International Journal of Non-Linear Mechanics 149 (2023) 104308

Contents lists available at ScienceDirect

International Journal of Non-Linear Mechanics

journal homepage: www.elsevier.com/locate/nlm

Review

An efficiency comparison of different ANCF implementations

Michael Taylor, Radu Serban, Dan Negrut *

University of Wisconsin-Madison, Department of Mechanical Engineering, Madison, WI, 53706, USA

ARTICLE INFO

Keywords:
Finite element method
Flexible multibody dynamics
Absolute nodal coordinate formulation
Linear viscoelastic

ABSTRACT

For multibody dynamics simulation using the Absolute Nodal Coordinate Formulation, multiple strategies are reported in the linear elastic material literature for calculating the generalized internal force and its Jacobian matrix. When examining the presentation of these strategies, which are all sound, it is difficult to assess which method is more efficient. We seek to clarify this issue by reporting the results of a comprehensive study that included five different ANCF solution strategies discussed in the literature. To increase the relevance of the study, we first extended these methods to incorporate a linear viscoelastic material model to account for damping effects within the elements. A beam, a shell, and a hexahedral element are each examined to provide a broader comparison. Both simple hand calculations and actual timing comparisons on a multi-core CPU architecture are investigated. For the simple beam element, only small differences manifest among the methods studied. However, for the shell and hexahedral elements, we noticed pronounced performance and storage cost differences among the methods.

Contents

1. Introduction

In virtual prototyping, the speed of simulation is crucial. It is not uncommon for a model to be run tens of thousands of times to identify a solution that is optimal under design constraints. For nonlinear finite element analysis as enabled by the Absolute Nodal Coordinate Formulation (ANCF), different numerical solution methodologies can

* Corresponding author.
E-mail address: negrut@wisc.edu (D. Negrut).
URL: https://https://sbel.wisc.edu/ (D. Negrut).

https://doi.org/10.1016/j.ijnonlinmec.2022.104308
Received 11 April 2022; Received in revised form 5 November 2022; Accepted 15 November 2022
Available online 25 November 2022
0020-7462/

implement *the same* ANCF method at different memory and execution speed costs. The situation is further complicated by the fact that the underlying hardware used to run the simulation does come into play. Ultimately, when examining various calculation methods for the generalized internal force and its Jacobian matrix within ANCF [1,2], it is unclear which method should be chosen [3–6]. This paper dwells on this observation, as it aims to provide a comparison of several ANCF implementation methods from multiple perspectives, including execution speed and memory footprint. For completeness, we utilize a beam, a shell, and a hexahedral element in this study.

Each of the compared methods adopts a different strategy for calculating the same generalized internal force and generalized internal force Jacobian matrix. While theoretical comparisons and back-of-the-envelope calculations to estimate performance can be made, as shown in [6] and later in this paper, the architecture of the target computer hardware cannot be ignored when examining execution performance. We factor in this aspect by choosing for our study the most common computer hardware architecture, i.e., multicore CPUs. We focus on Computer Aided Engineering-grade CPUs, which typically support vectorized AVX2 and FMA3 instruction sets that perform multiple floating-point operations in parallel.

While both structural mechanics and continuum mechanics approaches exist within ANCF [7], the methods herein utilize the continuum mechanics-based approach. Although [8] provided a calculation method for a linear viscoelastic material model with two coefficients for the damping contribution of the model, all implementations used in this comparison were modified to utilize the straightforward viscoelastic material presented in [9]. This material model only requires a single coefficient to approximate the damping mechanism within the material and it integrates easily with each of the methods.

The paper is organized as follows. Section 2 provides details on each of the ANCF implementation methods included in this comparison. Section 3 discusses the ANCF elements considered and clarifies how the number of Gauss quadrature points was selected for each element type. Section 4 reports on the hand-calculated memory footprint and computational cost for each solution approach. While it might seem that these theoretical comparisons should be sufficient to gauge implementation performance, it will be shown in Section 5 that this is not the case. Therein, we provide detailed timing information for the generalized internal force and Jacobian evaluation including roofline analyses [10]. Section 6 summaries the findings across this study.

2. Methods compared

The oldest method included in this comparison was proposed by García-Vallejo et al. [3]. They described an implementation method for both the generalized internal force and its Jacobian in which the computational effort was reduced by separating the nodal coordinates from the required integral across the volume of the element. Their approach results in numerous constant matrices that only need to be computed once, prior to the start of the simulation. The matrices can then be reused as needed during the generalized internal force and Jacobian evaluations that occur throughout the simulation. While the authors presented the methodology using large sparse precomputed matrices, they did note that symmetry and sparsity patterns could be leveraged to improve the performance of the method.

The next oldest method in this comparison was proposed by Gerstmayr and Shabana [4]. Instead of attempting to separate out the nodal coordinates from the required integral across the volume of the element, the authors focused on making this integration as efficient as possible for each generalized internal force evaluation throughout the simulation. They presented efficiency gains associated with the use of the First Piola–Kirchoff Stress Tensor; elimination of unnecessary multiplications by zeros; precomputing terms that are constant throughout the simulation; and reducing of the number of numerical integration points employed. The authors did not provide an analytical expression for the Jacobian matrix, but noted that the Jacobian could be calculated using numeric differentiation or it could be approximated by a co-rotated linear stiffness matrix.

The third oldest method in this comparison was proposed by Liu et al. [5]. The authors returned to the strategy of separating the nodal coordinates from the required integral across the volume of the element. They started from the expression for the generalized internal force in terms of the First Piola–Kirchoff Stress Tensor as presented in [4]. They then leveraged tensor notation to separate out the nodal coordinates from the integral. During their presentation, they also described an analytical calculation method for the Jacobian, highlighting the common terms between the generalized internal force and Jacobian calculations. Instead of the numerous constant matrices required in [3], only three larger precomputed matrices were needed. Since their work focused on modeling multi-layer shell elements, they pointed out that unlike the method presented in [4], the computational cost within a simulation for their method was independent of the number of numerical integration points used across the volume of the element. This is an important consideration when modeling shells with many discrete layers.

The last two ANCF implementations included in the comparison are variations of the approach presented in [6]. Therein, we proposed a modification of the generalized internal force method presented in [4]. The computational flow was reordered in an attempt to improve the memory alignment associated with the underlying computational hardware. A new calculation method for the analytical expression of the Jacobian matrix was also presented that embraced the same philosophy adopted for the generalized internal force calculation. The only difference between the two implementations discussed in [6] was whether or not intermediate calculations from the generalized internal force evaluation were saved and then reused as part of the analytical Jacobian calculation.

To help simplify the presentation, each of the ANCF implementation strategies discussed herein is assigned a letter identification. Methods “C”, “D”, and “E” are the exact same methods as “C”, “D”, and “E” of [6]. The generalized internal force calculation for method “C” is based on [4] and is performed using a loop over one Gauss quadrature point at a time. Instead of using a numeric Jacobian as in [4], we use our analytical expression for the Jacobian developed in [6]. The analytical Jacobian for method “C” is calculated one Gauss quadrature point at a time as well, thus keeping with the structure of the generalized internal force calculation.

We developed “D” using the same number of required operations as method “C”. We only changed the order, that is, sequence, in which the computation was carried out at each time step to calculate the generalized internal force and its Jacobian matrix across all the Gauss quadrature points during each step of the calculations. This was done in an attempt to better align the computations with the constraints of the hardware as well as to leverage existing efficient matrix multiplication routines.

Method “E” is a variation of method “D”. During the generalized internal force calculation, certain intermediate quantities, e.g., the value of the deformation gradient at each Gauss quadrature point, are calculated and subsequently reused in the Jacobian evaluation. Implementations “C” and “D” do not save these intermediate results, yet “E” does. While this leads to a smaller number of floating-point operations for the Jacobian evaluation, it increases the code complexity, the memory footprint per element, the memory transfers, and the internal force evaluation execution time. A choice between “D” and “E” represents a trade-off between increasing the execution time of the Jacobian or the generalized internal force evaluation.

Method “G” is based on the strategies presented by García-Vallejo et al. in [3] and it is discussed in detail in Appendices A and B. In “G”, a large number of constant matrices are evaluated once prior to the start of the simulation and then reused during each generalized internal force and Jacobian evaluation. Like method “E”, intermediate results

from the generalized internal force evaluation are stored and then reused during the Jacobian evaluation. While developing the implementation of this method, additional efficiency gain opportunities that went beyond the methodology in [3] were identified and implemented. Specifically, the repetitive nature of the shape function matrices leveraged in [4] can also be accounted for here to condense the required calculations. While this slightly changes how the generalized internal force and Jacobian are calculated, the concept of multiple constant matrices that can be computed prior to the start of the simulation is still retained.

Method "L" is largely based on the strategies presented by Liu et al. [5] and it is discussed in detail in Appendices C and D. In "L", only three large constant matrices are computed prior to the start of the simulation. In fact, the two largest matrices are simply reordered forms of each other. While only one of these two reordered matrices needs to be saved, the execution cost for doing so was found to be quite high. As a result, all three large matrices were stored and utilized for this method. Like in methods "E" and "G", intermediate results from the generalized internal force evaluation are stored and reused during the Jacobian evaluation.

Note that for "C", "D", and "E", which belong to the class of "Continuous Integration" methods, the number of floating-point operations for the generalized internal force and Jacobian evaluations depends on the number of Gauss quadrature points used for the element. In "G" and "L", which belong to the class of "Pre-Integration" methods, there is no dependency on the number of integration points used during the generalized internal force and Jacobian evaluations. This difference is important to keep in mind for applications with shell elements with multiple discrete layers, which was of interest in [5].

Since many of the implementation details for "C", "G", and "L" were not spelled out when these methods were proposed in [3–5], we made a best attempt to implement each method as efficiently as possible. It is important to point out three aspects when engaging in an effort to assess the relative performance of "C", "D", "E", "G", and "L": the methods need to be implemented in the same code to draw on the same software infrastructure; the simulations need to run on the same hardware; and it is highly desirable to use a compiled language like C or C++, as opposed to an interpreted language such as Python or Matlab. This effort to gauge relative performance was motivated by the need to understand which implementation should be used in our Chrono open source solver [11,12] to make it as fast as possible. As such, to the best of our abilities, we tried to identify the fastest implementation out of the ones considered herein. Finally, we believe it is next to impossible to compare the performance of these methods by simply reading a paper that describes them.

3. ANCF elements used for the comparisons

For thoroughness, three different existing ANCF continuum mechanics-based elements with increasing complexity will be used within this study. Using the element identification scheme proposed in [13], the simplest element considered is the fully parameterized 2-node Beam 3243 element [4]. It is implemented herein with the Enhanced Continuum Mechanics method [7], and has two nodes with a full set of three position vector gradients at each node for a total of 24 degrees of freedom per element. For the "medium complexity" category, we used the fully parameterized 4-node Shell 3443 element [14]. It has four nodes with a full set of three position vector gradients at each node for a total of 48 degrees of freedom per element. Finally, as an illustration of a complex element, we used the fully parameterized 8-node Hexahedral 3843 element [15]. It has eight nodes with a full set of three position vector gradients at each node for a total of 96 degrees of freedom per element. While other more accurate ANCF elements exist [16,17], these particular elements were selected since they provide consistent steps of increasing complexity for the comparisons.

While the number of unique shape functions for each element is determined by the number of nodes and the degrees of freedom at each node, there is leeway in relation to the number of numeric integration points that should be used for the generalized internal force and its Jacobian matrix. Assuming a straight and undistorted reference configuration, one can precisely determine the number of Gauss quadrature points required to exactly integrate the partial derivative of the strain energy density across the volume of the element. However, as has already been shown [4,18,19], a smaller number of Gauss quadrature points can be used without significantly affecting the results of the calculations. To determine the number of Gauss quadrature points required for a fair comparison of the "Continuous Integration" methods with the "Pre-Integration" methods, we carried out for each element a series of static tests typically utilizing realistic material properties with moderately large strains; each test used a different but fixed number of Gauss quadrature points.

The first test was a simple axial pull test that involved a slender beam. The properties for the beam were taken from [20]. A 0.508 m (20 in) long by 0.0127 m (0.5 in) wide by 0.003175 m (0.125 in) thick aluminum 7075-T651 cantilever beam is fixed on one end and a 15 kN axial load is applied to the center of the free cross-section of the beam. To model the beam, we used a mesh of 20 identical elements along the beam axis. At the fixed end, all the position and position vector gradient coordinates were held constant. Young's modulus was 71.7 GPa; Poisson's ratio was 0.33; gravity was ignored. Plasticity was not included, but the peak stresses can be put into context by keeping in mind that the yield strength for this material is approximately 500 MPa. Both the axial displacement of the tip and the maximum von Mises stress in the beam were used as the outputs of interest.

The second test was a cantilever beam bending test. The same model, mesh, setup, and properties were used as in the first test. However, instead of an axial tip load, a 6.6723 N (1.5 lbf) vertical tip load was applied along the thickness direction at the center of the tip of the beam. This corresponds to the peak load for the 90 degree orientation from the physical experiments in [20]. The vertical displacement of the tip and the maximum von Mises stress in the beam were used as the outputs of interest.

The third test was a combined bending load case with the same cantilever beam. For this test, the width direction of the beam was rotated so that it pointed 30 degrees from vertical and a point load of −17.7929 N (4 lbf) was applied at the center of the tip of the beam in the vertical direction. This corresponds to the peak load for the +30 degree orientation from the physical experiments in [20]. Both the vertical displacement and lateral displacement of the tip in global coordinates and the maximum von Mises stress in the beam were used as the outputs of interest. The sign of the vertical displacement is opposite compared to the space-fixed coordinates reported in the physical experiment.

The fourth test was an axial twist load case with the same cantilever beam. For this test, a moment of 10 N m was applied at the center of the tip of the beam. Like the axial displacement test, this axial twist test was not conducted as part of the physical experiments in [20]. It is simply used to examine the twist characteristics of the elements with different numbers of Gauss quadrature points.

All four tests above utilized a uniform mesh where the reference configuration of each element was straight and undistorted. This allowed for the exact integration of the partial derivative of the strain energy density across the element volume, which defines the expression of the generalized internal force. To examine the case of non-straight reference configurations, a test was created for the beam element and a separate existing test was used for the shell and hexahedral elements.

For the Beam 3243 element, a uniform mesh of 30 aluminum 7075-T651 beam elements with a cross-section width of 0.01 m and thickness of 0.005 m was created such that the beam axis of the elements formed a complete circle of radius 0.1 m. Over the course of the circumference of the circle, the position vector gradients were linearly rotated about the beam axis to create one full twist of the cross-section. The thickness direction of the beam was vertical at both the beginning and end of the

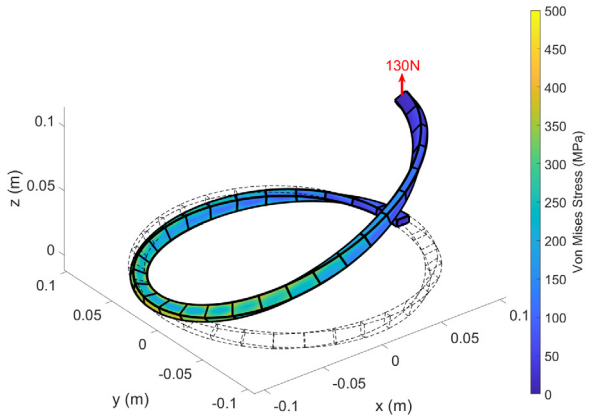

Fig. 1. Beam 3243 twisted split circle test showing the initial reference configuration of the mesh in dotted lines and the final loaded position of the mesh along with the calculated von Mises stresses for the case with $3 \times 2 \times 2$ $\mathbf{D}^0$ and $3 \times 1 \times 1$ $\mathbf{D}^\nu$ Gauss quadrature points.

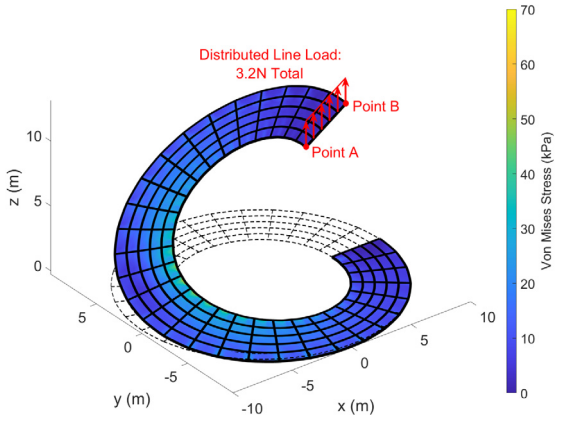

Fig. 2. Hexahedral 3843 thin split annular plate test showing the initial reference configuration of the mesh in dotted lines and the final loaded position of the mesh along with the calculated von Mises stresses for the case with $4 \times 4 \times 4$ Gauss quadrature points.

circle. The position vector and position vector gradients at the first node were fixed and a 130 N point load was applied in the vertical direction at the center of the free cross-section. The global displacement of the center of the free cross-section in all three directions and the maximum von Mises stress across all the beam elements were used as the outputs of interest. For reference, the initial configuration is plotted with dotted lines, and the final loaded position for the mesh is plotted with solid lines along with the final von Mises stresses in Fig. 1.

For the Shell 3443 and Hexahedral 3843 elements, we adopted the thin slit annular plate test from [21]. We used a mesh of 30 elements in the circumferential direction, 5 elements in the radial direction, and 1 element through the thickness. The annular plate has an outer radius of 10 m, an inner radius of 6 m, a thickness of 0.03 m, a Young's modulus of 21 MPa, and a Poisson's ratio of 0. The gravitational pull is ignored. All the position vector and position vector gradients along the fixed slit edge of the thin annular plate are held constant. A distributed line load of 0.8 N/m, 3.2 N total, is applied to the free end of the slit. The output of interest was the vertical displacement of the points on the inner radius, Point A, and outer radius, Point B, in the mid-surface at the free end of the slit, as well as the maximum von Mises stress across all the elements. Fig. 2 shows the initial reference configuration in dotted lines, the final loaded position for the mesh, and the von Mises stresses.

For all these tests, the objective was to gauge how the number of Gauss quadrature points used in the solution influenced its quality relative to using the full number of Gauss quadrature points. While it may be desirable to choose fewer Gauss quadrature points than selected for this study to soften the elements and help alleviate some of the known locking concerns [5,7], this path was not taken here so that a conservative timing comparison between methods "C", "D", and "E" could be performed relative to methods "G" and "L". If fewer Gauss quadrature points are used, then the execution performance of methods "C", "D", and "E" should all improve while the execution performance of methods "G" and "L" should stay constant.

Examining the test results for the Beam 3243 element shown in Table 1, where $5 \times 3 \times 3$ Gauss quadrature points are required to exactly integrate across the volume of the element for the generalized internal force evaluation, it can be seen that nearly identical results for each test can be achieved with as few as $3 \times 2 \times 2$ Gauss quadrature points for the $\mathbf{D}^0$ terms and $3 \times 1 \times 1$ Gauss quadrature points for the $\mathbf{D}^\nu$ terms within the Enhanced Continuum Mechanics method [7] used for comparisons with this element. However, when using only $2 \times 2 \times 2$ Gauss quadrature points for the $\mathbf{D}^0$ terms and $2 \times 1 \times 1$ Gauss quadrature points for the $\mathbf{D}^\nu$ terms, the maximum von Mises stress increases by 23% for the axial pull test. For the twisted split circle test, the maximum von Mises stress increases by 9% and there are notable changes in the tip Y displacement and to a lesser extent the tip Z displacement.

Examining the test results for the Shell 3443 element shown in Table 2, where $7 \times 7 \times 3$ Gauss quadrature points are required to exactly integrate across the volume of the element during the generalized internal force evaluation, it can be seen that nearly identical results for each test can be achieved with as few as $4 \times 4 \times 2$ Gauss quadrature points. When using only $3 \times 3 \times 2$ Gauss quadrature points, the results for all the tests except for the thin slit annular plate test are very close. Since the locking behavior of this element has been noted in existing studies in the literature [16], an argument could easily be made for using $3 \times 3 \times 2$ Gauss quadrature points to slightly soften the element to help address this locking behavior. However, to keep the generalized internal force results very close to those calculated with $7 \times 7 \times 3$ Gauss quadrature points, $4 \times 4 \times 2$ Gauss quadrature points are used for all the comparisons in this paper.

Finally, examining the test results for the Hexahedral 3843 element shown in Table 3, where $7 \times 7 \times 7$ Gauss quadrature points are required to exactly integrate across the volume of the element during the generalized internal force evaluation, it can be seen that nearly identical results for each test can be achieved with as few as $4 \times 4 \times 4$ Gauss quadrature points. When using only $3 \times 3 \times 3$ Gauss quadrature points, notable differences crop up in the results from the axial twist test and the thin slit annular plate test. Similar to Shell 3443, an argument could be made for softening this element and using $3 \times 3 \times 3$ Gauss quadrature points instead of the $4 \times 4 \times 4$ Gauss quadrature points that were selected for the comparisons. However, this was not done since it can result in some over softening of this element.

