3 = \frac{1}{4} \tag{20}$$

Typical components take the form

$$\begin{gathered}J_{11}^0 = c_1(\, c_2(\, X_{1\,10} - X_{1\,12} + X_{1\,14} - X_{1\,16} - X_{1\,17} + X_{1\,18} + X_{1\,19} - X_{1\,20}) + \\ X_{1\,1} - X_{1\,2} - X_{1\,3} + X_{1\,4} + X_{1\,5} - X_{1\,6} - X_{1\,7} + X_{1\,8}) \\ J_{11}^1 = c_3(c_2(-X_{1\,11} - X_{1\,13} - X_{1\,15} - X_{1\,9}) + X_{1\,1} + X_{1\,2} + X_{1\,3} + X_{1\,4} + X_{1\,5} + \\ X_{1\,6} + X_{1\,7} + X_{1\,8}) \\ J_{11}^2 = c_3(X_{1\,17} - X_{1\,18} + X_{1\,19} - X_{1\,20}) \\ J_{11}^3 = c_3(-X_{1\,10} + X_{1\,12} + X_{1\,14} - X_{1\,16})\end{gathered}$$

Importantly, only three constants (20) are necessary for the evaluation of partial Jacobian matrices. The above expressions are an output of the Computer Algebra System (CAS); they are pre-derived and implemented into the subroutine. Now, partial derivatives $\tilde{J}_q, (q = 0, 1, 2, 3)$ are evaluated using (12),(13); Theese expressions are valid for all 3D elements. Generalized coefficient matrices $M_{IJq}^{JD}, (q = 0, 1, 2, 3)$ are

computed (15) with the help of shape functions (24) and definition (7). Importantly, there are four generalized coefficient matrices, but they consist of only 17 independent values given by

$$\begin{array}{ccccc} m_1 = \dfrac{28}{135}, & m_2 = \dfrac{64}{135}, & m_3 = \dfrac{22}{135}, & m_4 = \dfrac{4}{27}, & m_5 = \dfrac{17}{135} \\ m_6 = \dfrac{32}{135}, & m_7 = -\dfrac{26}{135}, & m_8 = -\dfrac{2}{15}, & m_9 = \dfrac{8}{27}, & m_{10} = -\dfrac{26}{135} \\ m_{11} = \dfrac{16}{135}, & m_{12} = \dfrac{2}{45}, & m_{13} = \dfrac{4}{135}, & m_{14} = \dfrac{1}{27}, & m_{15} = \dfrac{2}{27} \\ & m_{16} = \dfrac{2}{135}, & m_{17} = \dfrac{8}{135} & & \end{array} \tag{21}$$

Explicit form of the generalized coefficient matrices $M^{JD}_{IJq}, (q = 0, 1, 2, 3)$ in terms of the above values (21) are detailed in (29), (30), (31), and (32). To complete the entire mass matrix evaluation using a sufficiently accurate JD formula, one has to store only $17 + 3 = 20$ independent values (20), (21). While using the ST approach (4), (3), for only one integration point, which is meaningless in terms of accuracy, one must store $N_{I1}, (I = 1, .., 20)$ together with $N_{I;\boldsymbol{\xi}}, (I = 1, .., 20, \xi = 1, 2, 3)$ evaluated at the point, namely additional 60 numbers!

For the varying initial density (see subsection (3.1)), we adopt the assumption of linear density distribution in the initial element domain (18). Thus, additional input is given by

$$\begin{array}{cc} \rho_{00} = \rho_0\big|_{(\xi=0,\eta=0,\zeta=0)} & \rho_{01} = \rho_0\big|_{(\xi=1,\eta=0,\zeta=0)} \\ \rho_{02} = \rho_0\big|_{(\xi=0,\eta=1,\zeta=0)} & \rho_{03} = \rho_0\big|_{(\xi=0,\eta=0,\zeta=1)} \end{array} \tag{22}$$

With the help of the following definition, the JD becomes (19)

$$\begin{array}{c} \tilde{\rho}_0 = \dfrac{1}{4}\left(\rho_{00} + \rho_{01} + \rho_{02} + \rho_{03}\right), \quad \tilde{\rho}_m = \left(\rho_{0m} - \rho_{00}\right), \quad (m = 1, 2, 3) \\ \tilde{J}^{\rho}_0 = \tilde{\rho}_0 \tilde{J}_0, \quad \tilde{J}^{\rho}_n = \tilde{\rho}_n \tilde{J}_0 + \tilde{\rho}_0 \tilde{J}_n, \quad (n = 1, 2, 3) \end{array} \tag{23}$$

Once again, the varying density generalization added only several operations associated with the above definitions, no notable computational cost, no theoretical difficulty, and no change to the integrator structure or its generalized coefficient matrices.

5 Comparative accuracy study

Now, when the proposed model has been established both theoretically and in all of its practical details, it is essential to demonstrate its accuracy. In our opinion, the most convincing way is to run global system simulations. Then, it will become clear that our element-level integrator yields a sufficiently accurate global mass matrix. However, this approach is beyond the scope of our study. Nevertheless, we'll adopt an alternative attitude. First, we identify a practical 14-point quadrature adopted by one of the leading commercial packages, ANSYS. Clearly, it has been established (by ANSYS) that this scheme is sufficiently accurate for the global system convergence; otherwise, it would not be implemented. Then, we'll demonstrate based on element-level computation that our JD model is superior in accuracy. Then, in the next Section (6),

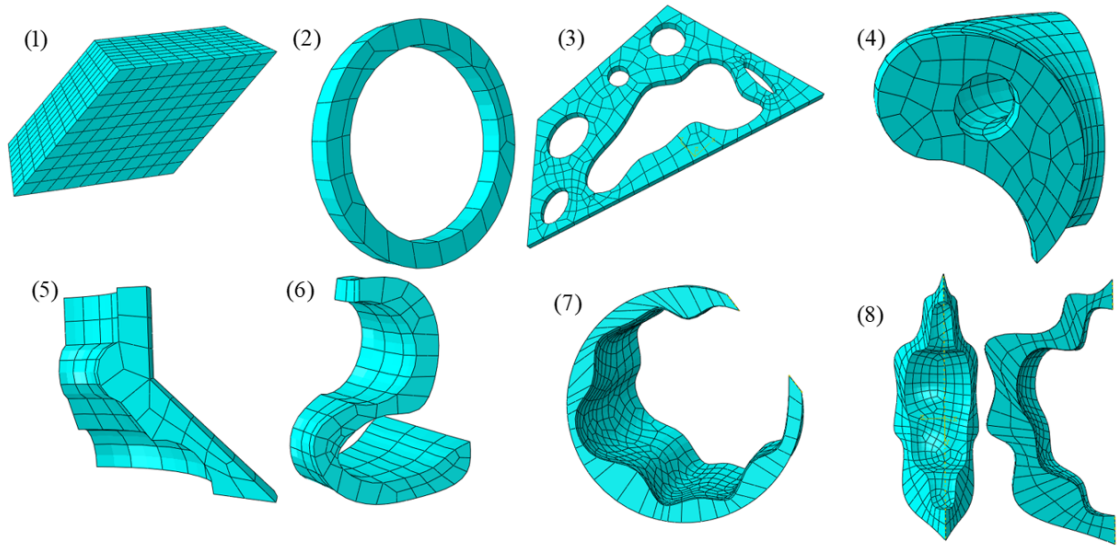

Figure 1: Showing the eight test-set geometries generated with the help of ABAQUS automatic mesh generator

