COMPUTATIONAL MECHANICS 2023
38th conference with international participation
Srní
October 23 - 25, 2023

Comparison of co-rotational and other algorithms for geometrically nonlinear static analysis with small strains

R. Páleník[a,b], Z. Poruba[b]

[a] IT4Innovations, VSB – Technical University of Ostrava, 17. listopadu 2172/15, 708 00, Ostrava-Poruba, Czech Republic
[b] Department of Applied Mechanics, Faculty of Mechanical Engineering, VSB – Technical University of Ostrava, 17. listopadu 2172/15, 708 00, Ostrava-Poruba, Czech Republic

1. Introduction

Geometrically nonlinear problems of structural mechanics are typically solved by a fully nonlinear (NL) algorithm utilizing the total or updated Lagrangian formulation. However, in cases where the large displacements result mainly from the rigid body motion and induce only small strains, using the co-rotational (CR) algorithm is advantageous. Such an algorithm utilizes the CR formulation which yields a faster pseudolinear solution compared to its fully NL counterpart.

This paper aims to demonstrate the capabilities of the implemented CR algorithm on a practical example of an orthopaedic shoe insole. However, the purpose is not the strength analysis of the insole.

The previously published CR algorithm [3] is extended by incorporating the tetrahedral element, and strain and stress post-processing. Firstly, the CR and NL algorithm are theoretically compared, as they were implemented in MATLAB. Secondly, both the implemented algorithms are practically tested in a static analysis of the orthopaedic shoe insole discretized with tetrahedral elements. The resulting displacements, strains, and stresses are verified with finite element commercial software results.

2. Description of the CR and NL algorithm

This section provides a comparative theoretical description of two algorithms utilizing the finite element method to solve geometrically nonlinear static analysis with small strains. The CR algorithm implemented according to [2] utilizes the CR formulation, whereas the NL algorithm implemented according to Chapter 9 in [1] utilizes the total Lagrangian formulation. Both the implemented algorithms are applicable to any solid finite element with linear shape functions. Also, when coupled with a proper time-integration scheme, the algorithms can be used for a transient dynamic analysis. A linear elastic material model is assumed.

The main steps of a static analysis with the NL and CR algorithm are presented in Table 1. Both algorithms can be divided into two parts. Firstly, the nodal displacements $\mathbf{u}_{\mathrm{new}}$ of the whole model are calculated using the conventional iterative Newton-Raphson (NR) procedure. Secondly, the element strains and stresses are computed from the resulting displacements on a per-element basis. Throughout the paper, the e subscript denotes the e-th element of the mesh.

The exceptionality and computational efficiency of the CR algorithm come from the utilization of the deformational nodal displacements $\boldsymbol{u}_{ed}$, whereas the NL algorithm uses only the standard (i.e., total) nodal displacements $\boldsymbol{u}_e$ on the element level. The deformational displacements are that part of displacements which causes strain, and they are extracted via the

polar decomposition of the deformational gradient computed in the element centroid. In the case of geometrically nonlinear problems with small strains, the large displacements are mainly caused by the rigid body motions. Therefore, the extracted deformational displacements are small, which enables linear calculation on the element level.

In the case of the NL algorithm, only the constitutional matrix $\mathbf{C}$ can be precomputed. Then, in each load step and each NR iteration inside of it, the NL formulation computes the element tangent stiffness matrix $\boldsymbol{K}_{te}$ and the element internal force vector $\boldsymbol{f}_{ie}$ using the Gauss quadrature integration. The integration procedure starts from the known constitutional matrix, initial element coordinates $\boldsymbol{x}_{0e}$ and the current nodal displacements $\boldsymbol{u}_e$, and it utilizes the deformation gradient, the Green-Lagrange strain tensor, and the second Piola-Kirchhoff stress tensor.

Whereas, in the case of the CR algorithm, the linear element stiffness matrix $\boldsymbol{K}_e$ can be precomputed for each element. Then, during the NR iterations, they are just modified inside the CR formulation onto $\boldsymbol{K}_{te}$. The modification contains mainly a both-sided multiplication of $\boldsymbol{K}_e$ with an orthogonal rotation matrix. The rotation matrix is obtained from the polar decomposition of the deformational gradient computed at the element centroid, and it represents the rigid body rotation of the element with respect to its initial coordinates.

Table 1. Main steps of a static analysis with the NL and CR algorithm

NL algorithm	**CR algorithm**
Prepare constitutional matrix $\boldsymbol{C}$	Compute linear element stiffness matrices $\boldsymbol{K}_e$
Start load steps loop, inside of it start NR iterations	
$(\boldsymbol{C}, \boldsymbol{x}_{0e}, \boldsymbol{u}_e) \xrightarrow{\text{NL formulation}} (\boldsymbol{K}_{te}, \boldsymbol{f}_{ie})$	$(\boldsymbol{K}_e, \boldsymbol{x}_{0e}, \boldsymbol{u}_e) \xrightarrow{\text{CR formulation}} (\boldsymbol{K}_{te}, \boldsymbol{f}_{ie}, \boldsymbol{u}_{ed})$
Assemble global tangent stiffness matrix $\boldsymbol{K}_t$ from $\mathbf{K}_{\mathrm{te}}$ and global internal forces $\boldsymbol{f}_{int}$ from $\boldsymbol{f}_{ie}$ and apply boundary conditions	
Compute vector of residual forces $\boldsymbol{r} = \boldsymbol{f}_{ext} - \boldsymbol{f}_{int}$	
NR iterations of displacements: $\boldsymbol{u}_{new} = \boldsymbol{u}_{old} + \boldsymbol{K}_t^{-1}\boldsymbol{r}$	
Exit NR iterations if $\|\boldsymbol{r}\|/\|\boldsymbol{f}_{ext}\|$ is smaller than predefined NR tolerance	
Increase external forces $\boldsymbol{f}_{ext}$ up to full load and then exit load steps loop	
Extract element displacements $\boldsymbol{u}_e$ from $\boldsymbol{u}_{new}$	Use element deformational displacements $\boldsymbol{u}_{ed}$
Compute deformational gradient: $\boldsymbol{F}_e = \frac{\partial(\boldsymbol{x}_{0e}+\boldsymbol{u}_e)}{\partial \boldsymbol{x}_{0e}}$	Compute linear strain-displacement matrix: $\boldsymbol{B}_e$
Compute Green-Lagrange strain tensor: $\boldsymbol{E}_e = 1/2\left(\boldsymbol{F}_e{}^T\boldsymbol{F}_e - \boldsymbol{I}\right)$, rewrite into vector $\boldsymbol{\varepsilon}_e$	Compute infinitesimal strain vector: $\boldsymbol{\varepsilon}_e = \boldsymbol{B}_e\,\boldsymbol{u}_{ed}$
Compute element stress vector: $\boldsymbol{\sigma}_e = \boldsymbol{C}\,\boldsymbol{\varepsilon}_e$	