4. Theoretical comparisons

4.1. Data storage aspects

The memory footprint required by each implementation is dictated by the need to store: the constant mass matrix; precomputed values used for the generalized internal force evaluation and its Jacobian; and values saved during the internal force evaluation that are later reused for the Jacobian computation. The memory storage requirements of the implementations are compared below in the context of the three ANCF comparison elements.

For the "Continuous Integration" methods "C" and "D", $(3N + 1)\, N_Q$ values specific to these implementations need to be stored in memory. Here N is the number of unique shape functions for the given element, and N_Q is the number of integration points used during the generalized internal force and Jacobian evaluations. For method "E", which saves

Table 1
Gauss quadrature point test summary for the Beam 3243 element.

$\mathbf{D}^0$ Gauss quadrature points (ξ, η, ζ):	$5 \times 3 \times 3$	$5 \times 2 \times 2$	$4 \times 2 \times 2$	$\mathbf{3 \times 2 \times 2}$	$2 \times 2 \times 2$
$\mathbf{D}^\nu$ Gauss quadrature points (ξ, η, ζ):	$5 \times 1 \times 1$	$5 \times 1 \times 1$	$4 \times 1 \times 1$	$\mathbf{3 \times 1 \times 1}$	$2 \times 1 \times 1$
Axial pull test:					
Tip axial displacement (m)	2.5926E−03	2.5926E−03	2.5926E−03	2.5926E−03	2.6030E−03
Max Von Mises Stress (MPa)	397.92	397.92	397.92	397.92	488.71
Simple bending test:					
Tip vertical displacement (m)	−1.1351E−01	−1.1351E−01	−1.1351E−01	−1.1351E−01	−1.1368E−01
Max Von Mises Stress (MPa)	117.39	117.39	117.39	117.39	117.64
Combined bending test:					
Tip lateral displacement (m)	1.1804E−01	1.1804E−01	1.1804E−01	1.1804E−01	1.1833E−01
Tip vertical displacement (m)	−8.8948E−02	−8.8948E−02	−8.8948E−02	−8.8948E−02	−8.9128E−02
Max Von Mises Stress (MPa)	222.66	222.66	222.66	222.66	223.27
Axial twist test:					
Tip Angle (deg)	18.91	18.91	18.91	18.91	18.91
Max Von Mises Stress (MPa)	198.83	198.83	198.83	198.83	198.89
Twisted split circle test:					
Tip X displacement (m)	−1.5303E−02	−1.5303E−02	−1.5312E−02	−1.5311E−02	−1.5762E−02
Tip Y displacement (m)	−1.9225E−02	−1.9224E−02	−1.9232E−02	−1.9243E−02	−1.6695E−02
Tip Z displacement (m)	1.0984E−01	1.0984E−01	1.0987E−01	1.0987E−01	1.1471E−01
Max Von Mises Stress (MPa)	492.19	492.19	492.43	492.23	537.68

Table 2
Gauss quadrature point test summary for the Shell 3443 element.

Gauss quadrature points (ξ, η, ζ):	$7 \times 7 \times 3$	$7 \times 7 \times 2$	$6 \times 6 \times 2$	$5 \times 5 \times 2$	$\mathbf{4 \times 4 \times 2}$	$3 \times 3 \times 2$
Axial pull test:						
Tip axial displacement (m)	2.6009E−03	2.6009E−03	2.6009E−03	2.6009E−03	2.6009E−03	2.6012E−03
Max Von Mises Stress (MPa)	435.66	435.66	435.66	435.66	435.66	437.08
Simple bending test:						
Tip vertical displacement (m)	−1.0093E−01	−1.0093E−01	−1.0093E−01	−1.0093E−01	−1.0093E−01	−1.0093E−01
Max Von Mises Stress (MPa)	135.93	135.93	135.93	135.93	135.93	135.91
Combined bending test:						
Tip lateral displacement (m)	1.0691E−01	1.0691E−01	1.0691E−01	1.0691E−01	1.0691E−01	1.0692E−01
Tip vertical displacement (m)	−8.5095E−02	−8.5095E−02	−8.5095E−02	−8.5095E−02	−8.5095E−02	−8.5139E−02
Max Von Mises Stress (MPa)	268.50	268.50	268.50	268.50	268.50	268.29
Axial twist test:						
Tip Angle (deg)	80.47	80.47	80.47	80.47	80.47	80.53
Max Von Mises Stress (MPa)	431.02	431.03	431.03	431.03	431.02	434.12
Thin slit annular plate test:						
Point A vertical displacement (m)	5.7467E+00	5.7467E+00	5.7469E+00	5.7472E+00	5.7920E+00	6.1776E+00
Point B vertical displacement (m)	6.8873E+00	6.8873E+00	6.8875E+00	6.8888E+00	6.9360E+00	7.4343E+00
Max Von Mises Stress (MPa)	0.06	0.06	0.06	0.06	0.06	0.05

Table 3
Gauss quadrature point test summary for the Hexahedral 3843 element.

Gauss quadrature points (ξ, η, ζ):	$7 \times 7 \times 7$	$6 \times 6 \times 6$	$5 \times 5 \times 5$	$\mathbf{4 \times 4 \times 4}$	$3 \times 3 \times 3$
Axial pull test:					
Tip axial displacement (m)	2.6091E−03	2.6091E−03	2.6091E−03	2.6091E−03	2.6099E−03
Max Von Mises Stress (MPa)	517.08	517.08	517.08	517.08	516.23
Simple bending test:					
Tip vertical displacement (m)	−1.1000E−01	−1.1000E−01	−1.1000E−01	−1.1000E−01	−1.1024E−01
Max Von Mises Stress (MPa)	176.78	176.78	176.78	176.78	174.96
Combined bending test:					
Tip lateral displacement (m)	1.1768E−01	1.1768E−01	1.1768E−01	1.1768E−01	1.1828E−01
Tip vertical displacement (m)	−9.4273E−02	−9.4273E−02	−9.4273E−02	−9.4273E−02	−9.5533E−02
Max Von Mises Stress (MPa)	322.10	322.10	322.10	322.09	319.02
Axial twist test:					
Tip Angle (deg)	90.76	90.76	90.76	90.76	99.91
Max Von Mises Stress (MPa)	569.52	569.52	569.52	569.56	715.50
Thin slit annular plate test:					
Point A vertical displacement (m)	10.6700	10.6612	10.7862	10.5396	10.7862
Point B vertical displacement (m)	11.7355	11.7253	11.8719	11.5803	11.8719
Max Von Mises Stress (MPa)	0.06	0.06	0.06	0.06	0.06

select data from the generalized internal force calculation for later use, an additional $24N_Q$ values need to be stored for a typical element. For the "Pre-Integration" method "G", $0.5N^4+0.5N^3+18N^2$ values need to be stored for the generalized internal force and Jacobian evaluations. For the "Pre-Integration" method "L", the number of values that needs to be stored scales like $2\left(N^4+N^2\right)$. This can be condensed to N^4+2N^2

Table 4
Number of bytes of required storage per element for the precomputed terms for the generalized internal force evaluation, the Jacobian evaluation, the mass matrix, and any intermediate calculations during the generalized internal force evaluation for each of the different implementation methods assuming double precision (8 bytes per value) and a single layer Shell 3443 element.

	Beam 3243	Shell 3443	Hexa 3843
N	8	16	32
N_Q	15	32	64
Mass matrix	288	1,088	4,224
Method			
C and D	3,000	12,544	49,664
E	5,880	18,688	61,952
G	27,648	315,392	4,472,832
L	66,560	1,052,672	16,793,600

Table 5
Amount of elements that can be stored in 16 GiB of RAM approximated by assuming that only the values in Table 4 need to be stored.

Method	Beam 3243	Shell 3443	Hexa 3843
C and D	5,225,021	1,260,260	318,806
E	2,785,322	868,723	259,608
G	614,972	54,284	3,837
L	256,999	16,303	1,022

by generating the $\left[N^2 \times N^2\right]$ matrix $\boldsymbol{\Theta}^{(2)}$ from $\boldsymbol{\Theta}^{(1)}$ as part of the Jacobian evaluation since $\boldsymbol{\Theta}^{(2)}$ is just a reordered form of $\boldsymbol{\Theta}^{(1)}$ (see Appendix C). However, it was found that there is a significant increase in the execution cost of the Jacobian evaluation for generating $\boldsymbol{\Theta}^{(2)}$ from $\boldsymbol{\Theta}^{(1)}$ on the fly.

Next, since the mass matrix stays the same across all implementations, its memory footprint only depends on the DOF of the element. Due to the symmetry and repetitive pattern of the mass matrix [6], only the diagonal and upper or lower triangular terms of the compact mass matrix shown in Eq. (1) need to be stored, that is, $N(N+1)/2$ values.

$$\mathbf{M}^{compact} = \int_{V_\xi} \left(\rho \bar{\mathbf{S}} \bar{\mathbf{S}}^{\mathsf{T}}\right) \det\left(\mathbf{J}_{0\xi}\right) \, dV_\xi. \tag{1}$$

Table 4 summarizes the storage cost for each of the three elements for each of the five methods. The "Continuous Integration" methods ("C", "D", and "E") have significantly smaller memory footprints compared to either of the "Pre-Integration" methods ("G" and "L"). For instance, for the Hexahedral 3843 element, the "Continuous Integration" methods require approximately two orders of magnitude less storage than the "Pre-Integration" methods. To put the memory requirements listed in Table 4 in perspective, the number of elements that can be stored in 16GiB of RAM (the amount of RAM on the laptop utilized in Section 5), is shown in Table 5. These element counts are in fact slightly overestimated given that only the entries in the mass matrix, values saved from the generalized internal force evaluation, and the pre-computed values are factored in.

4.2. FLOP estimates

The focus in this subsection is on providing an approximate count of the number of floating-point operations (FLOPs) required by the ANCF implementations of interest. Please note the distinction between FLOPs and FLOPS, the latter referring to the number of floating point operations *per second*. In this exercise, we will differentiate between multiplication and addition operations. Moreover, while more sophisticated matrix multiplication methods exist that can use fewer operations [22], the direct formula for matrix multiplication will be assumed here. Specifically, two matrices of size $[a \times b]$ and $[b \times c]$ require $a \times b \times c$ multiplications and $a \times (b-1) \times c$ additions. Using this formula, the approximate FLOP counts for the generalized internal force evaluation are shown in Table 6. The corresponding values for the Jacobian matrix evaluation are shown in Table 7.

Table 6
Hand calculated number of floating-point operations, in thousands (kFLOPs), for each of the implementation methods for the generalized internal force calculation (assuming a single layer for the Shell 3443 element).

	Beam 3243	Shell 3443	Hexa 3843
N	8	16	32
N_Q	15	32	64
Method			
C, D, and E	8.2	34	123
G	20.7	252	3539
L	8.9	134	2109

Table 7
Hand calculated number of floating-point operations, in thousands (kFLOPs), for each of the implementation methods for the Jacobian calculation (assuming a single layer for the Shell 3443 element).

	Beam 3243	Shell 3443	Hexa 3843
N	8	16	32
N_Q	15	32	64
Method			
C and D	112	1014	7,664
E	106	990	7,581
G	47	633	9,247
L	74	1181	18,878
Numeric (C, D, or E)	403	3294	23,781

While the data in Tables 6 and 7 is informative by itself, greater insights can be gained by examining the major contributors to these FLOP counts. We will probe deeper by considering the ANCF shell 3443 element with a single layer. Starting with methods "C", "D", and "E", roughly 50% of the FLOPs for the generalized internal force evaluation come from calculating the deformation gradient and the time derivative of the deformation gradient for each of the required Gauss quadrature points. With methods "D" and "E", this calculation is expressed as a single matrix multiplication of size $[6 \times N] \times \left[N \times 3N_Q\right] = [6 \times 16] \times [16 \times 96]$. With method "C", these same calculations are performed using many smaller matrix multiplications over the course of looping through all the Gauss quadrature points. Another roughly 25% of the FLOPs comes from multiplying the shape function derivative matrix, $\bar{\mathbf{S}}^D$ by the transpose of the First Piola Kirchhoff stress tensor, $\mathbf{P}$. With methods "D" and "E", this calculation is written as a single matrix multiplication of size $\left[N \times 3N_Q\right] \times \left[3N_Q \times 3\right] = [16 \times 96] \times [96 \times 3]$ versus many smaller matrix multiplications with method "C". The final roughly 25% of the FLOPs for the generalized internal force evaluation are due to the smaller calculations required to generate the First Piola Kirchhoff Stress Tensor from the deformation gradient and the time derivative of the deformation gradient at each of the required Gauss quadrature points. Based on this analysis, approximately 3/8 of the total FLOPs are required to include the simple linear viscoelastic material law when going beyond a purely linear elastic law.

Examining the Jacobian matrix evaluation for method "E", which stores and reuses intermediate calculations from the generalized internal force evaluation, roughly 90% of the required FLOPs are due to the $\left[3N \times 6N_Q\right] \times \left[6N_Q \times 3N\right] = [48 \times 192] \times [192 \times 48]$ multiplication $\left(\frac{\partial \boldsymbol{\varepsilon}^{Vec}}{\partial \mathbf{e}}\right)^{\mathsf{T}} \mathbb{H}$ which is performed as 6 relatively square $\left[3N \times N_Q\right] \times \left[N_Q \times 3N\right] = [48 \times 32] \times [32 \times 48]$ multiplications in the actual implemented code. While it seems like not reusing roughly 75% of the FLOPs from the generalized internal force calculation for methods "C" and "D" is significant, this only increases their Jacobian matrix FLOP counts by about 2% versus method "E". Like the generalized internal force evaluation, the inclusion of the simple linear viscoelastic material law adds significant overhead to methods "C", "D", and "E" over a purely linear elastic material law since it breaks the symmetry of the product from $\left(\frac{\partial \boldsymbol{\varepsilon}^{Vec}}{\partial \mathbf{e}}\right)^{\mathsf{T}} \mathbb{H}$. As a result, all the terms in this product need to be calculated instead of just the diagonal and upper or lower triangular terms required with a purely linear elastic material law.

For method "G", 99% of the FLOPs for the force evaluation come from the $N(N+1)/2 = 136$ smaller matrix multiplications of size $[3\times N][N\times N]\ [N\times 3] = [3\times 16][16\times 16][16\times 3]$ required to generate the matrix $\mathbf{K}_2$, see Appendix A. For the Jacobian evaluation, the majority of the FLOPs come from the steps required to calculate the matrix $\mathbf{K}_3$ defined in Appendix A. Out of these steps, approximately 22% of the total FLOPs come from calculating the $N^2 = 256$ matrix-vector products of size $[N\times N][N\times 1] = [16\times 16][16\times 1]$ required to produce the vectors $\mathbf{G}$ defined in Appendix B. Another 67% of the total FLOPs come from the $N^2 = 256$ matrix multiplications of size $[3\times N][N\times N] = [3\times 16][16\times 16]$ required to calculate the terms $\mathbf{L}$ defined in Appendix B. For method "G", the change from a simple linear elastic material to the simple linear viscoelastic material used in this paper adds less than 200 FLOPs to the generalized internal force evaluation. However, for the Jacobian matrix evaluation, the inclusion of the simple linear viscoelastic material significantly increases the computational cost since $\mathbf{K}_3$ is no longer symmetric. As a result, each entry in $\mathbf{K}_3$ must now be calculated.

Examining the FLOP counts for method "L", 98% of the FLOPs for the generalized internal force evaluation are due to the $\left[N^2\times N^2\right]\times\left[N^2\times 1\right] = [256\times 256]\times[256\times 1]$ matrix–vector product $\boldsymbol{\Theta}^{(1)}\boldsymbol{\Pi}^{(1)}$. For the Jacobian matrix evaluation, almost 100% of the FLOPs are due to the $\left[9\times N^2\right]\times\left[N^2\times N^2\right] = [9\times 256]\times[256\times 256]$ matrix multiplication of $\boldsymbol{\Pi}^{(2)}\boldsymbol{\Theta}^{(2)}$. It is important to note that a significant reordering of the terns from the product of this matrix multiplication is required afterward which does not appear as required FLOPs. For this method, the inclusion of the simple linear viscoelastic material adds less than approximately 200 FLOPs to both force and Jacobian evaluations.

4.3. Computations per stored byte

The arithmetic intensity, defined as the ratio of the number of floating-point operations per number of bytes of data accessed to compute the internal force and Jacobian, provides early insights into the expected performance of each ANCF implementation. This subsection reports on a succinct analysis that produces for each implementation an estimate of its arithmetic intensity. This value provides an early indication of how memory-bound each implementation is, at least on paper. In Section 5.2, it will also be used when the implementations are compared via a roofline analysis, which represents the ultimate test for assessing whether the execution is memory-bound or compute-bound.

Table 8 reports estimated arithmetic intensities for the generalized internal force calculations. Note that for all implementations, they stay relatively constant as the complexity of the element increases, with the exception of "E" which gradually approaches methods "C" and "D". The estimated arithmetic intensity for the "Continuous Integration" methods ("C", "D", and "E") are all higher than the two "Pre-Integration" methods. The ratio for method "L" is quite low due to the nature of the matrix–vector multiplication that contains most of the required operations. In general, it is desirable to have high arithmetic intensity, since one memory access can take from four compute cycles when hitting L1 cache, to about 20 clock cycles when hitting L2 cache, to 70 clock cycles when serviced by L3 cache, to 150 and above upon cache miss and a trip to main memory [23]. When pipelined, an arithmetic instruction, e.g., fused multiply-add, can be done on a modern architecture in one clock cycle. As such, it is advantageous to have many calculations (which can be done one per one clock cycle), as opposed to data movement, which might require up to hundreds of clock cycles.

The ratios for the Jacobian matrix evaluations shown in Table 9 are all significantly higher than the corresponding entries for the generalized internal force calculations. Unlike the generalized internal force ratios, these ratios notably increase with element complexity for the three "Continuous Integration" methods. This indicates that these three calculation methods may be less and less memory-bound as the complexity of the element increases, and start to be compute-bound. While these ratios do increase for the two "Pre-Integration" methods, they do so at a much slower rate than for the three "Continuous Integration" methods.

Table 8
The ratio of the number of calculated floating-point operations to the required amount stored data for the generalized internal force calculation along with the nodal coordinates, time derivative of the nodal coordinates, and the resulting generalized internal force vector for each of the methods.

Method	Beam 3243	Shell 3443	Hexa 3843
C and D	2.28	2.47	2.37
E	1.26	1.71	1.91
G	0.74	0.80	0.79
L	0.26	0.25	0.25

Table 9
The ratio of the number of calculated floating-point operations to the required amount stored data for the generalized internal force calculation along with the unique compact entries in the mass matrix, the nodal coordinates, time derivative of the nodal coordinates, and the resulting Jacobian matrix for each of the methods.

Method	Beam 3243	Shell 3443	Hexa 3843
C and D	13.51	30.87	59.34
E	9.54	25.41	53.60
G	1.42	1.88	2.03
L	1.92	2.16	2.23

5. Implementation comparisons

While back-of-the-envelope calculations and metrics can be insightful when comparing competing implementations, the ultimate comparison metric goes back to running the implementations on an actual compute hardware. Herein, we are using multicore CPUs supporting vectorized AVX2 and FMA3 instructions. The use of vectorized AVX2 instructions enables four double-precision additions or multiplications to be computed in one clock cycle; vectorized FMA3 instructions can compute four double-precision fused multiply-adds, like $x = \alpha y + x$, per clock cycle when pipelined. To tap these instructions, memory must be properly aligned and the compiler must deem these instructions safe to issue.

As discussed in Section 4.3, there are two sides of the performance coin: speed of floating point operations, and speed of memory/data access. For the latter, the required data may travel all the way from main memory and through three CPU cache levels before it lands in a register for processing. The lower the cache level that services a memory operation, the lower the latency of the request. It can be one hundred times faster to access from L1 cache than from main memory. The constraint faced though is that L1 cache is very small—of the order of 32 KiB. For L2 cache, one can count on roughly 256 KiB. Level 3 caches are typically of the order 10–20 MiB, with recent chips sporting as much as 256 MiB. This is important to keep in mind, since implementations "G" and "L" are memory hungry, and therefore bound to make less efficient use of L1 and L2 caches. However, this argument is not definitive, since modern architectures have very strong instruction level parallelism support that engages in data prefetching, which alleviates some of the memory pressure manifest in "G" and "L". In general, it is rare that an implementation can reach its maximum compute limit. Almost always, the applications are in fact memory-bound, which does not bode well for "G" and "L".