	G1	G2	G3	G4	G5	G6	G7	G8
# Elements	1000	30	198	376	102	126	357	372
# Nodes	4961	310	1731	2087	692	890	2702	2307

Table 1: - records the number of elements and nodes for each test geometry G1,..,G8

we'll demonstrate its unbelievable efficiency dominance.

To this end, eight test geometries have been generated and meshed with the help of the ABAQUS automatic mesh generator, see Fig. (1). The general idea was to make the mesh gradually worse. While the first geometry includes a regular (constant Jacobian function) mesh, the last geometry consists of several elements for which the Jacobian vanishes at several points inside the element domain. Table (1) details the number of elements and nodes at each test geometry.

First, for every geometry, all elemental mass matrices have been evaluated exactly, so that we'll be able to estimate errors % exactly. Then, all elemental mass matrices have been evaluated using the 14-point ST and the JD approaches. Finally, for every elemental mass matrix, we evaluated the averaged (among its components) error %. Table (2) summarizes for each geometry, the average of elemental matrices error %, the minimum averaged error %, and the maximum averaged error %, i.e., the worst matrix evaluation in this geometry.

Based on the first geometry results, it is evident that the 14-point quadrature

	Geom 1		Geom 2		Geom 3		Geom 4		Geom 5		Geom 6		Geom 7		Geom 8	
	14-p ST	JD	14-p ST	JD	14-p ST	JD	14-p ST	JD	14-p ST	JD	14-p ST	JD	14-p ST	JD	14-p ST	JD
Avg:	5.6	0.0	5.7	1.4	5.7	0.8	5.7	1.0	5.7	1.8	5.7	1.4	5.8	0.9	5.8	1.0
Max:	5.6	0.0	5.9	4.5	7.2	7.1	5.9	6.1	6.2	6.2	6.1	4.3	7.6	4.7	6.8	6.2
Min:	5.6	0.0	5.4	0.2	5.4	0	5.3	0.1	5.3	0.2	5.4	0.0	5.5	0.1	5.5	0.1

Figure 2: Summary of the accuracy comparison in % between the 14-point ST quadrature and the proposed JD scheme, for all the test-set geometries

does not admit even the regular (constant Jacobian function) mesh. Recall that the JD is built in to be exact for the constant and linear Jacobian function; in that sense, it satisfies the "full integration" requirement. In addition, for all the geometries, JD demonstrates superior accuracy. Importantly, our model excels in terms of accuracy over the standard 14-point quadrature, which must be sufficiently accurate; otherwise, it would not be accepted by the industry's giants.

6 Complexity

Comprehensive complexity analysis of the promoted method is beyond the scope of this Section (6). However, we consider a practical case, such as a 20-node hexahedral element. Based on that particular widely used element, we estimate the memory necessary to store the essential constants and the relative 'polynomial' complexity. Relative 'polynomial' complexity is evaluated using the CAS system (MAPLE). We compare the ST approach using **one** integration point (4) vs the complete, sufficiently accurate, JD formula (15).

First, we wish to know how many numbers must be stored for 1-point ST integration (4), (3)? There are n_{nodes} shape functions N_I evaluated at the first integration point $\boldsymbol{\xi}^1 = (\xi = \xi^1, \eta = \eta^1, \zeta = \zeta^1)$, given by N_{I1} i.e., at most 20 distinct values plus one value for the associated weight w_1. In addition, there are $n_{nodes} \times n_{dim} = 60$ partial derivatives $N_{I;\boldsymbol{\xi}}$ evaluated at the integration point $N_{I;\boldsymbol{\xi}}\big|_{\boldsymbol{\xi}^1}$. Thus, overall, the 1-point ST method requires at most $20 + 1 + 60 = 81$ constants to be stored. Lastly, we recall once again that the least expensive quadrature used by a practical (ANSYS) implementation is the 14-point scheme.

For the complete, sufficiently accurate JD formula (9), two sets of coefficients are required. The first includes **three** distinct coefficients (20) for the $[\tilde{J}^m],\ (m = 0, 1, 2, 3)$ computation given by (25)-(28). The other (21) consists of 17 distinct coefficient needed for the generalized coefficient matrices $M^{JD}_{IJq}, (m = 0, 1, 2, 3)$ given by (29)-(32). Therefore, the suggested JD formula (9) requires only $3 + 17 = 20$ coefficients to be stored, which is about four times fewer than 1-point ST quadrature.

Now we consider two flow charts; the first Figure (3) details the 1-point ST pro-

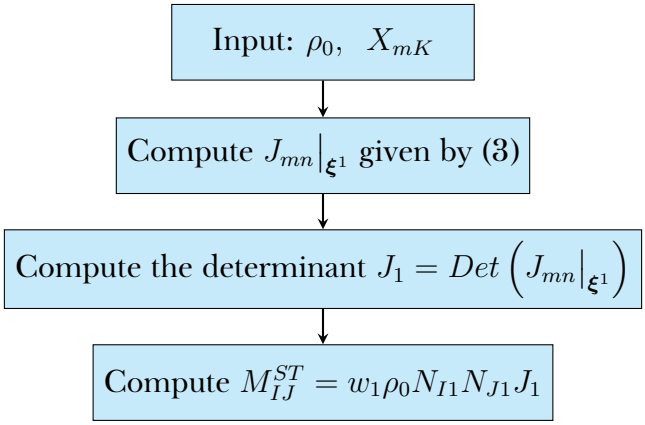

Figure 3: Flow diagram of the ST integrator using only **one** integration point

cedure, while the other Figure (4) details the necessary steps for the complete, sufficiently accurate JD formula. We evaluate the relative 'polynomial' complexity associated with each line and add it up. The resulting numbers possess no objective meaning beyond quantitative comparison between the two methods.