Due to the complexity and variety of modern processors, the performance of the implementations of each method will be examined in multiple stages. First, in Section 5.1, the execution performance outside the context of a simulation is presented for *one* generalized internal force evaluation and *one* Jacobian matrix evaluation. Section 5.2 provides a more in-depth analysis of the execution performance by taking into account the limitations of the underlying hardware. Finally, Section 5.3 presents timing results for actual dynamic simulations to gain additional insights into how other components of the simulation, e.g., solution of the linear system and matrix factorization, play into the overall performance equation. To compare the implementations against each other, each element and method was implemented in C++ as an extension of the Chrono v7.0.1 multi-physics engine [11]. The source

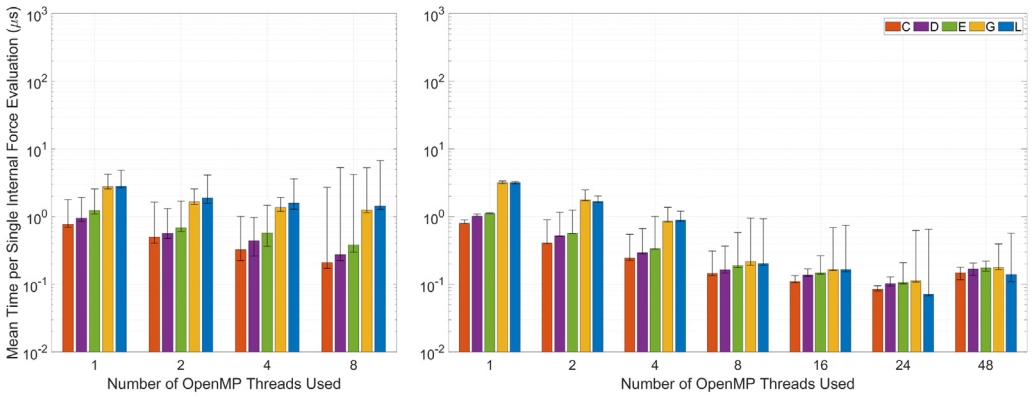

Fig. 3. Mean times for a single generalized internal force evaluation for a fully parameterized ANCF beam 3243 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors, with error bars showing the minimum and maximum measured times.

Table 10
Mean times for a single generalized internal force evaluation for a fully parameterized ANCF beam 3243 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors for different numbers of OpenMP threads for each of the five methods.

Threads	C	D	E	G	L
1	0.77	0.95	1.23	2.77	2.82
2	0.50	0.57	0.68	1.67	1.88
4	0.32	0.44	0.57	1.37	1.59
8	0.21	0.27	0.38	1.26	1.44
1	0.80	1.02	1.11	3.16	3.18
2	0.41	0.52	0.57	1.76	1.67
4	0.24	0.29	0.34	0.86	0.89
8	0.15	0.16	0.19	0.22	0.20
16	0.11	0.14	0.15	0.17	0.17
24	0.08	0.10	0.11	0.11	0.07
48	0.15	0.17	0.17	0.18	0.14

Table 11
Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF beam 3243 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors for different numbers of OpenMP threads for each of the five methods.

Threads	C	D	E	G	L
1	7.76	5.18	4.96	10.48	5.53
2	4.35	2.82	2.79	5.72	3.28
4	3.32	2.21	1.93	4.46	2.34
8	2.03	1.35	1.27	2.69	1.70
1	8.43	6.99	6.44	8.97	6.98
2	4.22	3.50	3.23	4.47	3.57
4	2.15	1.80	1.65	2.20	1.78
8	1.09	0.92	0.84	0.98	0.63
16	0.94	0.66	0.60	0.81	0.51
24	0.66	0.48	0.44	0.57	0.33
48	0.70	0.52	0.49	0.63	0.38

code for all of these implementations and associated examples can be found at [24].

Two different processors are used in the comparisons that follow. The first processor, referred to as "Intel i7-7700HQ", is an Intel® Core™ i7-7700HQ processor hosted on a laptop. Intel i7-7700HQ has four physical cores with Hyper-Threading, which turns the laptop into an eight virtual core machine. The code was compiled using LLVM Clang 12.0.1 running on Ubuntu within Windows 10 Subsystem for Linux 2 (WSL2) using the {-O3 -DNDEBUG} compiler flags. The second processor is a second-generation AMD EPYC™ 7272 CPU. Note that *two* such processors are hosted on one workstation; this configuration will be identified below as "AMD EPYC 7272". Each AMD EPYC™ 7272 processor has 12 physical cores with simultaneous multithreading, for a total of 24 physical cores and 48 virtual cores on the workstation used (recall there are two AMD processors involved). The source code for computer AMD EPYC 7272 was compiled using LLVM Clang 13 running on Fedora using the {-O3 -DNDEBUG} compiler flags.

5.1. Average generalized internal force and Jacobian execution times

First, we present the results of repeated force and Jacobian evaluations. This exercise eliminates from the performance equation the choice of numerical integrator, linear solver, etc. Thus, for each element–implementation pair, a simple mesh of elements was created and then the time to evaluate the generalized internal force and the Jacobian matrix for all the elements in the mesh was measured for the implementation considered. This time was then normalized by the number of elements in the mesh. This process was repeated multiple times with different random nodal locations to generate a range and mean of normalized times for each element-implementation pair.

The Beam 3243 element benchmark test used a mesh of 1024 equal elements arranged in a line and 1000 iterations of small random changes in the nodal coordinates to generate the timing statistics. Examining the results shown in Fig. 3 and Table 10, it can be seen that all three of the "Continuous Integration" methods ("C", "D", and "E") have lower execution times than either of the "Pre-Integration" methods ("G" and "L") for the Intel i7-7700HQ processor. Both "Pre-Integration" methods have similar execution times even though method "G" has over twice the calculated number of required operations. For the AMD EYPC 7272 workstation, this same trend continues through 16 OpenMP [25] threads. If additional threads are used, then method "L" appears to perform the best. To investigate this difference, a series of dynamic simulations were conducted with a uniform mesh of either 48 or 1024 beam elements based on the three-dimensional pendulum discussed in [26]. The average time to evaluate the generalized internal force, normalized per element, was generated over the course of a 1 s simulation with a fixed time step of 10^{-3} s. Comparing the results from the mesh of 48 beam elements shown in Fig. 4 against the mesh of 1024 beam elements shown in Fig. 5, the relative performance of method "L" is quite different. For the smaller mesh, method "L" generally has one of the shortest execution times. However, with the larger mesh, method "L" generally has one of the longest execution times. This difference in relative timing highlights the complexity of comparing and studying the expected performance of different ANCF implementations.

It is also interesting to note that for just the generalized internal force evaluation for the Beam 3243 element, method "C", which calculates the generalized internal force utilizing a loop over each Gauss quadrature point, is slightly faster than both method "D" and "E", which utilize larger matrix multiplications in an attempt to better align with the constraints of the hardware. The likely reason for this difference is discussed in [6].

Examining the Jacobian matrix evaluation times in Fig. 6 and Table 11, methods "D" and "E" perform the best on the Intel i7-7700HQ processor followed by methods "L", "C", and "G" respectively. Looking

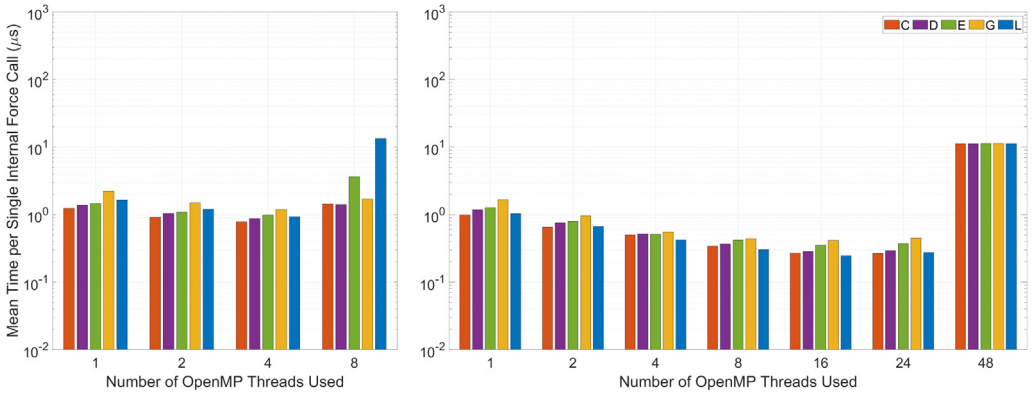

Fig. 4. Mean times for a single generalized internal force evaluation for a fully parameterized ANCF beam 3243 element onthe Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors measured during a 1 s dynamic simulation with a mesh of 48 elements.

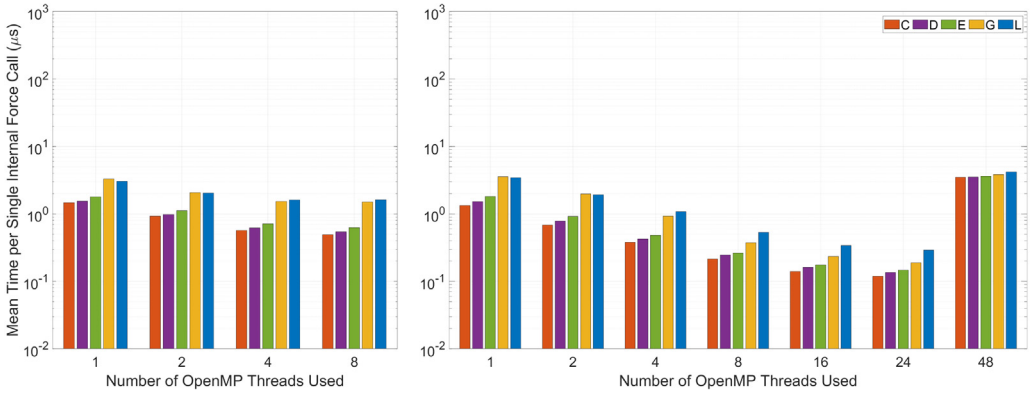

Fig. 5. Mean times for a single generalized internal force evaluation for a fully parameterized ANCF beam 3243 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors measured during a 1 s dynamic simulation with a mesh of 1024 elements.

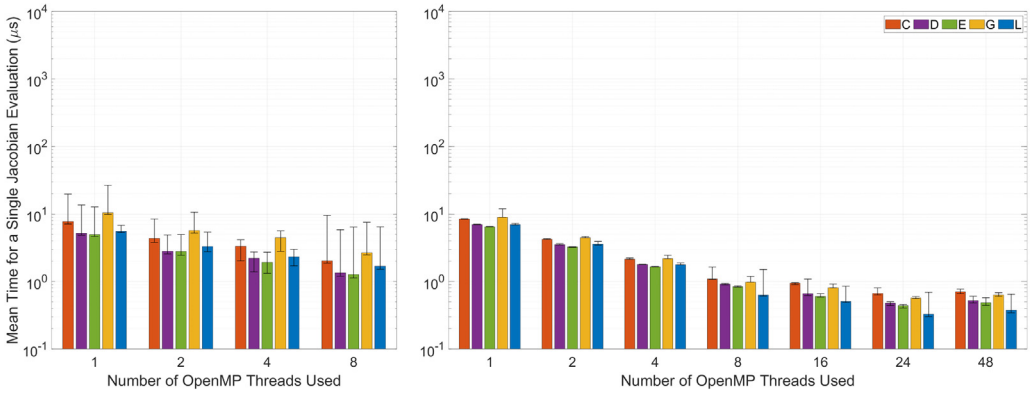

Fig. 6. Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF beam 3243 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors with error bars showing the minimum and maximum measured times.

at the AMD EYPC 7272 processor, the Jacobian matrix evaluation times are closer together, especially among methods “D”, “E”, and “L”. While method “L” does become slightly faster as more threads are used, as with the generalized internal force evaluations, this is likely a result of the size of the test setup. This conclusion is based on the relative differences in the mean Jacobian evaluation times for the dynamic simulation with a mesh of 48 elements in Fig. 7 compared to the mesh of 1024 elements in Fig. 8.

The Shell 3443 element benchmark tests used a mesh of 1024 equal elements connected edge to edge in a single line and 100 iterations of small random changes in the nodal coordinates to generate the timing statistics. Examining the generalized internal force evaluation times in Fig. 9 and Table 12, the “Continuous Integration” methods “C”, “D”, and “E” have notably lower evaluation times than either of the two “Pre-Integration” methods, with method “D” generally being the fastest. Looking at the case of 24 threads on the AMD EYPC 7272 processors, methods “D” and “E” have the same lowest generalized internal force evaluation time. Method “C” requires approximately 3 times as much time and methods “G” and “L” both require approximately 37 times as much time per evaluation. This timing difference is much larger than the calculated difference in required FLOPs shown in Table 6 where method “L” only requires approximately 3 times as many operations and method “G” requires approximately 7 times as many operations as methods “C”, “D”, and “E”.

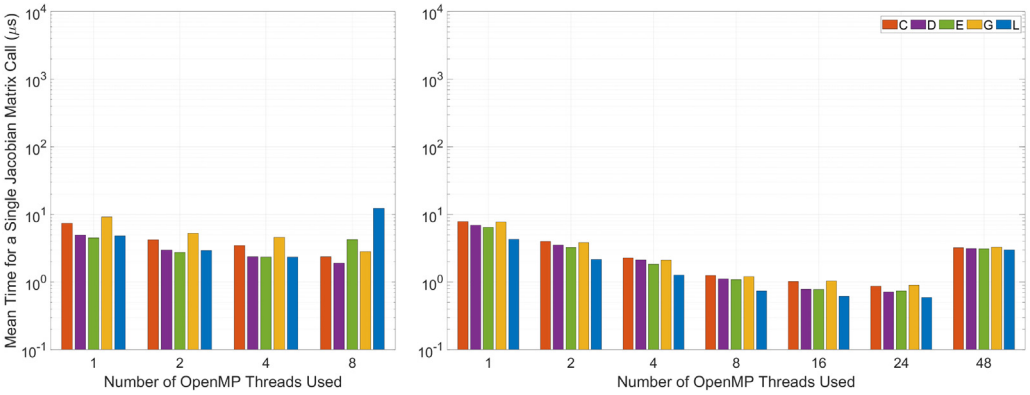

Fig. 7. Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF beam 3243 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors measured during a 1 s dynamic simulation with a mesh of 48 elements.

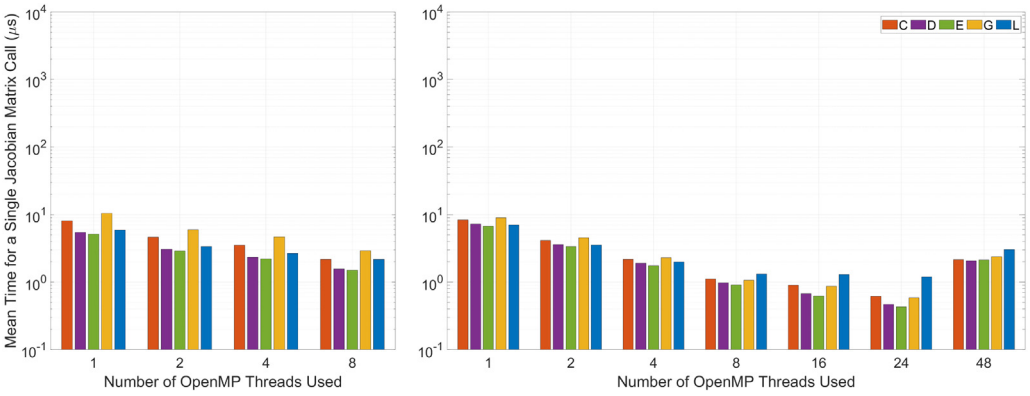

Fig. 8. Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF beam 3243 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors measured during a 1 s dynamic simulation with a mesh of 1024 elements.

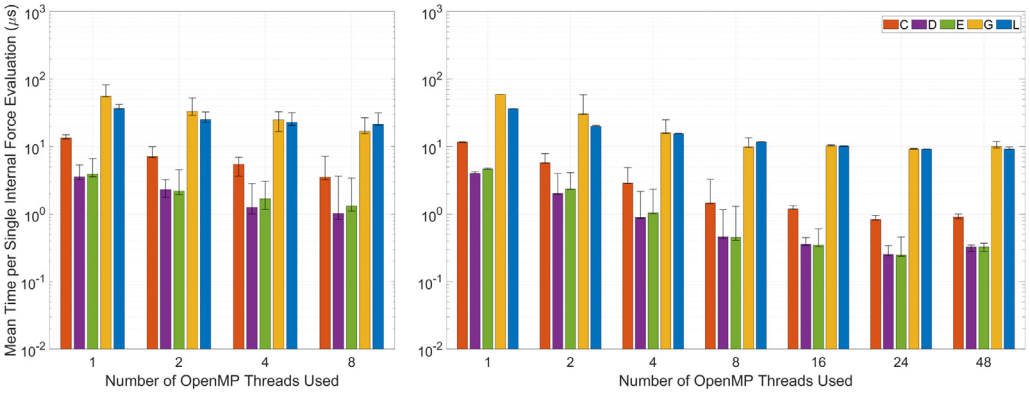

Fig. 9. Mean times for a single generalized internal force evaluation for a fully parameterized ANCF shell 3443 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors with error bars showing the minimum and maximum measured times.

Examining the Jacobian matrix evaluation times for the Shell 3443 element in Fig. 10 and Table 13, a similar ordering of the comparison methods can be seen except where method "E" is generally the fastest instead of method "D" as expected due to its lower FLOP count. The difference among the methods is much smaller than it is with the generalized internal force evaluations. Looking again at the case of 24 threads on the AMD EYPC 7272 processors, method "C" only requires approximately 1.5 times as much time as method "E" and methods "G" and "L" both require approximately 3 times as much time as method "E". As again, this ordering does not align with the calculated number of required FLOPs as shown in Table 7 where method "G" was predicted to be the fastest method by a factor of 1.5 to 2.

When examining these results, it is important to keep in mind that these tests assume a single-layer shell element and not the discrete multilayer shell element discussed by Liu et al. [5]. If a single equivalent layer could not be used, then the evaluation time for methods "C", "D", and "E" would need to be multiplied by the number of discrete layers, whereas the evaluation times for methods "G" and "L" would remain constant. In the worst-case ratio of a pure Newton–Raphson method where the number of generalized internal force and Jacobian matrix evaluations are identical, a five-layer shell would have similar total times for methods "D", "E", "G", and "L". If the ratio of generalized internal force evaluations was higher, the break-even point between the methods would occur with a much larger number of layers. For the

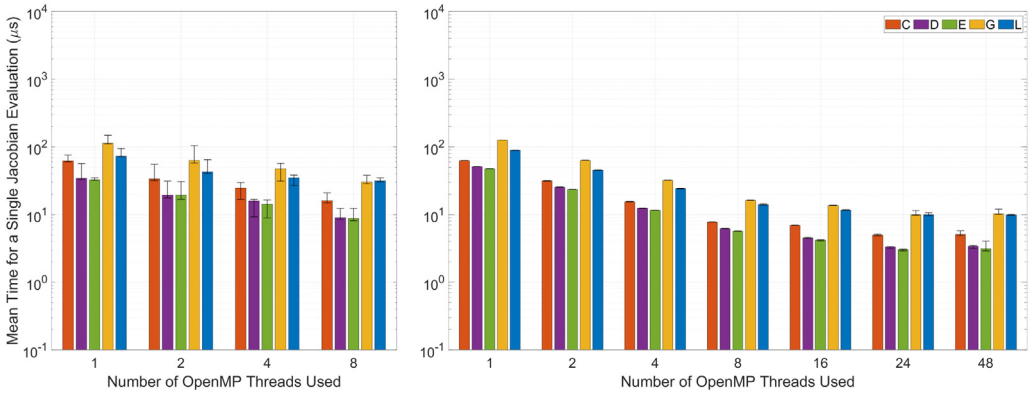

Fig. 10. Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF shell 3443 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors with error bars showing the minimum and maximum measured times.

Table 12
Mean times for a single generalized internal force evaluation for a fully parameterized ANCF shell 3443 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors for different numbers of OpenMP threads for each of the five methods.

Threads	C	D	E	G	L
1	13.49	3.55	3.90	56.43	36.89
2	7.16	2.31	2.20	33.18	25.08
4	5.45	1.26	1.68	24.98	22.96
8	3.53	1.03	1.32	16.80	21.60
1	11.60	4.03	4.67	59.56	36.59
2	5.79	2.03	2.37	30.64	20.00
4	2.89	0.90	1.05	16.00	15.62
8	1.47	0.46	0.45	9.93	11.70
16	1.19	0.36	0.35	10.34	10.17
24	0.83	0.25	0.25	9.25	9.18
48	0.91	0.32	0.32	10.02	9.22

Table 13
Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF shell 3443 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors for different numbers of OpenMP threads for each of the five methods.