The overall relative complexity associated with 1-point ST is 139892, while the overall relative complexity resulting from the JD evaluation is 158707. Thus, according to our findings, for the 20-node brick element, our JD model possesses a computational cost equivalent to 1-point ST, i.e., $\frac{158707}{139892} = 1.134$. At the same time, it is superior in performance to the 14-point ST rule, see Table (2).

It is important to recall that the more points used for the standard approach, the more complexity and memory requirements grow. The least amount of points acceptable for a 20-node brick, provided the standard approach, is 14! Our integrator overperforms the 14-point ST in accuracy, but also requires about four times fewer numbers to be remembered than a single integration point and is roughly as complex as a single integration point.

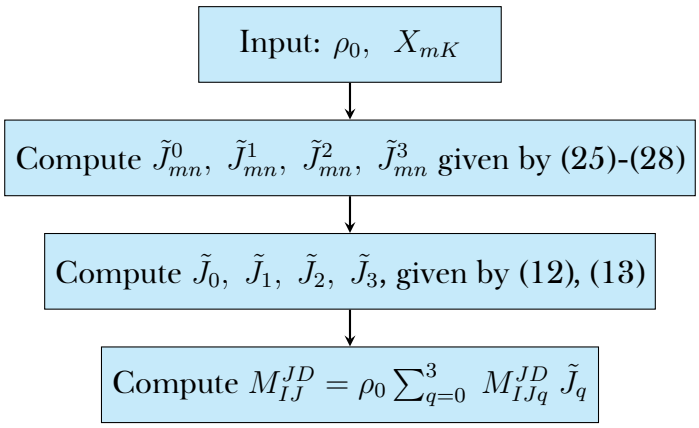

Figure 4: Flow diagram of the **complete** JD formula

7 Summary and Conclusions

Our contribution concentrates on the field of solid mechanics, Finite Element Technology (FET), specifically element-level integration for the mass matrix. We propose a novel, highly efficient method that is not based on the notion of integration points. The suggested J-Derivatives (JD) approach applies to all solid elements, whether 2D or 3D, and to first-order or higher-order elements, consistent or diagonal (lumped) formulation, allowing for both constant and varying density. Specifying the proposed method for a particular element, such as a 20-node brick in this study, results in a case-specific integrator, which is a numerical scheme (formula) used to evaluate the element mass matrix.

Formulation-wise, the JD approach is a further advancement of the recently developed Semi-Analytical (SA) family of element-level mass matrix integrators. Similarly to SA, it identifies and exploits the internal integrand's structure, consisting of two multiplicative parts (mesh-dependent and mesh-independent). In addition, it also

uses a systematic approximation of the Jacobian function, followed by an analytical integration for the generalized coefficient matrices. However, the dramatic efficiency increase, compared to the SA (and the ST) approach, is possible because the partial derivatives of the Jacobian function are intelligently represented in terms of the entries of the approximated Jacobian matrix, rather than in terms of the mesh.

For a practical demonstration, we apply the method to the widely used 20-node hexahedral element to derive all the necessary details for in-code implementation. To this end, a linear approximation of the Jacobian function is found to be sufficiently accurate. Clearly, one can consider higher orders of approximations, leading to even more accurate integrators, though they are more computationally expensive. Our goal is to present the least costly and sufficiently accurate integrator. Importantly, the presented derivation admits exactly the regular mesh (constant Jacobian function/full integration) and the linear Jacobian function. One interested in higher accuracy integrators follows the same systematic JD guidelines, up to an entirely exact scheme (which is obviously redundant for practical use).

We study the complexity, the memory requirements, and the accuracy of the resulting JD formula. The relative 'polynomial' complexity of the JD and ST approaches is evaluated using CAS, comparing the proposed JD integrator to the 1-point ST approach. Remarkably, the complete JD integrator is roughly as complex as only 1-point ST integration ($JD/ST = 1.13$). Then, we count how many constants have to be stored in memory for the complete JD formula vs the ST 1-point integration. Impressively, the complete JD requires about four times fewer independent numbers to be stored. Finally, we demonstrate that our JD scheme is more accurate than the currently employed 14-point quadrature (cubature) by leading commercial software (ANSYS). Here we must recall that for the ST integration, complexity and memory requirement grow linearly with the number of integration points, and the least acceptable number of points is 14! All those results demonstrate dramatic JD superiority to the ST numerical integration approach. Practically speaking, for the 20-node hexahedral element, a sufficiently accurate element-level consistent mass matrix can be evaluated using computational effort of only *ONE* integration point.

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] 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.

[3] 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.

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

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

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

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

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

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

[42] 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.

[43] L. Videla, M. Cerrolaza, and N. Aparicio. Explicit integration of the stiffness matrix of a four-noded-plane-elasticity finite element. *Communications in numerical methods in engineering*, 12(11):731–743, 1996.

[44] 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.

[45] 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.

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

[47] 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.

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

8 Appendix A

This Appendix keeps a record of all the necessary details for practical JD implementation for the 20-node brick element (e.g. [46]). Shape functions are given by

$$
\begin{aligned}
N_1 &= \frac{1}{8}(-1+\xi)(-1+\eta)(-1+\zeta)(2+\xi+\eta+\zeta)\\
N_2 &= \frac{1}{8}(1+\xi)(-1+\eta)(-1+\zeta)(-2+\xi-\eta-\zeta)\\
N_3 &= -\frac{1}{8}(1+\xi)(1+\eta)(-1+\zeta)(-2+\xi+\eta-\zeta)\\
N_4 &= -\frac{1}{8}(-1+\xi)(1+\eta)(-1+\zeta)(2+\xi-\eta+\zeta)\\
N_5 &= -\frac{1}{8}(-1+\xi)(-1+\eta)(1+\zeta)(2+\xi+\eta-\zeta)\\
N_6 &= -\frac{1}{8}(1+\xi)(-1+\eta)(1+\zeta)(-2+\xi-\eta+\zeta)\\
N_7 &= \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)(-2+\xi+\eta+\zeta)\\
N_8 &= \frac{1}{8}(-1+\xi)(1+\eta)(1+\zeta)(2+\xi-\eta-\zeta)
\end{aligned}
\tag{24}
$$

$$
N_9 = -\frac{1}{4}(-1+\eta)(-1+\zeta)\left(\xi^2-1\right), \quad N_{10} = \frac{1}{4}(1+\xi)(-1+\zeta)\left(\eta^2-1\right)
$$

$$
N_{11} = \frac{1}{4}(1+\eta)(-1+\zeta)\left(\xi^2-1\right), \quad N_{12} = -\frac{1}{4}(-1+\xi)(-1+\zeta)\left(\eta^2-1\right)
$$

$$
N_{13} = \frac{1}{4}(-1+\eta)(1+\zeta)\left(\xi^2-1\right), \qquad N_{14} = -\frac{1}{4}(1+\xi)(1+\zeta)\left(\eta^2-1\right)
$$

$$
N_{15} = -\frac{1}{4}(1+\eta)(1+\zeta)\left(\xi^2-1\right), \qquad N_{16} = \frac{1}{4}(-1+\xi)(1+\zeta)\left(\eta^2-1\right)
$$

$$
N_{17} = -\frac{1}{4}(-1+\xi)(-1+\eta)\left(\zeta^2-1\right), \quad N_{18} = \frac{1}{4}(1+\xi)(-1+\eta)\left(\zeta^2-1\right)
$$

$$
N_{19} = -\frac{1}{4}(1+\xi)(1+\eta)\left(\zeta^2-1\right), \qquad N_{20} = \frac{1}{4}(-1+\xi)(1+\eta)\left(\zeta^2-1\right)
$$

Partial derivatives matrices $[J^m], (m = 0, 1, 2, 3)$ given by (10) of the Jacobian matrix (3) must be evaluated. We use CAS for explicit expressions derivation with later

incorporation as is, into subroutines. Coefficients are listed in (20). $[J^0]$ is given by

$$\begin{gathered}
J^0_{11} = c_1(c_2(X_{1,10} - X_{1,12} + X_{1,14} - X_{1,16} - X_{1,17} + X_{1,18} + X_{1,19} - X_{1,20}) + \\
X_{1,1} - X_{1,2} - X_{1,3} + X_{1,4} + X_{1,5} - X_{1,6} - X_{1,7} + X_{1,8}) \\
J^0_{12} = c_1(c_2(X_{1,11} - X_{1,13} + X_{1,15} - X_{1,17} - X_{1,18} + X_{1,19} + X_{1,20} - X_{1,9}) + \\
X_{1,1} + X_{1,2} - X_{1,3} - X_{1,4} + X_{1,5} + X_{1,6} - X_{1,7} - X_{1,8}) \\
J^0_{13} = c_1(c_2(-X_{1,10} - X_{1,11} - X_{1,12} + X_{1,13} + X_{1,14} + X_{1,15} + X_{1,16} - X_{1,9}) + \\
X_{1,1} + X_{1,2} + X_{1,3} + X_{1,4} - X_{1,5} - X_{1,6} - X_{1,7} - X_{1,8}) \\
J^0_{21} = c_1(c_2(X_{2,10} - X_{2,12} + X_{2,14} - X_{2,16} - X_{2,17} + X_{2,18} + X_{2,19} - X_{2,20}) + \\
X_{2,1} - X_{2,2} - X_{2,3} + X_{2,4} + X_{2,5} - X_{2,6} - X_{2,7} + X_{2,8}) \\
J^0_{22} = c_1(c_2(X_{2,11} - X_{2,13} + X_{2,15} - X_{2,17} - X_{2,18} + X_{2,19} + X_{2,20} - X_{2,9}) + \\
X_{2,1} + X_{2,2} - X_{2,3} - X_{2,4} + X_{2,5} + X_{2,6} - X_{2,7} - X_{2,8}) \\
J^0_{23} = c_1(c_2(-X_{2,10} - X_{2,11} - X_{2,12} + X_{2,13} + X_{2,14} + X_{2,15} + X_{2,16} - X_{2,9}) + \\
X_{2,1} + X_{2,2} + X_{2,3} + X_{2,4} - X_{2,5} - X_{2,6} - X_{2,7} - X_{2,8}) \\
J^0_{31} = c_1(c_2(X_{3,10} - X_{3,12} + X_{3,14} - X_{3,16} - X_{3,17} + X_{3,18} + X_{3,19} - X_{3,20}) + \\
X_{3,1} - X_{3,2} - X_{3,3} + X_{3,4} + X_{3,5} - X_{3,6} - X_{3,7} + X_{3,8}) \\
J^0_{32} = c_1(c_2(X_{3,11} - X_{3,13} + X_{3,15} - X_{3,17} - X_{3,18} + X_{3,19} + X_{3,20} - X_{3,9}) + \\
X_{3,1} + X_{3,2} - X_{3,3} - X_{3,4} + X_{3,5} + X_{3,6} - X_{3,7} - X_{3,8}) \\
J^0_{33} = c_1(c_2(-X_{3,10} - X_{3,11} - X_{3,12} + X_{3,13} + X_{3,14} + X_{3,15} + X_{3,16} - X_{3,9}) + \\
X_{3,1} + X_{3,2} + X_{3,3} + X_{3,4} - X_{3,5} - X_{3,6} - X_{3,7} - X_{3,8})
\end{gathered} \tag{25}$$

$[J^1]$ takes the next form

$$\begin{gathered}
J^1_{11} = c_3(c_2(-X_{1,11} - X_{1,13} - X_{1,15} - X_{1,9}) + \\
X_{1,1} + X_{1,2} + X_{1,3} + X_{1,4} + X_{1,5} + X_{1,6} + X_{1,7} + X_{1,8}) \\
J^1_{12} = c_3(c_2(X_{1,17} - X_{1,18} + X_{1,19} - X_{1,20}) \\
J^1_{13} = c_3(c_2(-X_{1,10} + X_{1,12} + X_{1,14} - X_{1,16}) \\
J^1_{21} = c_3(c_2(-X_{2,11} - X_{2,13} - X_{2,15} - X_{2,9}) + \\
X_{2,1} + X_{2,2} + X_{2,3} + X_{2,4} + X_{2,5} + X_{2,6} + X_{2,7} + X_{2,8}) \\
J^1_{22} = c_3(c_2(X_{2,17} - X_{2,18} + X_{2,19} - X_{2,20}) \\
J^1_{23} = c_3(c_2(-X_{2,10} + X_{2,12} + X_{2,14} - X_{2,16}) \\
J^1_{31} = c_3(c_2(-X_{3,11} - X_{3,13} - X_{3,15} - X_{3,9}) + \\
X_{3,1} + X_{3,2} + X_{3,3} + X_{3,4} + X_{3,5} + X_{3,6} + X_{3,7} + X_{3,8}) \\
J^1_{32} = c_3(c_2(X_{3,17} - X_{3,18} + X_{3,19} - X_{3,20}) \\
J^1_{33} = c_3(c_2(-X_{3,10} + X_{3,12} + X_{3,14} - X_{3,16})
\end{gathered} \tag{26}$$