Threads	C	D	E	G	L
1	62.25	34.59	32.76	113.43	73.41
2	33.68	19.40	19.42	63.17	42.92
4	24.69	15.98	14.36	47.60	34.96
8	16.08	9.04	8.77	30.28	31.52
1	63.01	51.20	47.53	126.32	89.10
2	31.43	25.35	23.40	63.49	45.39
4	15.51	12.45	11.58	32.09	24.21
8	7.73	6.20	5.69	16.27	14.11
16	6.93	4.52	4.17	13.66	11.69
24	4.99	3.31	3.05	9.98	9.95
48	5.08	3.39	3.14	10.31	9.87

Table 14
Mean times for a single generalized internal force evaluation for a fully parameterized ANCF hexahedral 3843 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors for different numbers of OpenMP threads for each of the five methods.

Threads	C	D	E	G	L
1	45.47	11.07	11.58	564.57	430.28
2	25.34	5.59	6.57	330.95	361.66
4	17.11	5.32	5.53	257.47	338.87
8	10.15	3.62	4.26	207.45	346.96
1	37.11	12.28	13.09	809.54	330.79
2	18.30	6.23	6.81	417.99	247.24
4	9.15	3.05	3.46	223.41	239.47
8	4.56	1.24	1.37	149.48	185.06
16	3.53	0.96	1.04	135.38	160.60
24	2.51	0.69	0.70	124.50	145.42
48	2.63	0.80	0.83	128.76	145.36

Table 15
Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF hexahedral 3843 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors for different numbers of OpenMP threads for each of the five methods.

Threads	C	D	E	G	L
1	356.44	216.68	214.41	1559.98	1057.25
2	200.01	118.65	115.59	877.43	715.87
4	146.09	91.84	93.66	610.53	586.50
8	95.11	59.16	58.15	448.39	530.30
1	378.72	347.43	357.55	2487.82	1097.11
2	188.05	174.12	179.97	1244.72	608.21
4	94.18	87.05	89.91	623.97	401.58
8	47.10	43.48	44.89	316.54	278.21
16	45.01	29.55	29.93	271.38	258.56
24	32.40	22.32	22.55	207.78	243.83
48	32.77	22.48	22.77	209.07	250.29

dynamic simulation shown later in Section 5.3 that still has a relatively low ratio of 7.5 generalized internal force evaluations per Jacobian evaluation due to recalculating the Jacobian once every time step, the break-even point is around 15 to 16 layers.

The Hexahedral 3843 element benchmark test used a mesh of 512 equal elements connected face to face in a single line and 100 iterations of small random changes in the nodal coordinates to generate the statistics. Examining the generalized internal force evaluation times in Fig. 11 and Table 14, like the Shell 3443 element, the "Continuous Integration" methods "C", "D", and "E" have notably lower evaluation times that either of the "Pre-Integration" methods, where again method "D" is generally the fastest. Looking once more at the case of 24 threads on the AMD EYPC 7272 processors, method "C" still requires about 3.6 times as much time as method "D" but methods "G" and "L" have grown to 180 and 210 times the evaluation time of method "D", respectively.

Similar trends are visible for the Jacobian matrix evaluation times shown in Fig. 12 and Table 15. The evaluation times for methods "D" and "E" are very similar. The Jacobian matrix evaluation for method "C" requires approximately 1.5 times as much time as method "E", method "G" requires approximately 9 times as much time as method "E", and method "L" requires approximately 11 times as much time as method "E".

5.2. Roofline analysis

Additional insight into the timing comparisons shown in Section 5.1 can be gained through a roofline analysis of the implemented code [10]. Each of the standalone timing comparisons was profiled with Intel® Advisor on the Intel i7-7700HQ processor to generate cache-aware roofline plots [27] and other fine-grain performance statistics with a

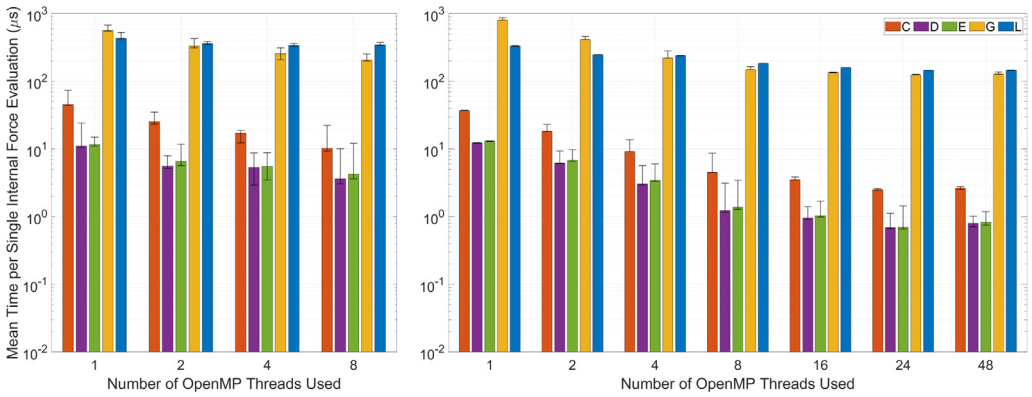

Fig. 11. Mean times for a single generalized internal force evaluation for a fully parameterized ANCF hexahedral 3843 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors with error bars showing the minimum and maximum measured times.

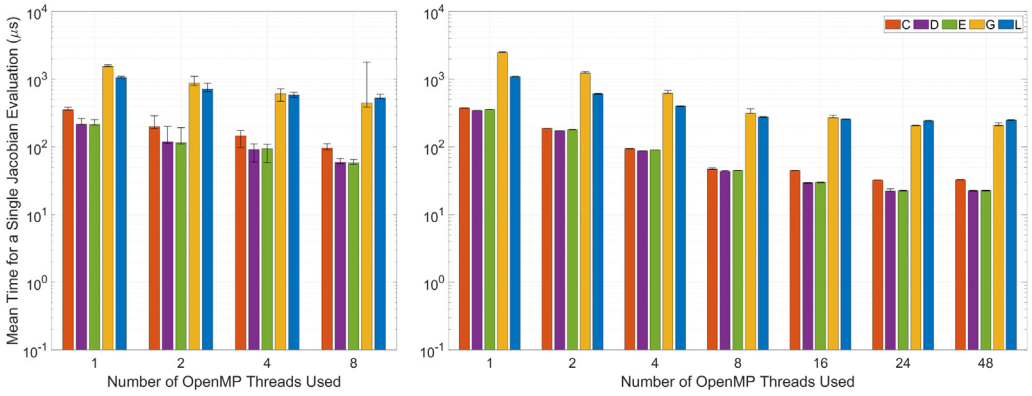

Fig. 12. Mean times for a single Jacobian matrix evaluation for a fully parameterized ANCF hexahedral 3843 element on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors with error bars showing the minimum and maximum measured times.

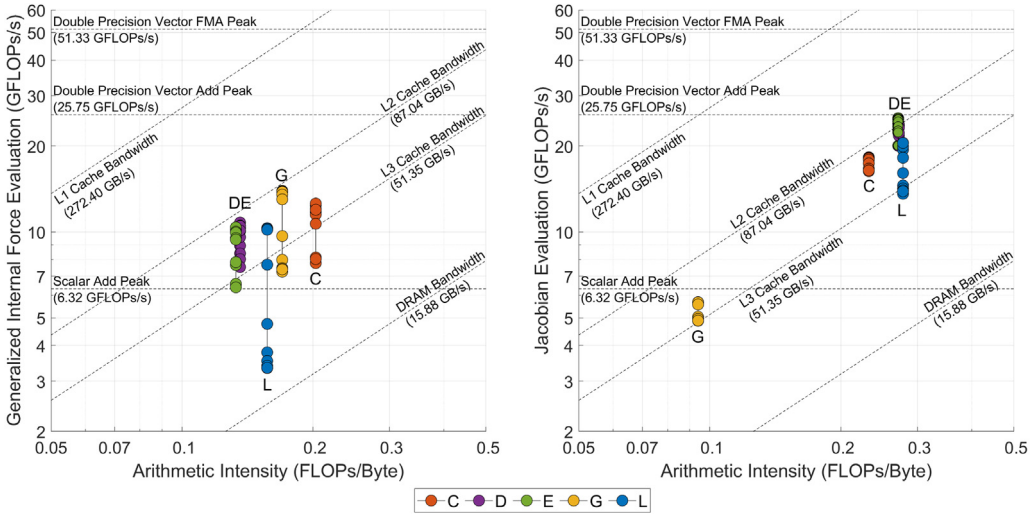

Fig. 13. Intel® Advisor cache-aware roofline plot for the ANCF beam 3243 element generalized internal force evaluation (left) and Jacobian matrix evaluation (right) on the Intel i7-7700HQ processor using a single thread.

variety of different mesh sizes. Since this profiling adds additional overhead, the resulting evaluation times are slightly different than those shown in Section 5.1.

For the roofline plots shown in Figs. 13, 14, and 15, the arithmetic intensity, defined as the number of floating-point operations per byte of data transferred, is plotted against the computational execution rate in GFLOPs/s as measured by Intel® Advisor. This is done for each implementation and mesh size. Along the vertical axis, which lists GFLOPs/second execution speeds, the roofline plot also displays with dotted horizontal lines the measured maximum attainable rates for scalar addition, vectorized addition, and vectorized fused multiply-add (FMA) instructions. For double precision on the Intel i7-7700HQ architecture, as expected, the attainable rate for vectorized additions is four times that of scalar operations; the rate for vectorized FMA instructions is eight times higher than that of scalar operations. To seize these speed gains, the implementation must be designed such

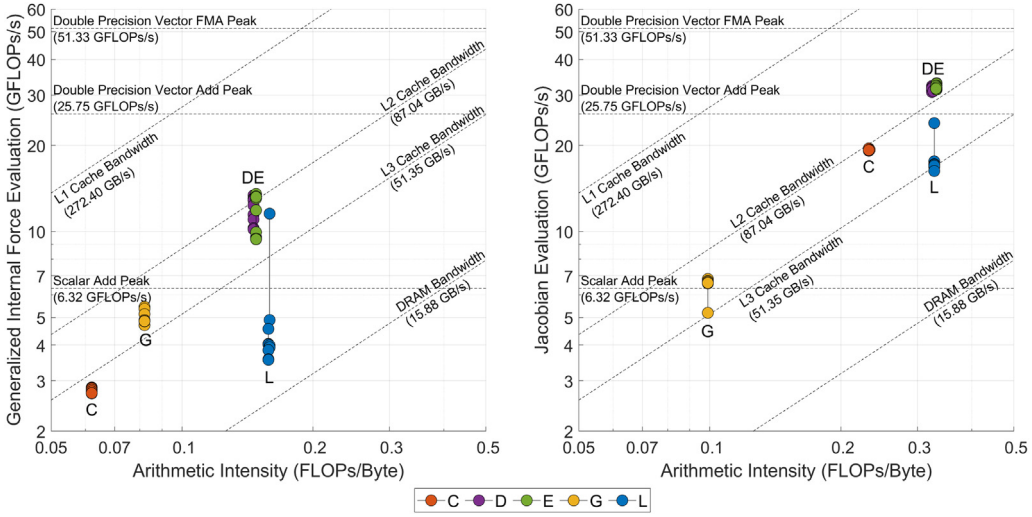

Fig. 14. Intel® Advisor cache-aware roofline plot for the ANCF shell 3243 element generalized internal force evaluation (left) and Jacobian matrix evaluation (right) on the Intel i7-7700HQ processor using a single thread.

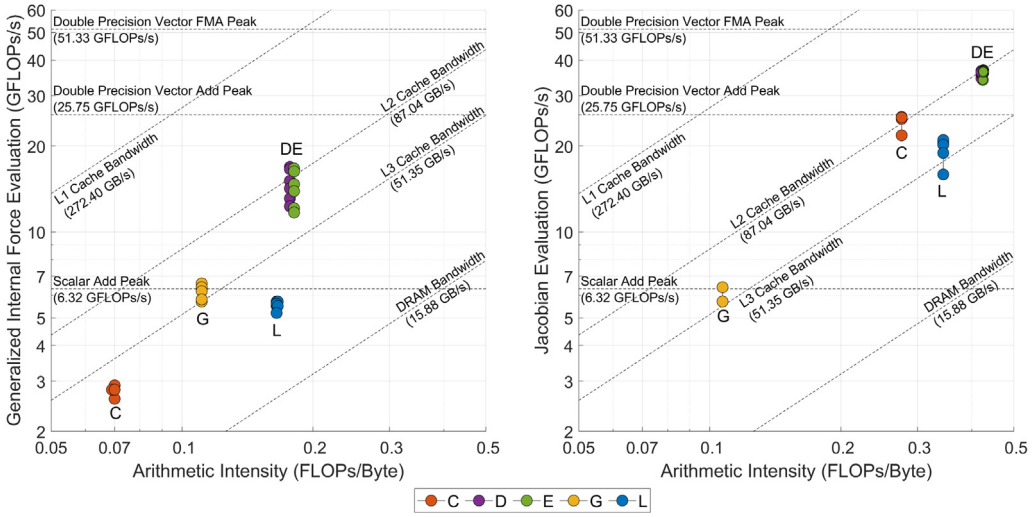

Fig. 15. Intel® Advisor cache-aware roofline plot for the ANCF hexahedral 3843 element generalized internal force evaluation (left) and Jacobian matrix evaluation (right) on the Intel i7-7700HQ processor using a single thread.

that the compiler can issue these AVX2 and/or FMA instructions, which mostly has to do with paying attention to memory alignment of data arrays, and avoiding vector aliasing, recursions, and conditional statements. The diagonal rooflines are set by the bandwidth limitations from main memory and each of the different CPU cache levels. For this computer, the bandwidth of the L1 cache is 17 times larger than that of the main memory. However, L1's size per core is approximately 500,000 times smaller than that of the main memory. As these roofline plots show, both the memory and FLOPs/s limitations of the hardware dictate upper bounds on the performance that can be achieved by an implementation. As a rule of thumb, if the implementation hits the slanted line in the plot, the execution is memory-bound; if one hits "the ceiling", that is, the horizontal part of the plot, the execution is compute-bound. If the execution is neither memory- nor compute-bound, there is internal overhead in the implementation, e.g., thread synchronization, abundance of conditionals, input/output operations, etc. As it is well known and confirmed by these roofline plots, the L2 is faster than L3 (about 60% increase in bandwidth) but it is also significantly smaller – for i7-7700HQ one has 32 KiB of L1 data cache per core, 256 KiB of L2 cache per core, and 6 MiB of L3 cache per processor. As such, "G" and "L", which have a very large memory footprint will exhaust for sure L2 and even the L3 cache and engage in transactions with the main memory. As the roofline plot shows, while a memory transaction from L1 cache is serviced at 272.4 GiB/s, it only clocks in at 15.88 GiB/s if serviced by the main memory. What obfuscates the performance analysis is the fact that each core engages in prefetching, which relieves some of the cache pressure.

Looking first at the generalized internal force evaluations in Figs. 13, 14, and 15 along with Tables 22, 24, and 26, the execution speed of method "L" is likely limited by the speed of main memory since its average execution rate is below the scalar addition roof for all but a few of the test cases. This is likely due to the inherent limitations of the matrix–vector product that composes the majority of these computations. This aligns with the ratios previously shown in Table 8. For methods "C" and "G", which both contain a fairly large number of smaller matrix multiplications, the compiler is able to better optimize these implementations for the Beam 3243 element than for the Shell 3443 or Hexahedral 3843 elements. The execution rates of methods "D" and "E" stay relatively constant with only a slight increase in speed as the complexity of the element increases. Looking at the tables, the large amount of stored data can be seen in the transfers from main memory through cache, especially for the Shell 3443 and Hexahedral 3843

Table 16
Intel® Advisor measured summary results for the ANCF beam 3243 generalized internal force evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using different mesh sizes with a single thread for each of the five example methods.

Average evaluation time (μs)											
Method	8	16	32	64	128	256	512	1024	2048	4096	8192
C	0.69	0.67	0.66	0.71	0.71	0.72	0.70	0.78	1.07	1.02	1.04
D	1.03	0.92	0.89	0.85	0.87	0.96	0.88	0.91	1.09	1.15	1.22
E	1.20	0.90	0.89	0.92	0.93	0.97	0.98	1.17	1.40	1.40	1.44
G	1.57	1.57	1.58	1.62	1.68	2.26	2.74	2.92	2.95	3.01	2.95
L	0.91	0.92	0.92	0.92	1.22	1.97	2.48	2.66	2.77	2.82	2.81
Average total arithmetic intensity											
C	0.203	0.203	0.203	0.203	0.203	0.203	0.203	0.203	0.203	0.203	0.203
D	0.136	0.136	0.136	0.136	0.136	0.136	0.136	0.136	0.136	0.136	0.136
E	0.133	0.133	0.133	0.133	0.133	0.133	0.133	0.133	0.133	0.133	0.133
G	0.170	0.170	0.170	0.170	0.170	0.170	0.170	0.170	0.170	0.170	0.170
L	0.157	0.157	0.157	0.157	0.157	0.157	0.157	0.157	0.157	0.157	0.157
Average total GFLOP/s											
C	12.0	12.4	12.6	11.7	11.8	11.5	12.0	10.7	7.8	8.1	8.0
D	8.9	10.0	10.3	10.8	10.5	9.6	10.5	10.1	8.4	8.0	7.5
E	7.7	10.2	10.4	10.0	9.9	9.5	9.4	7.8	6.6	6.6	6.4
G	13.9	13.9	13.8	13.5	13.0	9.7	8.0	7.5	7.4	7.3	7.4
L	10.3	10.2	10.3	10.2	7.7	4.8	3.8	3.5	3.4	3.3	3.3
DRAM memory transfers (MB)											
C	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.004	0.004
D	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.004	0.004
E	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.008	0.011	0.011	0.011
G	0.000	0.000	0.000	0.000	0.000	0.021	0.033	0.033	0.033	0.033	0.033
L	0.000	0.000	0.000	0.000	0.002	0.034	0.035	0.036	0.035	0.036	0.036

Table 17
Intel® Advisor measured summary results for the ANCF beam 3243 Jacobian matrix evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using different mesh sizes with a single thread for each of the five example methods.

Average evaluation time (μs)											
Method	8	16	32	64	128	256	512	1024	2048	4096	8192
C	8.0	7.2	7.2	7.3	7.3	7.3	7.3	7.5	7.9	8.0	8.0
D	5.8	5.0	4.9	4.9	4.9	4.9	4.9	5.0	5.2	5.4	5.2
E	5.5	4.4	4.4	4.4	4.4	4.6	4.5	4.7	4.9	4.9	4.9
G	9.0	9.1	9.0	9.0	9.1	10.1	10.3	10.4	10.4	10.4	10.4
L	3.9	3.8	3.7	3.7	4.2	4.7	5.2	5.4	5.6	5.4	5.5
Average total arithmetic intensity											
C	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232	0.232
D	0.272	0.272	0.272	0.272	0.272	0.272	0.272	0.272	0.272	0.272	0.272
E	0.271	0.271	0.271	0.271	0.271	0.271	0.271	0.271	0.271	0.271	0.271
G	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094
L	0.278	0.278	0.278	0.278	0.278	0.278	0.278	0.278	0.278	0.278	0.278
Average total GFLOP/s											
C	16.3	18.3	18.2	18.1	18.1	17.9	17.9	17.4	16.7	16.4	16.4
D	20.0	23.4	23.7	23.8	23.7	23.7	23.6	23.2	22.2	21.7	22.2
E	20.0	25.1	24.9	24.8	24.8	24.0	24.4	23.5	22.7	22.7	22.3
G	5.7	5.6	5.7	5.7	5.6	5.0	5.0	4.9	4.9	4.9	4.9
L	19.4	19.9	20.5	20.5	18.2	16.1	14.5	14.2	13.6	14.0	13.9
DRAM memory transfers (MB)											
C	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.005	0.005
D	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.004	0.005
E	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.006	0.007	0.008	0.007
G	0.000	0.000	0.000	0.000	0.000	0.020	0.029	0.029	0.029	0.029	0.029
L	0.000	0.000	0.000	0.000	0.002	0.033	0.035	0.035	0.035	0.035	0.035

elements. The measured number of FLOPs is also listed in the tables and all these values are just slightly higher than the hand-calculated values shown in Table 6.