$[J^2]$ becomes

$$\begin{gathered}
J^2_{11} = c_3(X_{1,17} - X_{1,18} + X_{1,19} - X_{1,20}) \\
J^2_{12} = c_3(c_2(-X_{1,10} - X_{1,12} - X_{1,14} - X_{1,16}) + \\
X_{1,1} + X_{1,2} + X_{1,3} + X_{1,4} + X_{1,5} + X_{1,6} + X_{1,7} + X_{1,8}) \\
J^2_{13} = c_3(-X_{1,11} - X_{1,13} + X_{1,15} + X_{1,9}) \\
J^2_{21} = c_3(X_{2,17} - X_{2,18} + X_{2,19} - X_{2,20}) \\
J^2_{22} = c_3(c_2(-X_{2,10} - X_{2,12} - X_{2,14} - X_{2,16}) + \\
X_{2,1} + X_{2,2} + X_{2,3} + X_{2,4} + X_{2,5} + X_{2,6} + X_{2,7} + X_{2,8}) \\
J^2_{23} = c_3(-X_{2,11} - X_{2,13} + X_{2,15} + X_{2,9}) \\
J^2_{31} = c_3(X_{3,17} - X_{3,18} + X_{3,19} - X_{3,20}) \\
J^2_{32} = c_3(c_2(-X_{3,10} - X_{3,12} - X_{3,14} - X_{3,16}) + \\
X_{3,1} + X_{3,2} + X_{3,3} + X_{3,4} + X_{3,5} + X_{3,6} + X_{3,7} + X_{3,8}) \\
J^2_{33} = c_3(-X_{3,11} - X_{3,13} + X_{3,15} + X_{3,9})
\end{gathered} \tag{27}$$

Finally, $[J^3]$ is given by

$$\begin{gathered}
J^3_{11} = c_3(-X_{1,10} + X_{1,12} + X_{1,14} - X_{1,16}) \\
J^3_{12} = c_3(-X_{1,11} - X_{1,13} + X_{1,15} + X_{1,9}) \\
J^3_{13} = c_3(c_2(-X_{1,17} - X_{1,18} - X_{1,19} - X_{1,20}) + \\
X_{1,1} + X_{1,2} + X_{1,3} + X_{1,4} + X_{1,5} + X_{1,6} + X_{1,7} + X_{1,8}) \\
J^3_{21} = c_3(-X_{2,10} + X_{2,12} + X_{2,14} - X_{2,16}) \\
J^3_{22} = c_3(-X_{2,11} - X_{2,13} + X_{2,15} + X_{2,9}) \\
J^3_{23} = c_3(c_2(-X_{2,17} - X_{2,18} - X_{2,19} - X_{2,20}) + \\
X_{2,1} + X_{2,2} + X_{2,3} + X_{2,4} + X_{2,5} + X_{2,6} + X_{2,7} + X_{2,8}) \\
J^3_{31} = c_3(-X_{3,10} + X_{3,12} + X_{3,14} - X_{3,16}) \\
J^3_{32} = c_3(-X_{3,11} - X_{3,13} + X_{3,15} + X_{3,9}) \\
J^3_{33} = c_3(c_2(-X_{3,17} - X_{3,18} - X_{3,19} - X_{3,20}) + \\
X_{3,1} + X_{3,2} + X_{3,3} + X_{3,4} + X_{3,5} + X_{3,6} + X_{3,7} + X_{3,8})
\end{gathered} \tag{28}$$

Generalized coefficient matrices $M^{JD}_{IJq}, (q = 0, 1, 2, 3)$ depend on only 17 independent values $(m_1, .., m_{17})$ listed in (21).

$$M^{JD}_{IJ0} =$$

$$\begin{bmatrix}
m_1 & m_3 & m_4 & m_3 & m_3 & m_4 & m_5 & m_4 & -m_6 & m_7 & m_7 & -m_6 & m_7 & m_8 & m_8 & m_7 & -m_6 & m_7 & m_8 & m_7 \\
m_3 & m_1 & m_3 & m_4 & m_4 & m_3 & m_4 & m_5 & -m_6 & -m_6 & m_7 & m_7 & m_7 & m_7 & m_8 & m_8 & m_7 & -m_6 & m_7 & m_8 \\
m_4 & m_3 & m_1 & m_3 & m_5 & m_4 & m_3 & m_4 & m_7 & -m_6 & -m_6 & m_7 & m_8 & m_7 & m_7 & m_8 & m_8 & m_7 & -m_6 & m_7 \\
m_3 & m_4 & m_3 & m_1 & m_4 & m_5 & m_4 & m_3 & m_7 & m_7 & -m_6 & -m_6 & m_8 & m_8 & m_7 & m_7 & m_7 & m_8 & m_7 & -m_6 \\
m_3 & m_4 & m_5 & m_4 & m_1 & m_3 & m_4 & m_3 & m_7 & m_8 & m_8 & m_7 & -m_6 & m_7 & m_7 & -m_6 & -m_6 & m_7 & m_8 & m_7 \\
m_4 & m_3 & m_4 & m_5 & m_3 & m_1 & m_3 & m_4 & m_7 & m_7 & m_8 & m_8 & -m_6 & -m_6 & m_7 & m_7 & m_7 & -m_6 & m_7 & m_8 \\
m_5 & m_4 & m_3 & m_4 & m_4 & m_3 & m_1 & m_3 & m_8 & m_7 & m_7 & m_8 & m_7 & -m_6 & -m_6 & m_7 & m_8 & m_7 & -m_6 & m_7 \\
m_4 & m_5 & m_4 & m_3 & m_3 & m_4 & m_3 & m_1 & m_8 & m_8 & m_7 & m_7 & m_7 & m_7 & -m_6 & -m_6 & m_7 & m_8 & m_7 & -m_6 \\
-m_6 & -m_6 & m_7 & m_7 & m_7 & m_7 & m_8 & m_8 & m_2 & m_9 & m_6 & m_9 & m_6 & m_4 & m_{11} & m_4 & m_9 & m_9 & m_4 & m_4 \\
m_7 & -m_6 & -m_6 & m_7 & m_8 & m_7 & m_7 & m_8 & m_9 & m_2 & m_9 & m_6 & m_4 & m_6 & m_4 & m_{11} & m_4 & m_9 & m_9 & m_4 \\
m_7 & m_7 & -m_6 & -m_6 & m_8 & m_8 & m_7 & m_7 & m_6 & m_9 & m_2 & m_9 & m_{11} & m_4 & m_6 & m_4 & m_4 & m_4 & m_9 & m_9 \\
-m_6 & m_7 & m_7 & -m_6 & m_7 & m_8 & m_8 & m_7 & m_9 & m_6 & m_9 & m_2 & m_4 & m_{11} & m_4 & m_6 & m_9 & m_4 & m_4 & m_9 \\
m_7 & m_7 & m_8 & m_8 & -m_6 & -m_6 & m_7 & m_7 & m_6 & m_4 & m_{11} & m_4 & m_2 & m_9 & m_6 & m_9 & m_9 & m_9 & m_4 & m_4 \\
m_8 & m_7 & m_7 & m_8 & m_7 & -m_6 & -m_6 & m_7 & m_4 & m_6 & m_4 & m_{11} & m_9 & m_2 & m_9 & m_6 & m_4 & m_9 & m_9 & m_4 \\
m_8 & m_8 & m_7 & m_7 & m_7 & m_7 & -m_6 & -m_6 & m_{11} & m_4 & m_6 & m_4 & m_6 & m_9 & m_2 & m_9 & m_4 & m_4 & m_9 & m_9 \\
m_7 & m_8 & m_8 & m_7 & -m_6 & m_7 & m_7 & -m_6 & m_4 & m_{11} & m_4 & m_6 & m_9 & m_6 & m_9 & m_2 & m_9 & m_4 & m_4 & m_9 \\
-m_6 & m_7 & m_8 & m_7 & -m_6 & m_7 & m_8 & m_7 & m_9 & m_4 & m_4 & m_9 & m_9 & m_4 & m_4 & m_9 & m_2 & m_6 & m_{11} & m_6 \\
m_7 & -m_6 & m_7 & m_8 & m_7 & -m_6 & m_7 & m_8 & m_9 & m_9 & m_4 & m_4 & m_9 & m_9 & m_4 & m_4 & m_6 & m_2 & m_6 & m_{11} \\
m_8 & m_7 & -m_6 & m_7 & m_8 & m_7 & -m_6 & m_7 & m_4 & m_9 & m_9 & m_4 & m_4 & m_9 & m_9 & m_4 & m_{11} & m_6 & m_2 & m_6 \\
m_7 & m_8 & m_7 & -m_6 & m_7 & m_8 & m_7 & -m_6 & m_4 & m_4 & m_9 & m_9 & m_4 & m_4 & m_9 & m_9 & m_6 & m_{11} & m_6 & m_2
\end{bmatrix} \tag{29}$$

$$M_{IJ1}^{JD} =$$

$$\begin{bmatrix}
-m_{12} & 0 & 0 & -m_{13} & -m_{13} & 0 & 0 & -m_{14} & 0 & -m_{13} & m_{16} & m_{15} & m_{16} & -m_{16} & m_{16} & m_{15} & m_{15} & -m_{13} & -m_{16} & m_{15} \\
0 & m_{12} & m_{13} & 0 & 0 & m_{13} & m_{14} & 0 & 0 & -m_{15} & -m_{16} & m_{13} & -m_{16} & -m_{15} & -m_{16} & m_{16} & m_{13} & -m_{15} & -m_{15} & m_{16} \\
0 & m_{13} & m_{12} & 0 & 0 & m_{14} & m_{13} & 0 & -m_{16} & -m_{15} & 0 & m_{13} & -m_{16} & -m_{15} & -m_{16} & m_{16} & m_{16} & -m_{15} & -m_{15} & m_{13} \\
-m_{13} & 0 & 0 & -m_{12} & -m_{14} & 0 & 0 & -m_{13} & m_{16} & -m_{13} & 0 & m_{15} & m_{16} & -m_{16} & m_{16} & m_{15} & m_{15} & -m_{16} & -m_{13} & m_{15} \\
-m_{13} & 0 & 0 & -m_{14} & -m_{12} & 0 & 0 & -m_{13} & m_{16} & -m_{16} & m_{16} & m_{15} & 0 & -m_{13} & m_{16} & m_{15} & m_{15} & -m_{13} & -m_{16} & m_{15} \\
0 & m_{13} & m_{14} & 0 & 0 & m_{12} & m_{13} & 0 & -m_{16} & -m_{15} & -m_{16} & m_{16} & 0 & -m_{15} & -m_{16} & m_{13} & m_{13} & -m_{15} & -m_{15} & m_{16} \\
0 & m_{14} & m_{13} & 0 & 0 & m_{13} & m_{12} & 0 & -m_{16} & -m_{15} & -m_{16} & m_{16} & -m_{16} & -m_{15} & 0 & m_{13} & m_{16} & -m_{15} & -m_{15} & m_{13} \\
-m_{14} & 0 & 0 & -m_{13} & -m_{13} & 0 & 0 & -m_{12} & m_{16} & -m_{16} & m_{16} & m_{15} & m_{16} & -m_{13} & 0 & m_{15} & m_{15} & -m_{16} & -m_{13} & m_{15} \\
0 & 0 & -m_{16} & m_{16} & m_{16} & -m_{16} & -m_{16} & m_{16} & 0 & m_{17} & 0 & -m_{17} & 0 & m_{13} & 0 & -m_{13} & -m_{17} & m_{17} & m_{13} & -m_{13} \\
-m_{13} & -m_{15} & -m_{15} & -m_{13} & -m_{16} & -m_{15} & -m_{15} & -m_{16} & m_{17} & m_{6} & m_{17} & 0 & m_{13} & m_{11} & m_{13} & 0 & 0 & m_{4} & m_{4} & 0 \\
m_{16} & -m_{16} & 0 & 0 & m_{16} & -m_{16} & -m_{16} & m_{16} & 0 & m_{17} & 0 & -m_{17} & 0 & m_{13} & 0 & -m_{13} & -m_{13} & m_{13} & m_{17} & -m_{17} \\
m_{15} & m_{13} & m_{13} & m_{15} & m_{15} & m_{16} & m_{16} & m_{15} & -m_{17} & 0 & -m_{17} & -m_{6} & -m_{13} & 0 & -m_{13} & -m_{11} & -m_{4} & 0 & 0 & -m_{4} \\
m_{16} & -m_{16} & -m_{16} & m_{16} & 0 & 0 & -m_{16} & m_{16} & 0 & m_{13} & 0 & -m_{13} & 0 & m_{17} & 0 & -m_{17} & -m_{17} & m_{17} & m_{13} & -m_{13} \\
-m_{16} & -m_{15} & -m_{15} & -m_{16} & -m_{13} & -m_{15} & -m_{15} & -m_{13} & m_{13} & m_{11} & m_{13} & 0 & m_{17} & m_{6} & m_{17} & 0 & 0 & m_{4} & m_{4} & 0 \\
m_{16} & -m_{16} & -m_{16} & m_{16} & m_{16} & -m_{16} & 0 & 0 & 0 & m_{13} & 0 & -m_{13} & 0 & m_{17} & 0 & -m_{17} & -m_{13} & m_{13} & m_{17} & -m_{17} \\
m_{15} & m_{16} & m_{16} & m_{15} & m_{15} & m_{13} & m_{13} & m_{15} & -m_{13} & 0 & -m_{13} & -m_{11} & -m_{17} & 0 & -m_{17} & -m_{6} & -m_{4} & 0 & 0 & -m_{4} \\
m_{15} & m_{13} & m_{16} & m_{15} & m_{15} & m_{13} & m_{16} & m_{15} & -m_{17} & 0 & -m_{13} & -m_{4} & -m_{17} & 0 & -m_{13} & -m_{4} & -m_{6} & 0 & 0 & -m_{11} \\
-m_{13} & -m_{15} & -m_{15} & -m_{16} & -m_{13} & -m_{15} & -m_{15} & -m_{16} & m_{17} & m_{4} & m_{13} & 0 & m_{17} & m_{4} & m_{13} & 0 & 0 & m_{6} & m_{11} & 0 \\
-m_{16} & -m_{15} & -m_{15} & -m_{13} & -m_{16} & -m_{15} & -m_{15} & -m_{13} & m_{13} & m_{4} & m_{17} & 0 & m_{13} & m_{4} & m_{17} & 0 & 0 & m_{11} & m_{6} & 0 \\
m_{15} & m_{16} & m_{13} & m_{15} & m_{15} & m_{16} & m_{13} & m_{15} & -m_{13} & 0 & -m_{17} & -m_{4} & -m_{13} & 0 & -m_{17} & -m_{4} & -m_{11} & 0 & 0 & -m_{6}
\end{bmatrix} \tag{30}$$