Looking at the Jacobian matrix evaluations in Figs. 13, 14, and 15 along with Tables 23, 25, and 27, with the exception of method "G", both the arithmetic intensity and computational rates for the Jacobians are higher than those of the force evaluations. Note that for the Shell 3443 and Hexahedral 3843 elements, "D" and "E" are clearly using vectorized FMA instructions since the average computational rate is above the double precision vectorized arithmetic roof. The arithmetic intensity of "G" is by comparison low owing to how data is accessed in this implementation, and the increase in the amount of memory access offsets the lower number of required FLOPs. The approximately five-fold difference in computational rates seen in Table 27 between method "D" and "G" is an example of why FLOP counts alone should not be used to gauge the performance of implementations with similar orders of magnitude of operations. Finally, note that as was the case for the force calculations, the actual FLOP counts for the Jacobian are just slightly higher than the hand-calculated estimates provided in Table 7.

As discussed in Section 5.1, the effect of using different mesh sizes is clearly seen in the roofline plots shown in Figs. 13, 14, and 15 and the tabular form of the data in Tables 16, 17, 18, 19, 20,

Table 18
Intel® Advisor measured summary results for the ANCF shell 3443 generalized internal force evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using different mesh sizes with a single thread for each of the five example methods.

Average evaluation time (μs)											
Method	8	16	32	64	128	256	512	1024	2048	4096	8192
C	12.9	12.9	12.9	13.0	13.0	13.1	13.1	13.3	13.5	13.5	13.5
D	2.7	2.8	2.9	2.8	2.9	3.0	3.2	3.3	3.6	3.6	3.6
E	2.7	2.8	2.8	2.8	2.8	3.1	3.8	3.7	3.9	3.9	3.9
G	50.6	51.4	53.7	56.7	58.7	56.3	56.7	56.9	56.7	56.8	56.8
L	11.7	27.7	29.8	33.6	33.7	34.0	34.5	34.5	35.3	38.0	38.1
Average total arithmetic intensity											
C	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062	0.062
D	0.146	0.146	0.146	0.146	0.146	0.146	0.146	0.146	0.146	0.146	0.146
E	0.148	0.148	0.148	0.148	0.148	0.148	0.148	0.148	0.148	0.148	0.148
G	0.082	0.082	0.082	0.082	0.082	0.082	0.082	0.082	0.082	0.082	0.082
L	0.159	0.159	0.158	0.158	0.158	0.159	0.159	0.159	0.158	0.158	0.158
Average total GFLOP/s											
C	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.7	2.7	2.7
D	13.4	13.0	12.8	13.0	12.9	12.3	11.4	11.0	10.3	10.3	10.2
E	13.5	13.2	13.0	13.3	13.2	11.9	9.6	9.9	9.4	9.4	9.4
G	5.4	5.4	5.1	4.9	4.7	4.9	4.9	4.8	4.9	4.9	4.9
L	11.5	4.9	4.6	4.0	4.0	4.0	3.9	3.9	3.8	3.6	3.6
DRAM memory transfers (MB)											
C	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.014	0.015	0.015	0.015
D	0.000	0.000	0.000	0.000	0.000	0.000	0.010	0.014	0.015	0.015	0.015
E	0.000	0.000	0.000	0.000	0.000	0.008	0.027	0.027	0.027	0.027	0.027
G	0.000	0.098	0.332	0.335	0.335	0.335	0.335	0.335	0.335	0.335	0.335
L	0.027	0.528	0.532	0.532	0.532	0.532	0.532	0.532	0.532	0.532	0.532

Table 19
Intel® Advisor measured summary results for the ANCF shell 3443 Jacobian matrix evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using different mesh sizes with a single thread for each of the five example methods.

Average evaluation time (μs)											
Method	8	16	32	64	128	256	512	1024	2048	4096	8192
C	61	60	60	60	61	60	61	60	61	60	61
D	33	34	32	33	33	33	34	33	34	33	34
E	31	31	31	31	32	32	33	32	32	32	32
G	141	110	108	110	110	112	110	109	110	111	111
L	50	68	70	71	70	70	70	70	70	73	73
Average total arithmetic intensity											
C	0.232	0.232	0.232	0.232	0.232	0.233	0.232	0.233	0.232	0.233	0.232
D	0.324	0.324	0.324	0.325	0.324	0.324	0.324	0.325	0.324	0.324	0.326
E	0.332	0.332	0.332	0.332	0.332	0.331	0.332	0.332	0.332	0.332	0.331
G	0.099	0.099	0.099	0.099	0.099	0.099	0.099	0.099	0.099	0.099	0.099
L	0.328	0.328	0.328	0.328	0.328	0.328	0.328	0.328	0.328	0.328	0.328
Average total GFLOP/s											
C	19.1	19.5	19.4	19.5	19.2	19.3	19.2	19.3	19.3	19.3	19.3
D	31.9	31.2	32.3	32.0	31.6	31.5	31.1	31.3	30.8	31.7	31.0
E	32.9	33.0	32.9	32.9	32.4	32.3	31.4	31.7	31.6	31.9	31.6
G	5.2	6.7	6.8	6.7	6.7	6.6	6.7	6.7	6.7	6.6	6.6
L	23.9	17.6	17.2	16.8	17.2	17.1	17.1	17.1	17.0	16.3	16.3
DRAM memory transfers (MB)											
C	0.000	0.000	0.000	0.000	0.000	0.000	0.012	0.016	0.016	0.016	0.016
D	0.000	0.000	0.000	0.000	0.000	0.001	0.012	0.016	0.016	0.016	0.016
E	0.000	0.000	0.000	0.000	0.000	0.009	0.021	0.021	0.022	0.021	0.022
G	0.000	0.109	0.316	0.319	0.318	0.319	0.318	0.318	0.319	0.318	0.318
L	0.041	0.524	0.530	0.529	0.530	0.529	0.529	0.529	0.529	0.529	0.529

and 21. As the mesh sizes increase, the amount of required memory transactions also increases while the number of required FLOPs remains constant. These increased memory transactions can significantly affect the overall execution performance, especially as data is evicted from additional cache levels. This is clearly highlighted by the transition in the execution times for the beam element generalized internal force calculations for method "L" where data starts to be evicted out of L3 cache with a mesh size of 128 elements or larger. With this dependency on memory, it is important to note that the results presented here are measured outside of the context of a simulation. As such, memory demands from the rest of the simulation computations dictating what data can remain in cache at any given time are not accounted for here.

5.3. Relative simulation step execution times

To put the timing results shown earlier in this section into context, a series of dynamic simulations were run with each implementation within Project Chrono [11]. The Hilber–Hughes–Taylor (HHT) [28] implicit integrator was used along with the Eigen [29] SparseLU linear solver to execute one-second long simulations with a fixed time step

Table 20
Intel® Advisor measured summary results for the ANCF hexahedral 3843 generalized internal force evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using different mesh sizes with a single thread for each of the five example methods.

Average evaluation time (μs)							
Method	8	16	32	64	128	256	512
C	46	50	44	44	45	45	45
D	8	8	8	8	9	10	10
E	8	9	8	8	9	10	11
G	592	563	565	578	601	655	640
L	381	368	373	372	373	382	405
Average total arithmetic intensity							
C	0.070	0.070	0.070	0.070	0.070	0.069	0.070
D	0.177	0.177	0.177	0.177	0.177	0.177	0.177
E	0.181	0.181	0.181	0.181	0.181	0.181	0.181
G	0.111	0.111	0.111	0.111	0.111	0.111	0.111
L	0.165	0.165	0.165	0.166	0.165	0.166	0.165
Average total GFLOP/s							
C	2.8	2.6	2.9	2.9	2.8	2.8	2.8
D	16.6	15.1	16.9	16.6	14.2	13.1	12.3
E	16.3	14.7	16.7	16.3	13.9	12.1	11.7
G	6.3	6.6	6.6	6.4	6.2	5.7	5.8
L	5.5	5.7	5.6	5.7	5.6	5.5	5.2
DRAM memory transfers (MB)							
C	0.000	0.000	0.000	0.000	0.032	0.053	0.053
D	0.000	0.000	0.000	0.000	0.029	0.053	0.053
E	0.000	0.000	0.000	0.000	0.069	0.078	0.078
G	4.539	4.543	4.544	4.543	4.543	4.543	4.544
L	8.412	8.437	8.416	8.441	8.447	8.438	8.454

Table 21
Intel® Advisor measured summary results for the ANCF hexahedral 3843 Jacobian matrix evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using different mesh sizes with a single thread for each of the five example methods.

Average evaluation time (μs)							
Method	8	16	32	64	128	256	512
C	351	406	351	350	356	350	352
D	212	225	215	215	218	216	215
E	209	226	209	211	214	211	212
G	1575	1560	1573	1565	1570	1565	1765
L	1196	915	934	933	906	942	1003
Average total arithmetic intensity							
C	0.276	0.276	0.276	0.276	0.276	0.276	0.276
D	0.421	0.421	0.421	0.421	0.421	0.421	0.421
E	0.425	0.424	0.425	0.425	0.425	0.424	0.425
G	0.107	0.107	0.107	0.107	0.107	0.107	0.107
L	0.344	0.344	0.344	0.344	0.344	0.344	0.344
Average total GFLOP/s							
C	25.3	21.8	25.3	25.3	24.9	25.3	25.2
D	36.7	34.6	36.3	36.2	35.7	36.1	36.3
E	36.9	34.1	36.8	36.4	36.1	36.6	36.4
G	6.4	6.4	6.4	6.4	6.4	6.4	5.7
L	15.9	20.8	20.3	20.4	21.0	20.2	18.9
DRAM memory transfers (MB)							
C	0.003	0.000	0.002	0.001	0.045	0.058	0.058
D	0.000	0.000	0.000	0.000	0.043	0.057	0.057
E	0.000	0.000	0.000	0.002	0.066	0.068	0.068
G	4.539	4.515	4.535	4.521	4.492	4.519	4.496
L	8.782	8.812	8.881	8.848	8.850	8.825	8.833

Table 22
Intel® Advisor measured summary results for the ANCF beam 3243 generalized internal force evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using a mesh of 1024 elements with a single thread for each of the five example methods.

Method	Internal force evaluation time (μs)	Total arithmetic intensity	Total kFLOP	Total GFLOP/s	L1 memory transfers (MB)	L2 memory transfers (MB)	L3 memory transfers (MB)	DRAM memory transfers (MB)	Avg memory speed (GB/s)
C	0.8	0.203	8.3	10.7	0.041	0.004	0.004	0.000	52
D	0.9	0.136	9.2	10.1	0.068	0.005	0.004	0.000	74
E	1.2	0.133	9.2	7.8	0.069	0.015	0.011	0.008	59
G	2.9	0.170	21.8	7.5	0.128	0.035	0.033	0.033	44
L	2.7	0.157	9.4	3.5	0.060	0.037	0.036	0.036	22

Table 23
Intel® Advisor measured summary results for the ANCF beam 3243 Jacobian matrix evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using a mesh of 1024 elements with a single thread for each of the five example methods.

Method	Jacobian evaluation time (μs)	Total arithmetic intensity	Total kFLOP	Total GFLOP/s	L1 memory transfers (MB)	L2 memory transfers (MB)	L3 memory transfers (MB)	DRAM memory transfers (MB)	Avg memory speed (GB/s)
C	7.5	0.232	131	17.4	0.566	0.012	0.004	0.000	75
D	5.0	0.272	117	23.2	0.429	0.249	0.004	0.000	85
E	4.7	0.271	110	23.5	0.406	0.237	0.007	0.006	86
G	10.4	0.094	51	4.9	0.541	0.040	0.029	0.029	52
L	5.4	0.278	76	14.2	0.274	0.230	0.035	0.035	51

Table 24
Intel® Advisor measured summary results for the ANCF shell 3443 generalized internal force evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using a mesh of 1024 elements with a single thread for each of the five example methods.

Method	Internal force evaluation time (μs)	Total arithmetic intensity	Total kFLOP	Total GFLOP/s	L1 memory transfers (MB)	L2 memory transfers (MB)	L3 memory transfers (MB)	DRAM memory transfers (MB)	Avg memory speed (GB/s)
C	13.3	0.062	37	2.8	0.596	0.014	0.014	0.014	45
D	3.3	0.146	37	11.0	0.252	0.082	0.014	0.014	76
E	3.7	0.148	37	9.9	0.248	0.092	0.027	0.027	67
G	56.9	0.082	276	4.8	3.346	0.393	0.345	0.335	59
L	34.5	0.159	136	3.9	0.854	0.551	0.550	0.532	25

Table 25
Intel® Advisor measured summary results for the ANCF shell 3443 Jacobian matrix evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using a mesh of 1024 elements with a single thread for each of the five example methods.

Method	Jacobian evaluation time (μs)	Total arithmetic intensity	Total kFLOP	Total GFLOP/s	L1 memory transfers (MB)	L2 memory transfers (MB)	L3 memory transfers (MB)	DRAM memory transfers (MB)	Avg memory speed (GB/s)
C	60	0.233	1167	19.3	5.017	1.372	0.015	0.016	83
D	33	0.325	1046	31.3	3.221	1.274	0.016	0.016	96
E	32	0.332	1022	31.7	3.082	1.267	0.022	0.021	95
G	109	0.099	734	6.7	7.419	0.670	0.374	0.318	68
L	70	0.328	1194	17.1	3.637	2.743	0.714	0.529	52

Table 26
Intel® Advisor measured summary results for the ANCF hexahedral 3843 generalized internal force evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using a mesh of 512 elements with a single thread for each of the five example methods.

Method	Internal force evaluation time (μs)	Total arithmetic intensity	Total kFLOP	Total GFLOP/s	L1 memory transfers (MB)	L2 memory transfers (MB)	L3 memory transfers (MB)	DRAM memory transfers (MB)	Avg memory speed (GB/s)
C	45	0.070	127	2.8	1.827	0.056	0.052	0.053	41
D	10	0.177	127	12.3	0.718	0.564	0.086	0.053	70
E	11	0.181	127	11.7	0.701	0.519	0.110	0.078	64
G	640	0.111	3706	5.8	33.339	5.760	4.754	4.544	52
L	405	0.165	2108	5.2	12.758	12.905	8.470	8.454	32

Table 27
Intel® Advisor measured summary results for the ANCF hexahedral 3843 Jacobian matrix evaluation on the Intel i7-7700HQ processor. The results are averaged for a single element calculated using a mesh of 512 elements with a single thread for each of the five example methods.

Method	Jacobian evaluation time (μs)	Total arithmetic intensity	Total kFLOP	Total GFLOP/s	L1 memory transfers (MB)	L2 memory transfers (MB)	L3 memory transfers (MB)	DRAM memory transfers (MB)	Avg memory speed (GB/s)
C	352	0.276	8 860	25.2	32.153	13.240	0.100	0.058	91
D	215	0.421	7 791	36.3	18.507	9.801	2.817	0.057	86
E	212	0.425	7 704	36.4	18.146	9.481	2.709	0.068	86
G	1765	0.107	10 040	5.7	93.963	27.741	5.881	4.496	53
L	1003	0.344	18 998	18.9	55.187	38.606	10.188	8.833	55

of 10^{-3} s. A modified Newton–Raphson method was used where the Jacobian matrix was evaluated and factorized once per time step. While other techniques exist to reduce the number of Jacobian evaluations and factorizations, the focus here is on gauging the relative cost of the ANCF calculations versus the cost of the linear algebra required to solve the system of equations of motion.

For the Beam 3243 element, a uniform mesh of 1024 beam elements was generated based on the three-dimensional pendulum discussed in [26], where a spherical joint was used to attach the square cross-section pendulum to ground. Examining the contributions to total wall times for each of the methods shown in Fig. 16, the largest portion of time is due to the LU factorization of the Jacobian and not the

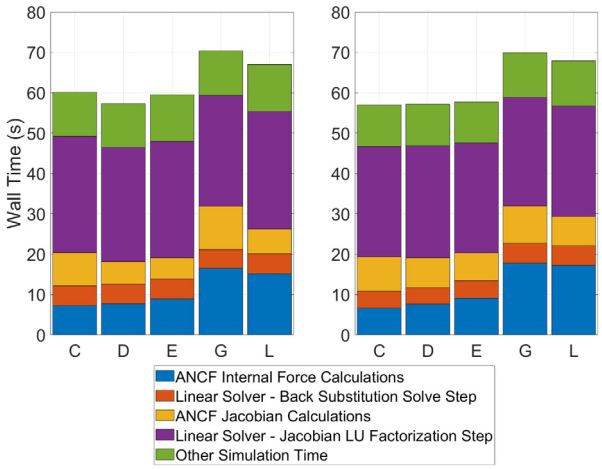

Fig. 16. Wall clock times for a 1 s simulations with a constant time step of 10^{-3} s for a 3D pendulum with a linear mesh composed of 1024 ANCF Beam 3243 elements conducted on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors using a single OpenMP thread.

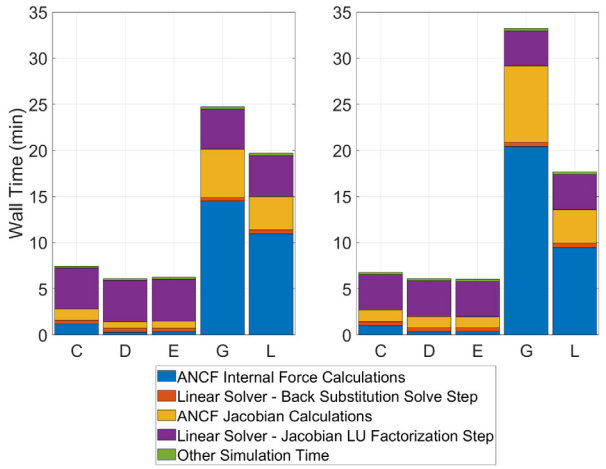

Fig. 18. Wall clock times for a 1 s simulations with a constant time step of 10^{-3} s for a 3D simple plate pendulum with a mesh composed of 200 ANCF Hexahedral 3843 elements conducted on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors using a single OpenMP thread.

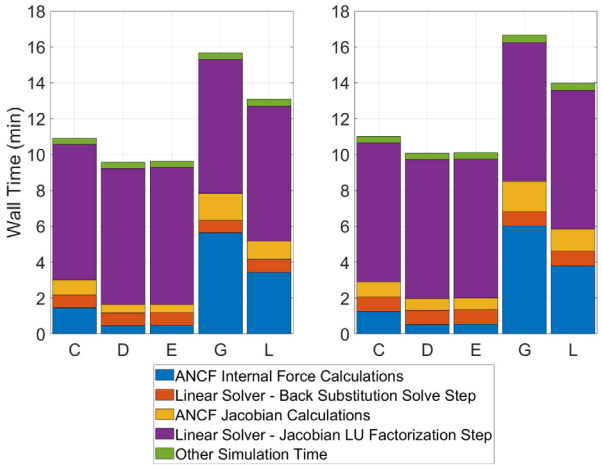

Fig. 17. Wall clock times for a 1 s simulations with a constant time step of 10^{-3} s for a 3D simple plate pendulum with a mesh composed of 800 ANCF Shell 3443 elements conducted on the Intel i7-7700HQ (left) and the AMD EYPC 7272 (right) processors using a single OpenMP thread.

ANCF Jacobian matrix evaluations. For the generalized internal force evaluations, the cost of the ANCF calculations is higher than those for the linear solver back substitution step. While it may appear from the figure that the generalized internal force evaluations are more costly than the Jacobian evaluations, this is because on average 4.9 generalized internal force evaluations were required per Jacobian matrix evaluation.

For the Shell 3443 element, the plate pendulum described in [14] is used with a mesh of 20 by 40 elements and a fixed time step of 10^{-3} s. Examining the relative execution time contributions for each method shown in Fig. 17, the Jacobian LU factorization once again composes the largest percent of the overall time even though on average 7.5 generalized internal force evaluations occurred per Jacobian evaluation. For all the methods, the ANCF Jacobian evaluation times are notably shorter than the LU factorizations. For methods "C", "G", and "L", the generalized internal force evaluation times are longer than the linear solver back-substitution step, while for methods "D" and "E", the back-substitution step is longer. Even though the generalized internal force and Jacobian evaluations for methods "C", "D", and "E" are notably shorter, the simulations for methods "G" and "L" only take about 1.5 times as long since the linear algebra steps compose a significant portion of the overall simulation time and eclipse gains in force/Jacobian calculations.

For the Hexahedral 3843 element, the same plate pendulum from the Shell 3443 simulations was reused here. The mesh had 10 by 20 by 1 elements, and the spherical joint was relocated from the midsurface to the bottom corner of the plate. On average 7.6 generalized internal force evaluations occurred per Jacobian evaluation for this simulation. Examining the relative execution time contributions for each method shown in Fig. 18, the solution component with the longest execution time depends on the method. For methods "G" and "L", the generalized internal force evaluations require the most time. For method "G", the Jacobian matrix evaluation time is longer than the Jacobian LU factorization. Looking at methods "C", "D", and "E", the Jacobian LU factorization again has the largest contribution to the overall execution time. For methods "D" and "E" the linear solver back substitution step is still longer than the generalized internal force evaluation. For this element, larger differences in the total simulation times can be seen with method "G" requiring roughly 5 times longer than methods "D" or "E".

6. Conclusions

The main conclusion of this work is that significant differences in terms of performance exist among the methods proposed in the literature to solve the same nonlinear flexible body dynamics problem within the ANCF framework. The three "Continuous Integration" implementations focused on making the integration across the volume of the element during each generalized internal force evaluation as efficient as possible. The two "Pre-Integration" methods mathematically separated out the nodal coordinates from the integration across the volume of the element. As a result, the integration across the volume of the element is only required once prior to the start of the simulation. While on the surface this may sound advantageous, it was demonstrated that the "Pre-Integration" methods were more sluggish both for the generalized internal force and its Jacobian matrix. Moreover, the "Continuous Integration" methods generally require significantly less storage than either of the two "Pre-Integration" methods, which enables simulations with much larger mesh sizes. As the complexity of an ANCF element increases, the relative advantages of the "Continuous Integration" methods also increase. For the Shell 3443 element used in

this comparison, a conservative number of Gauss quadrature points was chosen for this study. If a slightly smaller number of Gauss quadrature points was chosen for this element as discussed in Section 3, the relative advantages of the "Continuous Integration" methods over the "Pre-Integration" methods would increase even more. Additionally, as discussed in Section 4.2, if a purely linear elastic material law was used instead of the linear viscoelastic material law, then the relative advantages of the "Continuous Integration" methods would again increase. Although not discussed further in this paper, it was found that especially for the more complex elements, both of the "Pre-Integration" methods require a significantly larger amount of calculations during the initialization phase of a simulation. While for longer simulations this increased initialization time is likely not problematic, it could be detrimental for shorter simulations.

While the "Continuous Integration" methods were generally shown to have both the lowest execution times and the lowest required memory storage cost, there are cases in which the "Pre-Integration" style methods have the upper hand. Since both the evaluation times and memory storage costs for the "Continuous Integration" methods scale with the number of Gauss quadrature points used to integrate across the volume of the element, complex beam cross-sections or shells with many discrete layers could shift the advantage to the "Pre-Integration" style methods. However, for the case of the Shell 3443 element where a single equivalent layer could not be used, the number of layers required to shift the advantage to the "Pre-Integration" style methods was quite large.

We also noted that characteristics of the target compute hardware, e.g., cache sizes and bandwidths, AVX and FMA support, instruction level parallelism as reflected in pre-fetching and pipelining, play a critical role in defining the implementation's speed of execution. The approximately five-fold difference in FLOP rates shown in Section 5.2 highlights that while back-of-the-envelope estimations of FLOP counts provide some information, performance comparisons should be conducted by running on actual hardware. This is because the optimizations taking place on the processor, under the hood and with the possible participation of the compiler, are too complex to be factored in when one simply reads about an algorithm in a paper.

For the implementations compared, different trends could potentially exist under other comparison conditions and/or compute hardware. With this caveat, as shown in these comparisons with multicore CPU architectures, the generalized internal force and Jacobian evaluation implementation "D" and "E" introduced in [6] emerged as the more competitive ones both for execution speeds and the required amount of data storage per element.

CRediT authorship contribution statement

Michael Taylor: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Radu Serban:** Methodology, Software, Writing – review & editing, Supervision. **Dan Negrut:** Methodology, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This material is based upon work that was financially supported by: (a) the U.S. National Science Foundation (NSF) under Grants CISE1835674 OAC2209791 and (b) the U.S. Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) under Contract W56HZV-17-C-0095. The work was done to support: (a) the NSF project "Chrono - An Open-Source Simulation Platform for Computational Dynamics Problems" and (b) the HPCMP CREATE™-Ground Vehicles (CREATE-GV) portion of the Computational Research and Engineering Acquisition Tools and Environments (HPCMP CREATE™) Portfolio being executed by the DoD HPCMP Office. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. DoD or the NSF.

Appendix A. Math behind implementation "G"

While García-Vallejo et al. [3] originally presented their calculation method with a simple linear elastic material model and later on with a viscoelastic material model with two coefficients to control the damping effects [8], this presentation will demonstrate how to adapt this method for use with the simple linear viscoelastic material model used for the comparisons in this paper. This simpler viscoelastic material model, presented by Zhao et al. [9], only requires a single coefficient for the damping contribution. Local nodal coordinates will be assumed for this presentation, but these equations can easily be adapted to use global nodal coordinates as shown for methods "C", "D", and "E" in [6] to account for mesh discontinuities as described in [30].

The starting point is the equation defining the position of a material point within the ANCF element

$$r(\xi,\eta,\zeta,t) = \mathbf{S}(\xi,\eta,\zeta)\,\mathbf{e}(t)\,, \tag{2}$$

where it has been assumed that the shape function matrix S has been written in terms of normalized element coordinates that each span from -1 to 1. The shape function matrix is composed of the N unique shape functions each multiplied by the 3×3 identity matrix

$$\mathbf{S} = \begin{bmatrix} S_1\mathbf{I}_{3\times3} & S_2\mathbf{I}_{3\times3} & \dots & S_N\mathbf{I}_{3\times3}\end{bmatrix}, \tag{3}$$

where N is dependent on the specific element being implemented. Using the equation for the position within the element, the deformation gradient, $\mathbf{F}$ within the element can be written as

$$\begin{aligned}\mathbf{F} &= \frac{\partial\mathbf{r}}{\partial\mathbf{r}_0} = \frac{\partial\mathbf{r}}{\partial\boldsymbol{\xi}}\left(\frac{\partial\mathbf{r}_0}{\partial\boldsymbol{\xi}}\right)^{-1} = \frac{\partial\mathbf{r}}{\partial\boldsymbol{\xi}}\left(\mathbf{J}_{0\xi}^{-1}\right)\\ &= \begin{bmatrix}\frac{\partial\mathbf{S}}{\partial\xi}\mathbf{e} & \frac{\partial\mathbf{S}}{\partial\eta}\mathbf{e} & \frac{\partial\mathbf{S}}{\partial\zeta}\mathbf{e}\end{bmatrix}\begin{bmatrix}\frac{\partial\mathbf{S}}{\partial\xi}\mathbf{e}_0 & \frac{\partial\mathbf{S}}{\partial\eta}\mathbf{e}_0 & \frac{\partial\mathbf{S}}{\partial\zeta}\mathbf{e}_0\end{bmatrix}^{-1}\\ &= \begin{bmatrix}\mathbf{S}_{,1}\mathbf{e} & \mathbf{S}_{,2}\mathbf{e} & \mathbf{S}_{,3}\mathbf{e}\end{bmatrix},\end{aligned} \tag{4}$$

where

$$\mathbf{S}_{,i} = \left(\mathbf{J}_{0\xi}^{-1}\right)_{1i}\frac{\partial\mathbf{S}}{\partial\xi} + \left(\mathbf{J}_{0\xi}^{-1}\right)_{2i}\frac{\partial\mathbf{S}}{\partial\eta} + \left(\mathbf{J}_{0\xi}^{-1}\right)_{3i}\frac{\partial\mathbf{S}}{\partial\zeta}\,. \tag{5}$$

Using the expression for the deformation gradient, the Green–Lagrange strain tensor, $\mathbf{E}$, can be written in Voigt notation as

$$\boldsymbol{\varepsilon} = \begin{bmatrix}E_{11}\\E_{22}\\E_{33}\\2E_{23}\\2E_{13}\\2E_{12}\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\left(\mathbf{e}^{\mathsf{T}}\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,1}\mathbf{e}-1\right)\\ \frac{1}{2}\left(\mathbf{e}^{\mathsf{T}}\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,2}\mathbf{e}-1\right)\\ \frac{1}{2}\left(\mathbf{e}^{\mathsf{T}}\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,3}\mathbf{e}-1\right)\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,3}+\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,2}\right)\mathbf{e}\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,3}+\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,1}\right)\mathbf{e}\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,2}+\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,1}\right)\mathbf{e}\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(1)}\mathbf{e}-\frac{1}{2}\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(2)}\mathbf{e}-\frac{1}{2}\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(3)}\mathbf{e}-\frac{1}{2}\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(4)}\mathbf{e}\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(5)}\mathbf{e}\\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(6)}\mathbf{e}\end{bmatrix}. \tag{6}$$

Next, the time derivative of the Green–Lagrange strain tensor, $\mathbf{E}$, can be written in Voigt notation as

$$\dot{\boldsymbol{\varepsilon}} = \begin{bmatrix} \mathbf{e}^{\mathsf{T}}\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,1}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,2}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,3}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,3}+\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,2}\right)\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,3}+\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,1}\right)\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,2}+\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,1}\right)\dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(1)}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(2)}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(3)}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(4)}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(5)}\dot{\mathbf{e}} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(6)}\dot{\mathbf{e}} \end{bmatrix}. \tag{7}$$

The last term needed before defining the expression for the generalized internal force is the partial derivative of the Green–Lagrange strains with respect to the nodal coordinates, which can be written as

$$\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{e}} = \begin{bmatrix} \mathbf{e}^{\mathsf{T}}\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,1} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,2} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,3} \\ \mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,3}+\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,2}\right) \\ \mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,3}+\mathbf{S}_{,3}^{\mathsf{T}}\mathbf{S}_{,1}\right) \\ \mathbf{e}^{\mathsf{T}}\left(\mathbf{S}_{,1}^{\mathsf{T}}\mathbf{S}_{,2}+\mathbf{S}_{,2}^{\mathsf{T}}\mathbf{S}_{,1}\right) \end{bmatrix} = \begin{bmatrix} \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(1)} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(2)} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(3)} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(4)} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(5)} \\ \mathbf{e}^{\mathsf{T}}\mathbf{S}^{(6)} \end{bmatrix}. \tag{8}$$

Applying the material law, the expression for the generalized internal force can be written as

$$\begin{aligned} Q^{Internal} &= -\int_{V_\xi}\left(\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{e}}\right)^{\mathsf{T}}(\mathbf{D}\boldsymbol{\varepsilon}+\tau\mathbf{D}\dot{\boldsymbol{\varepsilon}})\det\left(\mathbf{J}_{0\xi}\right)\,dV_\xi \\ &= -\int_{V_\xi}\left(\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{e}}\right)^{\mathsf{T}}(\mathbf{D})(\boldsymbol{\varepsilon}+\tau\dot{\boldsymbol{\varepsilon}})\det\left(\mathbf{J}_{0\xi}\right)\,dV_\xi. \end{aligned} \tag{9}$$

Examining the definition of these terms, the Green–Lagrange strains and the time derivative of the Green–Lagrange strains can be combined as

$$\boldsymbol{\varepsilon}+\tau\dot{\boldsymbol{\varepsilon}} = \begin{bmatrix} \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(1)}(\mathbf{e}+2\tau\dot{\mathbf{e}})-\frac{1}{2} \\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(2)}(\mathbf{e}+2\tau\dot{\mathbf{e}})-\frac{1}{2} \\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(3)}(\mathbf{e}+2\tau\dot{\mathbf{e}})-\frac{1}{2} \\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(4)}(\mathbf{e}+2\tau\dot{\mathbf{e}}) \\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(5)}(\mathbf{e}+2\tau\dot{\mathbf{e}}) \\ \frac{1}{2}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(6)}(\mathbf{e}+2\tau\dot{\mathbf{e}}) \end{bmatrix}. \tag{10}$$

Using this expression, the generalized internal force can be written as the combination of two summations:

$$\begin{aligned} Q^{Internal} &= -\int_{V_\xi}\left(\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{e}}\right)^{\mathsf{T}}(\mathbf{D})(\boldsymbol{\varepsilon}+\tau\dot{\boldsymbol{\varepsilon}})\det\left(\mathbf{J}_{0\xi}\right)\,dV_\xi \\ &= -\int_{V_\xi}\left(\frac{1}{2}\sum_{t}^{6}\sum_{v}^{6}D_{tv}\mathbf{S}^{(t)}\mathbf{e}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(v)}(\mathbf{e}+2\tau\dot{\mathbf{e}})\right. \\ &\qquad \left.-\frac{1}{2}\sum_{t}^{6}\sum_{v}^{3}D_{tv}\mathbf{S}^{(t)}\mathbf{e}\right)\det\left(\mathbf{J}_{0\xi}\right)\,dV_\xi. \end{aligned} \tag{11}$$

Noting that the nodal coordinates or the combination of nodal coordinates can be pulled out of the integral from the right side, the generalized internal force can simply be written as the sum of two matrix vector products

$$Q^{Internal} = \mathbf{K}_2(\mathbf{e}+2\tau\dot{\mathbf{e}})+\mathbf{K}_1\mathbf{e}. \tag{12}$$

where

$$\mathbf{K}_2 = -\int_{V_\xi}\left(\frac{1}{2}\sum_{t}^{6}\sum_{v}^{6}D_{tv}\mathbf{S}^{(t)}\mathbf{e}\mathbf{e}^{\mathsf{T}}\mathbf{S}^{(v)}\right)\det\left(\mathbf{J}_{0\xi}\right)\,dV_\xi \tag{13}$$

and

$$\mathbf{K}_1 = \int_{V_\xi}\left(\frac{1}{2}\sum_{t}^{6}\sum_{v}^{3}D_{tv}\mathbf{S}^{(t)}\right)\det\left(\mathbf{J}_{0\xi}\right)\,dV_\xi. \tag{14}$$

Examining these matrices, $\mathbf{K}_1$ is constant throughout the simulation and has no dependency on the nodal coordinates. As such, this integral can be computed once prior to the start of the simulation and then reused when needed. However, $\mathbf{K}_2$ has a clear dependency on the nodal coordinates. As shown in [3], it is still possible to pull the nodal coordinates out of the integral of $\mathbf{K}_2$ even though they are in the middle of matrix multiplications on either side. Using tensor notation for the "matrix times a vector times a vector transposed times a matrix" structure of the terms in $\mathbf{K}_2$, it can be shown that each individual entry in $\mathbf{K}_2$ can be written as the product of the nodal coordinates transposed times a new matrix times the nodal coordinates as

$$\left(\mathbf{A}\mathbf{e}\mathbf{e}^{\mathsf{T}}\mathbf{B}\right)_{ij} = A_{ik}e_k e_l B_{lj} = e_k A_{ik}B_{lj}e_l = \mathbf{e}^{\mathsf{T}}\mathbf{C}^{ij}\mathbf{e}, \tag{15}$$

where the matrix can be calculated by

$$\mathbf{C}^{ij} = A_{row\,i}^{\mathsf{T}}B_{col\,j}^{\mathsf{T}}. \tag{16}$$

While this technique results in a large number of sparse "$\mathbf{C}_{\mathbf{K}_2}^{ij}$" matrices, all of these matrices can be integrated prior to the start of the simulation. As a result, the in-simulation generalized internal force evaluation is completely independent of the number of Gauss quadrature points required to integrate across the volume of the element. As discussed during the computer implementation of this method in Appendix B, there are properties that can be leveraged to reduce both the number and size of the constant matrices that need to be stored and used for computations.

When using an implicit integrator, the partial derivative of the generalized internal force with respect to the nodal coordinates and the time derivative of the nodal coordinates is needed. Using Eq. (12), the partial derivative with respect to the nodal coordinates can be written as

$$\frac{\partial Q^{Internal}}{\partial \mathbf{e}} = \frac{\partial \mathbf{K}_2}{\partial \mathbf{e}}(\mathbf{e}+2\tau\dot{\mathbf{e}})+\mathbf{K}_2+\mathbf{K}_1 = \mathbf{K}_3+\mathbf{K}_2+\mathbf{K}_1, \tag{17}$$

where all of the terms except for $\mathbf{K}_3$ are also needed for the generalized internal force evaluation. This final term can be written as

$$\left(\mathbf{K}_3\right)_{ik} = \sum_{j}\sum_{s}\left(e_s\left(C_{\mathbf{K}_2}^{ij}+\left(C_{\mathbf{K}_2}^{ij}\right)^{\mathsf{T}}\right)_{sk}\left(e_j+2\tau\dot{e}_j\right)\right). \tag{18}$$

The partial derivative of the generalized internal force with respect to the time derivative of the nodal coordinates contains no new terms and can simply be written as

$$\frac{\partial Q^{Internal}}{\partial \dot{\mathbf{e}}} = 2\tau\mathbf{K}_2. \tag{19}$$

Appendix B. Implementation details, method "G"

The presentation in [3] focused on the mathematical details of the computational method and did not provide many specific implementation details. The authors did provide several high level observations that they stated could be used to improve the efficiency of their method. First the $\mathbf{K}_2$ matrix is symmetric so only the diagonal and upper or lower triangular terms need to be calculated. Second, the $\mathbf{C}_{\mathbf{K}_2}^{ij}$ matrices are sparse, which both reduces the number of required calculations and the amount of data that needs to be stored.

Focusing on the $\mathbf{C}_{\mathbf{K}_2}^{ij}$ matrices, while there is a sparsity pattern to them, there is a repetitive pattern that can be leveraged as well. This pattern is a direct result of the structure of the shape function matrix used for this method as shown in Eq. (3), where each individual unique

shape function is multiplied by the $[3\times 3]$ identity matrix. The identity matrix carries through to each of the six $\mathbf{S}^{(t)}$ matrices which all have the symmetric structure

$$\mathbf{S}^{(t)} = \begin{bmatrix} a\mathbf{I}_{3\times3} & b\mathbf{I}_{3\times3} & \dots & c\mathbf{I}_{3\times3} \\ b\mathbf{I}_{3\times3} & d\mathbf{I}_{3\times3} & \dots & e\mathbf{I}_{3\times3} \\ \vdots & \vdots & \ddots & \vdots \\ c\mathbf{I}_{3\times3} & e\mathbf{I}_{3\times3} & \dots & f\mathbf{I}_{3\times3} \end{bmatrix}. \tag{20}$$

As a result, each $[3\times 3]$ block of $\mathbf{C}^{ij}_{\mathbf{K}_2}$ matrices contains the same values, just with a different sparsity pattern. For example, if

$$\mathbf{C}^{11}_{\mathbf{K}_2} = \left[\begin{array}{ccc|ccc|c} g & 0 & 0 & h & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \hline k & 0 & 0 & l & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right], \tag{21}$$

then

$$\mathbf{C}^{12}_{\mathbf{K}_2} = \left[\begin{array}{ccc|ccc|c} 0 & g & 0 & 0 & h & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \hline 0 & k & 0 & 0 & l & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right], \tag{22}$$

and

$$\mathbf{C}^{33}_{\mathbf{K}_2} = \left[\begin{array}{ccc|ccc|c} 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & g & 0 & 0 & h & \dots \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & k & 0 & 0 & l & \dots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right], \tag{23}$$

contain the same values as shown, just in different positions. If only the non-zero entries in the pattern are stored, this reduces the size of each $\mathbf{C}^{ij}_{\mathbf{K}_2}$ matrix from $[3N\times 3N]$ to $[N\times N]$. Additionally, this reduces the number of $\mathbf{C}^{ij}_{\mathbf{K}_2}$ matrices that must be stored from $[3N\times 3N]$ to $N(N+1)/2$ also accounting for the symmetric property of $\mathbf{K}_2$.

Since $\mathbf{K}_1$ and the compacted $\mathbf{C}^{mn}_{\mathbf{K}_2}$ matrices are constant throughout the simulation (they have no dependency on the nodal coordinates or the time derivative of the nodal coordinates), they only need to be calculated once prior to the start of the simulation. Matrix $\mathbf{K}_1$ can be calculated from Eq. (14), typically using Gauss quadrature to approximate the integral. The compacted $\mathbf{C}^{mn}_{\mathbf{K}_2}$ matrices can be calculated from Eq. (13) and (16) using a matrix of just the coefficients of the identity matrices for the $\mathbf{S}^{(t)}$ matrices or by using the $\mathbf{S}^{(t)}$ matrices in sparse form and then compacting the $\mathbf{C}^{ij}_{\mathbf{K}_2}$ matrices to $\mathbf{C}^{mn}_{\mathbf{K}_2}$ afterwards.