$$M_{IJ2}^{JD} =$$

$$\begin{bmatrix}
-m_{12} & -m_{13} & 0 & 0 & -m_{13} & -m_{14} & 0 & 0 & m_{15} & m_{16} & -m_{13} & 0 & m_{15} & m_{16} & -m_{16} & m_{16} & m_{15} & m_{15} & -m_{16} & -m_{13} \\
-m_{13} & -m_{12} & 0 & 0 & -m_{14} & -m_{13} & 0 & 0 & m_{15} & 0 & -m_{13} & m_{16} & m_{15} & m_{16} & -m_{16} & m_{16} & m_{15} & m_{15} & -m_{13} & -m_{16} \\
0 & 0 & m_{12} & m_{13} & 0 & 0 & m_{13} & m_{14} & m_{13} & 0 & -m_{15} & -m_{16} & m_{16} & -m_{16} & -m_{15} & -m_{16} & m_{16} & m_{13} & -m_{15} & -m_{15} \\
0 & 0 & m_{13} & m_{12} & 0 & 0 & m_{14} & m_{13} & m_{13} & -m_{16} & -m_{15} & 0 & m_{16} & -m_{16} & -m_{15} & -m_{16} & m_{13} & m_{16} & -m_{15} & -m_{15} \\
-m_{13} & -m_{14} & 0 & 0 & -m_{12} & -m_{13} & 0 & 0 & m_{15} & m_{16} & -m_{16} & m_{16} & m_{15} & m_{16} & -m_{13} & 0 & m_{15} & m_{15} & -m_{16} & -m_{13} \\
-m_{14} & -m_{13} & 0 & 0 & -m_{13} & -m_{12} & 0 & 0 & m_{15} & m_{16} & -m_{16} & m_{16} & m_{15} & 0 & -m_{13} & m_{16} & m_{15} & m_{15} & -m_{13} & -m_{16} \\
0 & 0 & m_{13} & m_{14} & 0 & 0 & m_{12} & m_{13} & m_{16} & -m_{16} & -m_{15} & -m_{16} & m_{13} & 0 & -m_{15} & -m_{16} & m_{16} & m_{13} & -m_{15} & -m_{15} \\
0 & 0 & m_{14} & m_{13} & 0 & 0 & m_{13} & m_{12} & m_{16} & -m_{16} & -m_{15} & -m_{16} & m_{13} & -m_{16} & -m_{15} & 0 & m_{13} & m_{16} & -m_{15} & -m_{15} \\
m_{15} & m_{15} & m_{13} & m_{13} & m_{15} & m_{15} & m_{16} & m_{16} & -m_{6} & -m_{17} & 0 & -m_{17} & -m_{11} & -m_{13} & 0 & -m_{13} & -m_{4} & -m_{4} & 0 & 0 \\
m_{16} & 0 & 0 & -m_{16} & m_{16} & m_{16} & -m_{16} & -m_{16} & -m_{17} & 0 & m_{17} & 0 & -m_{13} & 0 & m_{13} & 0 & -m_{13} & -m_{17} & m_{17} & m_{13} \\
-m_{13} & -m_{13} & -m_{15} & -m_{15} & -m_{16} & -m_{16} & -m_{15} & -m_{15} & 0 & m_{17} & m_{6} & m_{17} & 0 & m_{13} & m_{11} & m_{13} & 0 & 0 & m_{4} & m_{4} \\
0 & m_{16} & -m_{16} & 0 & m_{16} & m_{16} & -m_{16} & -m_{16} & -m_{17} & 0 & m_{17} & 0 & -m_{13} & 0 & m_{13} & 0 & -m_{17} & -m_{13} & m_{13} & m_{17} \\
m_{15} & m_{15} & m_{16} & m_{16} & m_{15} & m_{15} & m_{13} & m_{13} & -m_{11} & -m_{13} & 0 & -m_{13} & -m_{6} & -m_{17} & 0 & -m_{17} & -m_{4} & -m_{4} & 0 & 0 \\
m_{16} & m_{16} & -m_{16} & -m_{16} & m_{16} & 0 & 0 & -m_{16} & -m_{13} & 0 & m_{13} & 0 & -m_{17} & 0 & m_{17} & 0 & -m_{13} & -m_{17} & m_{17} & m_{13} \\
-m_{16} & -m_{16} & -m_{15} & -m_{15} & -m_{13} & -m_{13} & -m_{15} & -m_{15} & 0 & m_{13} & m_{11} & m_{13} & 0 & m_{17} & m_{6} & m_{17} & 0 & 0 & m_{4} & m_{4} \\
m_{16} & m_{16} & -m_{16} & -m_{16} & 0 & m_{16} & -m_{16} & 0 & -m_{13} & 0 & m_{13} & 0 & -m_{17} & 0 & m_{17} & 0 & -m_{17} & -m_{13} & m_{13} & m_{17} \\
m_{15} & m_{15} & m_{16} & m_{13} & m_{15} & m_{15} & m_{16} & m_{13} & -m_{4} & -m_{13} & 0 & -m_{17} & -m_{4} & -m_{13} & 0 & -m_{17} & -m_{6} & -m_{11} & 0 & 0 \\
m_{15} & m_{15} & m_{13} & m_{16} & m_{15} & m_{15} & m_{13} & m_{16} & -m_{4} & -m_{17} & 0 & -m_{13} & -m_{4} & -m_{17} & 0 & -m_{13} & -m_{11} & -m_{6} & 0 & 0 \\
-m_{16} & -m_{13} & -m_{15} & -m_{15} & -m_{16} & -m_{13} & -m_{15} & -m_{15} & 0 & m_{17} & m_{4} & m_{13} & 0 & m_{17} & m_{4} & m_{13} & 0 & 0 & m_{6} & m_{11} \\
-m_{13} & -m_{16} & -m_{15} & -m_{15} & -m_{13} & -m_{16} & -m_{15} & -m_{15} & 0 & m_{13} & m_{4} & m_{17} & 0 & m_{13} & m_{4} & m_{17} & 0 & 0 & m_{11} & m_{6}
\end{bmatrix} \tag{31}$$