For the generalized internal force evaluations during the simulation, the compacted $\mathbf{C}^{mn}_{\mathbf{K}_2}$ matrices could be expanded out to their full $[3N\times 3N]$ size and multiplied by the vector of nodal coordinates on each side. However, this would result in a large number of unnecessary multiplications by known zeros. Instead, $\mathbf{K}_2$ can be calculated in $[3\times 3]$ blocks using the $\mathbf{C}^{mn}_{\mathbf{K}_2}$ matrices in compact form. For this process, the nodal coordinates, $\mathbf{e}$, are written in the $[3\times N]$ matrix form presented in [4] as

$$\bar{\mathbf{e}} = \begin{bmatrix} e_1 & e_4 & \dots & e_{n-2} \\ e_2 & e_5 & \dots & e_{n-1} \\ e_3 & e_6 & \dots & e_n \end{bmatrix}. \tag{24}$$

Then the $[3\times 3]$ blocks of $\mathbf{K}_2$ can be calculated as

$$\left(\mathbf{K}_2\right)_{((1:3)+3(m-1))\,((1:3)+3(n-1))} = \bar{\mathbf{e}}\,\mathbf{C}^{mn}_{\mathbf{K}_2}\left(\bar{\mathbf{e}}^{\mathsf{T}}\right). \tag{25}$$

Since only the diagonal and upper or lower triangular compact $\mathbf{C}^{mn}_{\mathbf{K}_2}$ matrices were stored, the transpose of the $[3\times 3]$ blocks can be stored in the corresponding symmetric locations in $\mathbf{K}_2$. After calculating all of the entries in $\mathbf{K}_2$, the generalized internal force vector can be computed using Eq. (12).

When using an implicit integrator, the contributions to the Jacobian matrix from the partial derivatives of the generalized internal force with respect to the nodal coordinates and time derivative of the nodal coordinates need to be combined together with appropriate scaling factors that depend on the specific integrator and time step being used. Using K^K as the scale factor on the partial derivative with respect to the nodal coordinates and K^R on the partial derivative with respect to the time derivative of the nodal coordinates, the combined Jacobian matrix contribution, $\mathbf{J}$ can be written as

$$\begin{aligned}\mathbf{J} &= K^K\left(\mathbf{K}_3+\mathbf{K}_2+\mathbf{K}_1\right)+K^R\left(2\tau\mathbf{K}_2\right)\\ &= K^K\mathbf{K}_3+\left(K^K+2\tau K^R\right)\mathbf{K}_2+K^K\mathbf{K}_1.\end{aligned} \tag{26}$$

While $\mathbf{K}_2$ was calculated during the generalized internal force evaluation and $\mathbf{K}_1$ was calculation prior to the start of the simulation, $\mathbf{K}_3$ needs to be computed during the Jacobian matrix evaluation. Examining the subscripts, this can be partially rewritten using matrix multiplications as

$$\begin{aligned}\left(\mathbf{K}_3\right)_{(1:3)+3(m-1),\,(1:3)+3(k-1)} = \sum_n \Bigg[&\bar{\mathbf{e}}\left(\mathbf{C}^{mn}_{\mathbf{K}_2}\right)_{col\,k}\left(\bar{\mathbf{e}}+2\tau\dot{\bar{\mathbf{e}}}\right)^{\mathsf{T}}_{col\,n}\\ &+\left(\left(\mathbf{C}^{mn}_{\mathbf{K}_2}\right)_{row\,k}\bar{\mathbf{e}}^{\mathsf{T}}\left(\bar{\mathbf{e}}+2\tau\dot{\bar{\mathbf{e}}}\right)_{col\,n}\right)\mathbf{I}_{3\times3}\Bigg].\end{aligned} \tag{27}$$

Since this summation can be implemented using "*for*-loops", the number of required calculations can be reduced by calculating $\left(\bar{\mathbf{e}}+2\tau\dot{\bar{\mathbf{e}}}\right)$ and $\left(\bar{\mathbf{e}}^{\mathsf{T}}\left(\bar{\mathbf{e}}+2\tau\dot{\bar{\mathbf{e}}}\right)\right)$ prior to the loops and selecting the required columns where needed in the summation. Based on this, the following algorithm can be used to calculate the Jacobian contribution from the generalized internal force where only the diagonal and upper triangular compact $\mathbf{C}^{mn}_{\mathbf{K}_2}$ matrices are needed:

$\mathbf{J} \leftarrow \left(K^K+2\tau K^R\right)\mathbf{K}_2+K^K\mathbf{K}_1$
$\mathbf{A} \leftarrow \left(\bar{\mathbf{e}}+2\tau\dot{\bar{\mathbf{e}}}\right)$
$\mathbf{B} \leftarrow \bar{\mathbf{e}}^{\mathsf{T}}A$
for $m = 1$ to N **do**
  for $n = m$ to N **do**
    $\mathbf{G} \leftarrow \mathbf{C}^{mn}_{\mathbf{K}_2}\mathbf{B}_{col\,n}$
    $\mathbf{L} \leftarrow \bar{\mathbf{e}}\mathbf{C}^{mn}_{\mathbf{K}_2}$
    for $k = 1$ to N **do**
      $\mathbf{W} \leftarrow \mathbf{L}_{col\,k}\mathbf{A}^{\mathsf{T}}_{col\,n}$
      $\mathbf{W}_{11} \leftarrow \mathbf{W}_{11}+\mathbf{G}_k$
      $\mathbf{W}_{22} \leftarrow \mathbf{W}_{22}+\mathbf{G}_k$
      $\mathbf{W}_{33} \leftarrow \mathbf{W}_{33}+\mathbf{G}_k$
      $\mathbf{J}_{(1:3)+3(m-1),\,(1:3)+3(k-1)} \leftarrow \mathbf{J}_{(1:3)+3(m-1),\,(1:3)+3(k-1)}+K^K\mathbf{W}$
    end for
    if $m \neq n$ **then**
      $\mathbf{G} \leftarrow \left(\mathbf{C}^{mn}_{\mathbf{K}_2}\right)^{\mathsf{T}}\mathbf{B}_{col\,m}$
      $\mathbf{L} \leftarrow \bar{\mathbf{e}}\left(\mathbf{C}^{mn}_{\mathbf{K}_2}\right)^{\mathsf{T}}$
      for $k = 1$ to N **do**
        $\mathbf{W} \leftarrow \mathbf{L}_{col\,k}\mathbf{A}^{\mathsf{T}}_{col\,m}$
        $\mathbf{W}_{11} \leftarrow \mathbf{W}_{11}+\mathbf{G}_k$
        $\mathbf{W}_{22} \leftarrow \mathbf{W}_{22}+\mathbf{G}_k$
        $\mathbf{W}_{33} \leftarrow \mathbf{W}_{33}+\mathbf{G}_k$
        $\mathbf{J}_{(1:3)+3(n-1),\,(1:3)+3(k-1)} \leftarrow \mathbf{J}_{(1:3)+3(n-1),\,(1:3)+3(k-1)}+K^K\mathbf{W}$
      end for
    end if
  end for
end for

Appendix C. Math behind implementation "L"

While Liu et al. [5] presented their calculation method with a linear elastic material model, their approach can be extended to incorporate the simple linear viscoelastic material model used for the comparisons in this paper. This viscoelastic material model, presented by Zhao et al. [9], only requires a single coefficient to be defined for the damping contribution of the material. A similar presentation to that provided by Liu et al. will be given here using tensor notation. The common case of local nodal coordinates is assumed, but these equations can be modified for global nodal coordinates to account for mesh discontinuities as described in [30]. A detailed description for converting methods "C", "D", and "E" is given in [6]; a similar process can also be used for implementation "L".

Liu et al. started their presentation with the expression for the generalized internal force presented in [4], except written in tensor notation. Using the First Piola–Kirchhoff stress tensor, $\mathbf{P}$, the generalized internal force was written as

$$Q^{Internal}_{3(k-1)+m} = -\int_{V_0} P_{mn}\bar{S}^D_{kn}\,dV_0\,, \tag{28}$$

where m and n range from 1 to 3, k ranges from 1 to N, N is the number of unique shape functions for the element, and $\bar{S}^D$ is the modified matrix of shape function derivatives as discussed in [6] and briefly in Appendix D. The nodal coordinates and their time derivatives can be expanded out of this integral through the following series of steps. For these steps, the subscripts f, k, t, and v range from 1 to N and a, b, c, d, m, and n range from 1 to 3.

First, the first Piola–Kirchhoff stress tensor, $\mathbf{P}$, can be written as the product of the deformation gradient, $\mathbf{F}$, and the second Piola–Kirchhoff stress tensor $\mathbf{S}^{PK2}$.

$$Q^{Internal}_{3(k-1)+m} = -\int_{V_0} F_{ma}S^{PK2}_{an}\bar{S}^{Dp}_{kn}\,dV_0\,. \tag{29}$$

Next the deformation gradient can be written as the product of the nodal coordinate in matrix form, $\bar{\mathbf{e}}$, and the modified matrix of shape function derivatives, $\bar{S}^D$, as

$$Q^{Internal}_{3(k-1)+m} = -\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}S^{PK2}_{an}\bar{S}^D_{kn}\,dV_0\,. \tag{30}$$

The second Piola–Kirchhoff stress tensor can be written in terms of the assumed material law as

$$\begin{aligned} Q^{Internal}_{3(k-1)+m} = -\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}\Big(\mathbf{D}^{4th\,order} : \frac{1}{2}\left(\mathbf{F}^{\mathsf{T}}\mathbf{F} - \mathbf{I}_{3\times3}\right. \\ \left. +\tau\left(\dot{\mathbf{F}}^{\mathsf{T}}\mathbf{F} + \mathbf{F}^{\mathsf{T}}\dot{\mathbf{F}}\right)\right)\Big)_{an}\bar{S}^D_{kn}\,dV_0\,, \end{aligned} \tag{31}$$

where $\mathbf{D}^{4th\,order}$ is the elasticity tensor written in its 4th order tensor format, $\dot{\mathbf{F}}$ is the time derivative of the Deformation Gradient, and $\mathbf{I}_{3\times3}$ is the 3×3 identity matrix. The resulting terms from the double dot product with the elasticity tensor can be written as the summation of four different integrals as

$$\begin{aligned} Q^{Internal}_{3(k-1)+m} = &-\frac{1}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}F_{db}F_{dc}\bar{S}^D_{kn}\,dV_0 \\ &+\frac{1}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}I_{bc}\bar{S}^D_{kn}\,dV_0 \\ &-\frac{\tau}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}\dot{F}_{db}F_{dc}\bar{S}^D_{kn}\,dV_0 \\ &-\frac{\tau}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}F_{db}\dot{F}_{dc}\bar{S}^D_{kn}\,dV_0\,. \end{aligned} \tag{32}$$

Expressing the deformation gradient as the product of $\bar{\mathbf{e}}$ and $\bar{S}^D$ and the time derivative of the deformation gradient as the product of $\dot{\bar{\mathbf{e}}}$ and $\bar{S}^D$, the explicit dependency on the nodal coordinates and their time derivatives can be obtained as

$$\begin{aligned} Q^{Internal}_{3(k-1)+m} = &-\frac{1}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}\bar{e}_{df}\bar{S}^D_{fb}\bar{e}_{dv}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &+\frac{1}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}I_{bc}\bar{S}^D_{kn}\,dV_0 \\ &-\frac{\tau}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}\dot{\bar{e}}_{df}\bar{S}^D_{fb}\bar{e}_{dv}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &-\frac{\tau}{2}\int_{V_0} \bar{e}_{mt}\bar{S}^D_{ta}D_{anbc}\bar{e}_{df}\bar{S}^{Dp}_{fb}\dot{\bar{e}}_{dv}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,. \end{aligned} \tag{33}$$

Since the nodal coordinates and their time derivatives only depend on time and not the position within the element, they can be pulled out of the integrals as

$$\begin{aligned} Q^{Internal}_{3(k-1)+m} = &-\frac{1}{2}\bar{e}_{mt}\bar{e}_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &+\frac{1}{2}\bar{e}_{mt}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{kn}I_{bc}\,dV_0 \\ &-\frac{\tau}{2}\bar{e}_{mt}\dot{\bar{e}}_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &-\frac{\tau}{2}\bar{e}_{mt}\bar{e}_{df}\dot{\bar{e}}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,, \end{aligned} \tag{34}$$

where the fact that the terms in tensor notation can be reordered without affecting the equations was also used.

To simplify this equation, the last listed integral can be rewritten. Since subscripts in tensor notation can be renamed without affecting the results, the subscript f is swapped with subscript v and the subscript b is swapped with subscript c to yield

$$\begin{aligned} &-\frac{\tau}{2}\bar{e}_{mt}\bar{e}_{df}\dot{\bar{e}}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \rightarrow \\ &\qquad -\frac{\tau}{2}\bar{e}_{mt}\bar{e}_{dv}\dot{\bar{e}}_{df}\int_{V_0} D_{ancb}\bar{S}^D_{ta}\bar{S}^D_{vc}\bar{S}^D_{fb}\bar{S}^D_{kn}\,dV_0\,. \end{aligned} \tag{35}$$

Utilizing one of the symmetries of the 4th order elasticity tensor

$$D_{anbc} = D_{ancb} \tag{36}$$

and reordering terms, this integral can be written as

$$\begin{aligned} &-\frac{\tau}{2}\bar{e}_{mt}\bar{e}_{dv}\dot{\bar{e}}_{df}\int_{V_0} D_{ancb}\bar{S}^D_{ta}\bar{S}^D_{vc}\bar{S}^D_{fb}\bar{S}^D_{kn}\,dV_0 \rightarrow \\ &\qquad -\frac{\tau}{2}\bar{e}_{mt}\dot{\bar{e}}_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,. \end{aligned} \tag{37}$$

Since Eq. (37) exactly matches the third integral in Eq. (34), Eq. (34) can be simplified to

$$\begin{aligned} Q^{Internal}_{3(k-1)+m} = &-\frac{1}{2}\bar{e}_{mt}\bar{e}_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &+\frac{1}{2}\bar{e}_{mt}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{kn}I_{bc}\,dV_0 \\ &-\tau\bar{e}_{mt}\dot{\bar{e}}_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,. \end{aligned} \tag{38}$$

This can be further simplified by combining the first and third integrals as

$$\begin{aligned} Q^{Internal}_{3(k-1)+m} = &-\bar{e}_{mt}\left(\frac{1}{2}\bar{\mathbf{e}} + \tau\dot{\bar{\mathbf{e}}}\right)_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &+\frac{1}{2}\bar{e}_{mt}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{kn}I_{bc}\,dV_0\,. \end{aligned} \tag{39}$$

Since there are common terms between the expression for the generalized internal force vector shown in Eq. (39) and its Jacobian matrix, the next step is to take the partial derivatives of Eq. (39) with

respect to the nodal coordinates

$$\begin{aligned}\left(\frac{\partial Q^{Internal}}{\partial \mathbf{e}}\right)_{ij} &= \frac{\partial Q_i^{Internal}}{\partial e_j} = \\ &- \frac{\partial \bar{e}_{mt}}{\partial e_j}\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &- \bar{e}_{mt}\frac{\partial\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}}{\partial e_j}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &- \bar{e}_{mt}\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}\frac{\partial\bar{e}_{dv}}{\partial e_j}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &+ \frac{1}{2}\frac{\partial\bar{e}_{mt}}{\partial e_j}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{kn}I_{bc}\,dV_0\,,\end{aligned} \tag{40}$$

where the subscripts $i = 3(k-1)+m$ and j ranges from 1 to $3N$. Each of these terms can simplified. Starting with the first one,

$$\begin{aligned}\mathbf{K}^{(1)}_{ij} &= -\frac{\partial\bar{e}_{mt}}{\partial e_j}\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &= -\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,,\end{aligned} \tag{41}$$

where for $\mathbf{K}^{(1)}$, $i = 3(k-1)+m$ and $j = 3(t-1)+m$. Grouping terms, this can be expressed as

$$\mathbf{K}^{(1)}_{ij} = \boldsymbol{\Theta}^{(1)}_{(N(t-1)+k)\,(N(f-1)+v)}\boldsymbol{\Pi}^{(1)}_{N(f-1)+v}\,, \tag{42}$$

where

$$\boldsymbol{\Theta}^{(1)}_{N(t-1)+k\,,\,N(f-1)+v} = \int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \tag{43}$$

and

$$\boldsymbol{\Pi}^{(1)}_{N(f-1)+v} = -\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}\bar{e}_{dv}\,. \tag{44}$$

As a result, $\mathbf{K}^{(1)}$ can be generated by first calculating the $\left[N^2\times N^2\right]\left[N^2\times 1\right]$ matrix–vector product $\boldsymbol{\Theta}^{(1)}\boldsymbol{\Pi}^{(1)}$ and then rearranging terms as needed.

Next, the second term can be written as

$$\begin{aligned}\mathbf{K}^{(2A)}_{ij} &= -\bar{e}_{mt}\frac{\partial\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}}{\partial e_j}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &= -\frac{1}{2}\bar{e}_{mt}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,,\end{aligned} \tag{45}$$

where for $\mathbf{K}^{(2A)}$, $i = 3(k-1)+m$ and $j = 3(f-1)+d$.

The third term can be written as

$$\begin{aligned}\mathbf{K}^{(2B)}_{ij} &= -\bar{e}_{mt}\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}\frac{\partial\bar{e}_{dv}}{\partial e_j}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0 \\ &= -\bar{e}_{mt}\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,,\end{aligned} \tag{46}$$

where for $\mathbf{K}^{(2B)}$, $i = 3(k-1)+m$ and $j = 3(v-1)+d$. Within $\mathbf{K}^{(2B)}$, if the subscripts f and v are swapped with each other and the subscripts b and c are swapped with each other, this yields

$$\mathbf{K}^{(2B)}_{ij} = -\bar{e}_{mt}\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{dv}\int_{V_0} D_{ancb}\bar{S}^D_{ta}\bar{S}^D_{vc}\bar{S}^D_{fb}\bar{S}^D_{kn}\,dV_0\,, \tag{47}$$

where now $j = 3(f-1)+d$. Applying the symmetry of the 4th order elasticity tensor shown in Eq. (36) and reordering terms, $\mathbf{K}^{(2B)}$ can be written as

$$\mathbf{K}^{(2B)}_{ij} = -\bar{e}_{mt}\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,. \tag{48}$$

Based on the common terms in Eqs. (45) and (48), these equations can be added together

$$\mathbf{K}^{(2)}_{ij} = \mathbf{K}^{(2A)}_{ij} + \mathbf{K}^{(2B)}_{ij} = -\bar{e}_{mt}\left(\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,. \tag{49}$$

Similar to $\mathbf{K}^{(1)}$, terms within the equation for $\mathbf{K}^{(2)}$ can be grouped as

$$\mathbf{K}^{(2)}_{ij} = \boldsymbol{\Pi}^{(2)}_{3(d-1)+m\,,\,N(t-1)+v}\boldsymbol{\Theta}^{(2)}_{N(t-1)+v\,,\,N(f-1)+k}\,, \tag{50}$$

where

$$\boldsymbol{\Pi}^{(2)}_{3(d-1)+m\,,\,N(t-1)+v} = -\bar{e}_{mt}\left(\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{dv} \tag{51}$$

and

$$\boldsymbol{\Theta}^{(2)}_{N(t-1)+v\,,\,N(f-1)+k} = \int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,. \tag{52}$$

The matrix $\mathbf{K}^{(2)}$ can be calculated by first evaluating product of the $\left[9\times N^2\right]\left[N^2\times N^2\right]$ matrix multiplication $\boldsymbol{\Pi}^{(2)}\boldsymbol{\Theta}^{(2)}$ and then by rearranging terms as needed. It should also be noted that $\boldsymbol{\Theta}^{(1)}$ and $\boldsymbol{\Theta}^{(2)}$ are simply reordered forms of each other.

The fourth and final term can be written as

$$\begin{aligned}\mathbf{K}^{(3)}_{ij} &= \frac{1}{2}\frac{\partial\bar{e}_{mt}}{\partial e_j}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{kn}I_{bc}\,dV_0 \\ &= \frac{1}{2}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{kn}I_{bc}\,dV_0 \\ &= \frac{1}{2}\int_{V_0} D_{anbb}\bar{S}^D_{ta}\bar{S}^D_{kn}\,dV_0\,,\end{aligned} \tag{53}$$

where for $\mathbf{K}^{(3)}$, $i = 3(k-1)+m$ and $j = 3(t-1)+m$.

Since the partial derivative of the generalized internal force shown in Eq. (39) is also needed with respect to the time derivative of the nodal coordinates for use with an implicit integrator. This equation can be written as

$$\begin{aligned}\left(\frac{\partial Q^{Internal}}{\partial \dot{\mathbf{e}}}\right)_{ij} &= \frac{\partial Q_i^{Internal}}{\partial \dot{e}_j} = \mathbf{K}^{(4)}_{ij} = \\ &- \bar{e}_{mt}\frac{\partial\left(\frac{1}{2}\bar{\mathbf{e}}+\tau\dot{\bar{\mathbf{e}}}\right)_{df}}{\partial \dot{e}_j}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,,\end{aligned} \tag{54}$$

where the subscripts $i = 3(k-1)+m$ and j range from 1 to $3N$. It can then be simplified as

$$\mathbf{K}^{(4)}_{ij} = -\tau\bar{e}_{mt}\bar{e}_{dv}\int_{V_0} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,dV_0\,, \tag{55}$$

where $j = 3(f-1)+d$. Grouping terms within the equation yields

$$\mathbf{K}^{(4)}_{ij} = \boldsymbol{\Pi}^{(D)}_{3(d-1)+m\,,\,N(t-1)+v}\boldsymbol{\Theta}^{(2)}_{N(t-1)+v\,,\,N(f-1)+k}\,, \tag{56}$$

where

$$\boldsymbol{\Pi}^{(D)}_{3(d-1)+m\,,\,N(t-1)+v} = -\tau\bar{e}_{mt}\bar{e}_{dv} \tag{57}$$

and $\boldsymbol{\Theta}^{(2)}$ is defined in Eq. (52).

Returning to the expression for the generalized internal force in Eq. (39), the generalized internal force shares several of the same quantities as the Jacobian matrices. Utilizing the common terms, Eq. (39) can simply be written as

$$Q_i^{Internal} = Q^{Internal}_{3(k-1)+m} = \mathbf{K}^{(1)}_{ij}e_j + \mathbf{K}^{(3)}_{ij}e_j = \left(\mathbf{K}^{(1)}+\mathbf{K}^{(3)}\right)_{ij}e_j\,, \tag{58}$$

where $j = 3(t-1)+m$.

Appendix D. Implementation details, method "L"

Based on the equations presented in Appendix C and discussed in [5], three constant matrices can be calculated once prior to the start of the simulation and then reused as needed during the generalized internal force and Jacobian matrix evaluations. Focusing first on $\boldsymbol{\Theta}^{(1)}$ shown in Eq. (43), $\boldsymbol{\Theta}^{(1)}$ is a $\left[N^2\times N^2\right]$ matrix, where N is the number of unique shape functions for the particular ANCF element being implemented. While symbolic integration could be used to evaluate this matrix, number integration using Gauss quadrature is assumed here. The first step is to transform the integral from being computed over the volume of the element in its reference configuration, V_0, to an integral over the straight and normalized element coordinates (ξ,η,ζ) as

$$\boldsymbol{\Theta}^{(1)}_{N(t-1)+k\,,\,N(f-1)+v} = \int_{V_\xi} D_{anbc}\bar{S}^D_{ta}\bar{S}^D_{fb}\bar{S}^D_{vc}\bar{S}^D_{kn}\,\det\left(\mathbf{J}_{0\xi}\right)\,dV_\xi\,, \tag{59}$$

where ξ, η, and ζ each range from -1 to 1 and $\mathbf{J}_{0\xi}$ is the element Jacobian between the reference and normalized volumes. The element Jacobian between the reference and normalized volumes is defined as

$$\mathbf{J}_{0\xi} = \frac{\partial \mathbf{r}_0}{\partial \boldsymbol{\xi}} = \bar{\mathbf{e}}_0 \left(\frac{\partial \bar{\mathbf{S}}(\xi,\eta,\zeta)}{\partial \boldsymbol{\xi}} \right) = \bar{\mathbf{e}}_0 \begin{bmatrix} \frac{\partial S_1}{\partial \xi} & \frac{\partial S_1}{\partial \eta} & \frac{\partial S_1}{\partial \zeta} \\ \frac{\partial S_2}{\partial \xi} & \frac{\partial S_2}{\partial \eta} & \frac{\partial S_2}{\partial \zeta} \\ \frac{\partial S_3}{\partial \xi} & \frac{\partial S_3}{\partial \eta} & \frac{\partial S_3}{\partial \zeta} \\ \vdots & \vdots & \vdots \\ \frac{\partial S_N}{\partial \xi} & \frac{\partial S_N}{\partial \eta} & \frac{\partial S_N}{\partial \zeta} \end{bmatrix} , \tag{60}$$

where $\mathbf{r}_0$ is the position of a material points within the element in the unstressed reference configuration, $\bar{\mathbf{e}}_0$ are the nodal coordinates defining the reference configuration written in the $[3 \times N]$ matrix form presented in [4], and $S_1 \dots S_N$ are the unique shape function for the particular element being implemented. Next, $\boldsymbol{\Theta}^{(1)}$ can be approximated by Gauss quadrature as

$$\Theta^{(1)}_{N(t-1)+k\,,\,N(f-1)+v} \approxeq \sum_{N_Q} W \left(D_{anbc} \bar{S}^D_{ta} \bar{S}^D_{fb} \bar{S}^D_{vc} \bar{S}^D_{kn} \det\left(\mathbf{J}_{0\xi}\right) \right) , \tag{61}$$

where the Gauss quadrature weight, W, and all of the other terms are calculated at each specific Gauss quadrature point. Using the definition of the modified matrix of shape function derivatives,

$$\bar{\mathbf{S}}^D = \left(\frac{\partial \bar{\mathbf{S}}(\xi,\eta,\zeta)}{\partial \boldsymbol{\xi}} \right) \mathbf{J}_{0\xi}{}^{-1} , \tag{62}$$

$\boldsymbol{\Theta}^{(1)}$ can then be calculated by looping over all of the subscripts and Gauss quadrature points.

When examining the expression for $\boldsymbol{\Theta}^{(2)}$ given in Eq. (52), it can be seen that it is simply a reordered form of the terms in $\boldsymbol{\Theta}^{(1)}$. As such, it can easily be calculated using the following algorithm where $[N \times N]$ blocks of $\boldsymbol{\Theta}^{(1)}$ are transposed and placed into the correct locations in $\boldsymbol{\Theta}^{(2)}$:

for f = 1 to N **do**
  for t = 1 to N **do**
    $\boldsymbol{\Theta}^{(2)}_{N(t-1)+(1:N)\,,\,N(f-1)+(1:N)} \leftarrow \left(\boldsymbol{\Theta}^{(1)}_{N(t-1)+(1:N)\,,\,N(f-1)+(1:N)} \right)^{\mathsf{T}}$
  end for
end for

Moving to the third matrix, $\mathbf{K}^{(3)}$, it can be seen that there is a regular sparsity pattern within this entire $[3N \times 3N]$ matrix due to the subscript m and that only a $[N \times N]$ matrix actually needs to be computed. As shown later, this compact $[N \times N]$ form of $\mathbf{K}^{(3)}$, denoted as $\mathbf{K}^{(3Compact)}$, can be directly leveraged in the generalized internal force calculations. Therefore, it only needs to be computed and stored in this form. Additionally, noting the symmetry $D_{anbc} = D_{nabc}$ in the fourth order elasticity tensor, it can also be shown that $\mathbf{K}^{(3)}$ is symmetric. Leveraging these characteristic and switching to Gauss quadrature over the normalized element volume, $\mathbf{K}^{(3Compact)}$ can be calculated as

$$\mathbf{K}^{(3Compact)} = \frac{1}{2} \sum_{N_Q} \sum_{b}^{3} W \left(\bar{\mathbf{S}}^D \mathbf{D}_{anbb} \left(\bar{\mathbf{S}}^D \right)^{\mathsf{T}} \right) \det\left(\mathbf{J}_{0\xi}\right) , \tag{63}$$

where all of the terms are calculated for each specific Gauss quadrature point and $\mathbf{D}_{anbb}$ are $[3 \times 3]$ matrix slices of the fourth order elasticity tensor with the last two subscripts set to the value of b.

While $\boldsymbol{\Theta}^{(1)}$, $\boldsymbol{\Theta}^{(2)}$, and $\mathbf{K}^{(3Compact)}$ can be calculated once prior to the start of the simulation, the remainder of the steps required to calculate the generalized internal force and its Jacobian matrix must be performed during each evaluation within the simulation. As the first in-simulation step for generalized internal force evaluation, the vector $\boldsymbol{\Pi}^{(1)}$ can be calculated as the reshaped matrix product

$$\boldsymbol{\Pi}^{(1)} = -Reshape \left(\left(\frac{1}{2} \bar{\mathbf{e}} + \tau \dot{\bar{\mathbf{e}}} \right)^{\mathsf{T}} \bar{\mathbf{e}} \right) , \tag{64}$$

where the $Reshape\,(\mathbf{X})$ function defined in [6] is used that stacks the transpose of the rows of the argument matrix on top of each other in order to form a column vector. If row major memory storage is used, the $Reshape\,(\mathbf{X})$ function is simply a reinterpretation of the data in memory rather than a true manipulation of the data. Since $\mathbf{K}^{(1)}$ has the same sparsity and repetitive pattern as $\mathbf{K}^{(3)}$, it can be calculated in a compact $[N \times N]$ form using the inverse of the $Reshape\,()$ operator on the simple matrix product

$$Reshape\left(\mathbf{K}^{(1Compact)}\right) = \boldsymbol{\Theta}^{(1)} \boldsymbol{\Pi}^{(1)} . \tag{65}$$

Next $\mathbf{K}^{(1Compact)}$ and $\mathbf{K}^{(3Compact)}$ are combined as

$$\mathbf{K}^{(13Compact)} = \mathbf{K}^{(1Compact)} + \mathbf{K}^{(3Compact)} , \tag{66}$$

which is also stored for later reuse in the Jacobian calculation. Finally, the generalized internal force can be calculated as

$$\mathbf{Q}^{Internal} = Reshape\left(\mathbf{K}^{(13Compact)} \bar{\mathbf{e}}^{\mathsf{T}}\right) . \tag{67}$$

When using an implicit integrator, the Jacobian of the generalized internal force also needs to be calculated. Since a viscoelastic material law has been assumed, the partial derivative of the generalized internal force with respect to both the nodal coordinates and their time derivative needs to be computed and combined into the total Jacobian contribution, $\mathbf{J}$, from the generalized internal force. Using the scaling factors K^K and K^R that depend on the implicit integration scheme used and the size of the current time step, the total Jacobian contribution can be written as

$$\begin{aligned} \mathbf{J} &= K^K \left(\frac{\partial Q^{Internal}}{\partial \mathbf{e}} \right) + K^R \left(\frac{\partial Q^{Internal}}{\partial \dot{\mathbf{e}}} \right) \\ &= K^K \left(\mathbf{K}^{(1)} + \mathbf{K}^{(2)} + \mathbf{K}^{(3)} \right) + K^R \mathbf{K}^{(4)} . \end{aligned} \tag{68}$$

Focusing first on $K^K \mathbf{K}^{(2)} + K^R \mathbf{K}^{(4)}$, this can simply be written as

$$\left(K^K \mathbf{K}^{(2)} + K^R \mathbf{K}^{(4)} \right)_{ij} = \left(K^K \boldsymbol{\Pi}^{(2)} + K^R \boldsymbol{\Pi}^{(D)} \right)_{ab} \Theta^{(2)}_{bc} = \Pi^{(B)}_{ab} \Theta^{(2)}_{bc} , \tag{69}$$

where $i = 3(k-1)+m$, $j = 3(f-1)+d$, $a = 3(d-1)+m$, $b = N(t-1)+v$, and $c = N(f-1)+k$. While $\boldsymbol{\Theta}^{(2)}$ was calculated prior to the start of the simulation,

$$\Pi^{(B)}_{ab} = -\bar{e}_{mt} \left(\left(K^K + K^R \tau \right) \bar{\mathbf{e}} + K^K \tau \dot{\bar{\mathbf{e}}} \right)_{dv} = -\bar{e}_{mt} \bar{\mathbf{e}}^C_{dv} \tag{70}$$

needs to be calculated during every Jacobian evaluation. The matrix $\boldsymbol{\Pi}^{(B)}$ can be assembled in blocks from a series of matrices that are generated from outer products as

$$\boldsymbol{\Pi}^{(B)} = - \begin{bmatrix} \bar{\mathbf{e}}_{col1} \bar{\mathbf{e}}^C_{row1} & \bar{\mathbf{e}}_{col2} \bar{\mathbf{e}}^C_{row1} & \cdots & \bar{\mathbf{e}}_{colN} \bar{\mathbf{e}}^C_{row1} \\ \bar{\mathbf{e}}_{col1} \bar{\mathbf{e}}^C_{row2} & \bar{\mathbf{e}}_{col2} \bar{\mathbf{e}}^C_{row2} & \cdots & \bar{\mathbf{e}}_{colN} \bar{\mathbf{e}}^C_{row2} \\ \bar{\mathbf{e}}_{col1} \bar{\mathbf{e}}^C_{row3} & \bar{\mathbf{e}}_{col2} \bar{\mathbf{e}}^C_{row3} & \cdots & \bar{\mathbf{e}}_{colN} \bar{\mathbf{e}}^C_{row3} \end{bmatrix} . \tag{71}$$

Using $\boldsymbol{\Pi}^{(B)}$, the product $\left(\boldsymbol{\Pi}^{(B)} \boldsymbol{\Theta}^{(2)} \right)_{ac}$ can then be calculated. Examining subscripts, the $\left[9 \times N^2\right]$ result of this product needs to be reordered into the correct $[3N \times 3N]$ form for the Jacobian matrix, $\mathbf{J}$. This step, as well as combining in the remaining terms calculated during the generalized internal force evaluation, can be performed by the following algorithm to generate the complete Jacobian matrix, $\mathbf{J}$:

for f = 1 to N **do**
  for k = 1 to N **do**
    $\mathbf{J}_{3(k-1)+(1:3)\,,\,3(f-1)+1} \leftarrow \left(\boldsymbol{\Pi}^{(B)} \boldsymbol{\Theta}^{(2)} \right)_{(1:3)\,,\,N(f-1)+k}$
    $\mathbf{J}_{3(k-1)+(1:3)\,,\,3(f-1)+2} \leftarrow \left(\boldsymbol{\Pi}^{(B)} \boldsymbol{\Theta}^{(2)} \right)_{(4:6)\,,\,N(f-1)+k}$
    $\mathbf{J}_{3(k-1)+(1:3)\,,\,3(f-1)+3} \leftarrow \left(\boldsymbol{\Pi}^{(B)} \boldsymbol{\Theta}^{(2)} \right)_{(7:9)\,,\,N(f-1)+k}$
  end for
end for
for w = 1 to N **do**

for $v = 1$ to N **do**
$\mathbf{J}_{(3w+1)\ (3v+1)} \leftarrow \mathbf{J}_{(3w+1)\ (3v+1)} + K^K K_{wv}^{(13Compact)}$
$\mathbf{J}_{(3w+2)\ (3v+2)} \leftarrow \mathbf{J}_{(3w+2)\ (3v+2)} + K^K K_{wv}^{(13Compact)}$
$\mathbf{J}_{(3w+3)\ (3v+3)} \leftarrow \mathbf{J}_{(3w+3)\ (3v+3)} + K^K K_{wv}^{(13Compact)}$
end for
end for

References

[1] A.A. Shabana, R.Y. Yakoub, Three dimensional absolute nodal coordinate formulation for beam elements: Implementation and applications, ASME J. Mech. Des. 123 (2001) 614–621.

[2] A.A. Shabana, R.Y. Yakoub, Three dimensional absolute nodal coordinate formulation for beam elements: Theory, ASME J. Mech. Des. 123 (2001) 606–613.

[3] D. García-Vallejo, J. Mayo, J.L. Escalona, J. Domínguez, Efficient evaluation of the elastic forces and the Jacobian in the absolute nodal coordinate formulation, Nonlinear Dynam. 35 (4) (2004) 313–329.

[4] J. Gerstmayr, A.A. Shabana, Efficient integration of the elastic forces and thin three-dimensional beam elements in the absolute nodal coordinate formulation, in: Proceeding of Multibody Dynamics ECCOMAS Thematic Conference, Madrid, Spain, 2005.

[5] C. Liu, Q. Tian, H. Hu, Dynamics of a large scale rigid–flexible multibody system composed of composite laminated plates, Multibody Syst. Dyn. 26 (3) (2011) 283–305.

[6] M. Taylor, R. Serban, D. Negrut, Implementation implications on the performance of ANCF simulations, Int. J. Nonlinear Mech. (2022) 104328.

[7] K. Nachbagauer, P. Gruber, J. Gerstmayr, Structural and continuum mechanics approaches for a 3D shear deformable ANCF beam finite element: Application to static and linearized dynamic examples, J. Comput. Nonlinear Dynam. 8 (2) (2012) 021004.

[8] D. García-Vallejo, J. Valverde, J. Domínguez, An internal damping model for the absolute nodal coordinate formulation, Nonlinear Dynam. 42 (4) (2005) 347–369.

[9] C. Zhao, H. Yu, Z. Lin, Y. Zhao, Dynamic model and behavior of viscoelastic beam based on the absolute nodal coordinate formulation, Proc. Inst. Mech. Eng. K: J. Multi-Body Dyn. 229 (1) (2015) 84–96.

[10] S. Williams, A. Waterman, D. Patterson, Roofline: an insightful visual performance model for multicore architectures, Commun. ACM 52 (4) (2009) 65–76.

[11] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, D. Negrut, Chrono: An open source multi-physics dynamics engine, in: T. Kozubek (Ed.), High Performance Computing in Science and Engineering – Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 19–49.

[12] Project Chrono, Chrono: An open source framework for the physics-based simulation of dynamic systems, 2020, http://projectchrono.org, Accessed: 2020-03-03.

[13] O. Dmitrochenko, A. Mikkola, Digital nomenclature code for topology and kinematics of finite elements based on the absolute nodal co-ordinate formulation, Proc. Inst. Mech. Eng. K: J. Multi-Body Dyn. 225 (1) (2011) 34–51.

[14] A.M. Mikkola, A.A. Shabana, A non-incremental finite element procedure for the analysis of large deformation of plates and shells in mechanical system applications, Multibody Syst. Dyn. 9 (3) (2003) 283–309.

[15] A. Olshevskiy, O. Dmitrochenko, C.-W. Kim, Three-dimensional solid brick element using slopes in the absolute nodal coordinate formulation, J. Comput. Nonlinear Dyn. 9 (2) (2014) 021001.

[16] H. Ebel, M.K. Matikainen, V.-V. Hurskainen, A. Mikkola, Analysis of high-order quadrilateral plate elements based on the absolute nodal coordinate formulation for three-dimensional elasticity, Adv. Mech. Eng. 9 (6) (2017) 1687814017705069.

[17] H. Ebel, M.K. Matikainen, V.-V. Hurskainen, A. Mikkola, Higher-order beam elements based on the absolute nodal coordinate formulation for three-dimensional elasticity, Nonlinear Dynam. 88 (2) (2017) 1075–1091.

[18] G. Orzechowski, Analysis of beam elements of circular cross section using the absolute nodal coordinate formulation, Archive of Mechanical Engineering 59 (3) (2012).

[19] G. Orzechowski, A.A. Shabana, Analysis of warping deformation modes using higher order ANCF beam element, J. Sound Vib. 363 (2016) 428–445.

[20] E. Dowell, J.J. Traybar, An Addendum to an Experimental Study of the Nonlinear Stiffness of a Rotor Blade Undergoing Flap, Lag and Twist Deformations, Technical Repor NASA-CR-137969 - AMS Report No 1257, Princeton University, 1975.

[21] K. Sze, X. Liu, S. Lo, Popular benchmark problems for geometric nonlinear analysis of shells, Finite Elem. Anal. Des. 40 (11) (2004) 1551–1569.

[22] S. Huss-Lederman, E.M. Jacobson, J.R. Johnson, A. Tsao, T. Turnbull, Implementation of Strassen's algorithm for matrix multiplication, in: Supercomputing'96: Proceedings of the 1996 ACM/IEEE Conference on Supercomputing, IEEE, 1996, p. 32.

[23] R.E. Bryant, O. David Richard, Computer Systems: A Programmer's Perspective, third ed., Prentice Hall Upper Saddle River, 2015.

[24] M. Taylor, Supporting ANCF code and models, 2022, https://github.com/taylome/chrono/tree/thesis/linear_elastics.

[25] OpenMP, Specification Standard 5.2, 2021, Available online at http://openmp.org/.

[26] J. Gerstmayr, A. Shabana, Analysis of thin beams and cables using the absolute nodal co-ordinate formulation, Nonlinear Dynam. 45 (1) (2006) 109–130.

[27] A. Ilic, F. Pratas, L. Sousa, Cache-aware roofline model: Upgrading the loft, IEEE Comput. Archit. Lett. 13 (1) (2013) 21–24.

[28] D. Negrut, R. Rampalli, G. Ottarsson, A. Sajdak, On the use of the HHT method in the context of index 3 differential algebraic equations of multibody dynamics, ASME JCND 2 (2007).

[29] G. Guennebaud, B. Jacob, Eigen v3, 2010, http://eigen.tuxfamily.org.

[30] A.A. Shabana, A.M. Mikkola, Use of the finite element absolute nodal coordinate formulation in modeling slope discontinuity, J. Mech. Des. 125 (2) (2003) 342–350.