$$M_{IJ3}^{JD} =$$

$$\begin{bmatrix}
-m_{12} & -m_{13} & -m_{14} & -m_{13} & 0 & 0 & 0 & 0 & m_{15} & m_{15} & m_{15} & m_{15} & -m_{13} & -m_{16} & -m_{16} & -m_{13} & 0 & m_{16} & m_{16} & m_{16} \\
-m_{13} & -m_{12} & -m_{13} & -m_{14} & 0 & 0 & 0 & 0 & m_{15} & m_{15} & m_{15} & m_{15} & -m_{13} & -m_{13} & -m_{16} & -m_{16} & m_{16} & 0 & m_{16} & m_{16} \\
-m_{14} & -m_{13} & -m_{12} & -m_{13} & 0 & 0 & 0 & 0 & m_{15} & m_{15} & m_{15} & m_{15} & -m_{16} & -m_{13} & -m_{13} & -m_{16} & m_{16} & m_{16} & 0 & m_{16} \\
-m_{13} & -m_{14} & -m_{13} & -m_{12} & 0 & 0 & 0 & 0 & m_{15} & m_{15} & m_{15} & m_{15} & -m_{16} & -m_{16} & -m_{13} & -m_{13} & m_{16} & m_{16} & m_{16} & 0 \\
0 & 0 & 0 & 0 & m_{12} & m_{13} & m_{14} & m_{13} & m_{13} & m_{16} & m_{16} & m_{13} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & 0 & -m_{16} & -m_{16} & -m_{16} \\
0 & 0 & 0 & 0 & m_{13} & m_{12} & m_{13} & m_{14} & m_{13} & m_{13} & m_{16} & m_{16} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & -m_{16} & 0 & -m_{16} & -m_{16} \\
0 & 0 & 0 & 0 & m_{14} & m_{13} & m_{12} & m_{13} & m_{16} & m_{13} & m_{13} & m_{16} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & -m_{16} & -m_{16} & 0 & -m_{16} \\
0 & 0 & 0 & 0 & m_{13} & m_{14} & m_{13} & m_{12} & m_{16} & m_{16} & m_{13} & m_{13} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & -m_{16} & -m_{16} & -m_{16} & 0 \\
m_{15} & m_{15} & m_{15} & m_{15} & m_{13} & m_{13} & m_{16} & m_{16} & -m_{6} & -m_{4} & -m_{11} & -m_{4} & 0 & 0 & 0 & 0 & -m_{17} & -m_{17} & -m_{13} & -m_{13} \\
m_{15} & m_{15} & m_{15} & m_{15} & m_{16} & m_{13} & m_{13} & m_{16} & -m_{4} & -m_{6} & -m_{4} & -m_{11} & 0 & 0 & 0 & 0 & -m_{13} & -m_{17} & -m_{17} & -m_{13} \\
m_{15} & m_{15} & m_{15} & m_{15} & m_{16} & m_{16} & m_{13} & m_{13} & -m_{11} & -m_{4} & -m_{6} & -m_{4} & 0 & 0 & 0 & 0 & -m_{13} & -m_{13} & -m_{17} & -m_{17} \\
m_{15} & m_{15} & m_{15} & m_{15} & m_{13} & m_{16} & m_{16} & m_{13} & -m_{4} & -m_{11} & -m_{4} & -m_{6} & 0 & 0 & 0 & 0 & -m_{17} & -m_{13} & -m_{13} & -m_{17} \\
-m_{13} & -m_{13} & -m_{16} & -m_{16} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & 0 & 0 & 0 & 0 & m_{6} & m_{4} & m_{11} & m_{4} & m_{17} & m_{17} & m_{13} & m_{13} \\
-m_{16} & -m_{13} & -m_{13} & -m_{16} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & 0 & 0 & 0 & 0 & m_{4} & m_{6} & m_{4} & m_{11} & m_{13} & m_{17} & m_{17} & m_{13} \\
-m_{16} & -m_{16} & -m_{13} & -m_{13} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & 0 & 0 & 0 & 0 & m_{11} & m_{4} & m_{6} & m_{4} & m_{13} & m_{13} & m_{17} & m_{17} \\
-m_{13} & -m_{16} & -m_{16} & -m_{13} & -m_{15} & -m_{15} & -m_{15} & -m_{15} & 0 & 0 & 0 & 0 & m_{4} & m_{11} & m_{4} & m_{6} & m_{17} & m_{13} & m_{13} & m_{17} \\
0 & m_{16} & m_{16} & m_{16} & 0 & -m_{16} & -m_{16} & -m_{16} & -m_{17} & -m_{13} & -m_{13} & -m_{17} & m_{17} & m_{13} & m_{13} & m_{17} & 0 & 0 & 0 & 0 \\
m_{16} & 0 & m_{16} & m_{16} & -m_{16} & 0 & -m_{16} & -m_{16} & -m_{17} & -m_{17} & -m_{13} & -m_{13} & m_{17} & m_{17} & m_{13} & m_{13} & 0 & 0 & 0 & 0 \\
m_{16} & m_{16} & 0 & m_{16} & -m_{16} & -m_{16} & 0 & -m_{16} & -m_{13} & -m_{17} & -m_{17} & -m_{13} & m_{13} & m_{17} & m_{17} & m_{13} & 0 & 0 & 0 & 0 \\
m_{16} & m_{16} & m_{16} & 0 & -m_{16} & -m_{16} & -m_{16} & 0 & -m_{13} & -m_{13} & -m_{17} & -m_{17} & m_{13} & m_{13} & m_{17} & m_{17} & 0 & 0 & 0 & 0
\end{bmatrix} \tag{32}$$