In the second part of the NL algorithm, firstly, the element Green-Lagrange strain tensor $\boldsymbol{E}_e$ is computed from the element deformational gradient $\boldsymbol{F}_e$. Then, the element strain tensor components are rewritten to a vector form $\boldsymbol{\varepsilon}_e$ using the Voigt notation. The identity matrix $\boldsymbol{I}$ has dimensions 3×3 for 3-dimensional solid elements or 2×2 for 2-dimensional solid elements.

On the other hand, the CR algorithm is able to compute directly the element infinitesimal strain tensor components in a vector form $\boldsymbol{\varepsilon}_e$ linearly from the known deformational displacements $\boldsymbol{u}_{ed}$. The linear strain-displacement matrix $\boldsymbol{B}_e$ contains derivatives of shape functions with respect to the initial nodal coordinates.

Once the element strain tensor components are known, the element stress tensor components $\boldsymbol{\sigma}_e$, with respect to the initial configuration, are obtained easily by multiplying with $\boldsymbol{C}$. This, of course, applies only to the linear elastic material model.

Both the implemented algorithms compute its derivative quantity, $\boldsymbol{F}_e$ or $\boldsymbol{B}_e$, only in the element centroid, not in Gauss points. This is precise only for tetrahedral or triangular solid finite elements with linear shape functions because they have constant strain distribution through the whole element. However, based on the author's tests with hexahedral elements, this procedure gives satisfactorily precise results for other solid elements with linear shape functions, if a model contains relatively many small elements.

The equivalent strain and stress and all the other derived quantities on the element level are standardly computed from the element strain or stress tensor components. Optionally, the element strain and stress quantities can be averaged in nodes.

3. Numerical tests on an orthopaedic shoe insole undergoing large displacements

The orthopaedic shoe insole is discretized by 39595 tetrahedral elements with linear shape functions. The computational model contains 8896 nodes, and it is depicted in Fig. 1. The nodes marked by blue colour are fixed and 21 nodes marked by the red arrows are loaded in the direction of the z-axis. The loading force of 20 N is evenly distributed onto the loaded nodes. A linear elastic material model with Young's modulus of 1250 MPa and Poisson's ratio of 0.29 is assumed in the whole range of strains.

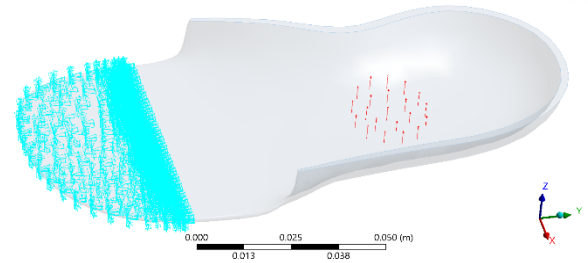

Fig. 1. Computational model: fixed nodes (blue), forced nodes (red)

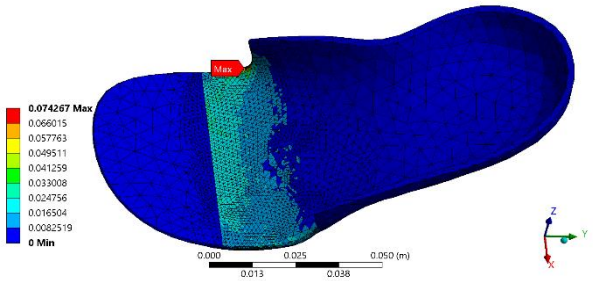

Fig. 2. Equivalent element strain (-) from commercial software

The geometrically nonlinear static analysis of the insole was performed using 3 different algorithms: the implemented CR, the implemented NL, and the one from commercial software which utilizes the updated Lagrangian formulation. The resulting maximal total displacements, maximal equivalent element strains and maximal equivalent element stresses are comparatively presented in Table 2, where the commercial software results are taken as a reference. The CR and NL algorithm results are almost identical and differ from the commercial software results by less than 2 %. To avoid the influence of the different nodal averaging techniques, the element (i.e., unaveraged) strains and stresses are compared. These are the results which directly come from the constant-strain tetrahedral finite elements. However, they are significantly higher than the nodal (i.e., averaged) results which are typically used for the strength evaluation.

Both the implemented algorithms required only one load step to reach the specified NR tolerance of 10^{-7}. Although, the NL algorithm used fewer NR iterations than the CR one, computation with the CR algorithm was about 1.2 times faster, because the CR formulation calculations are computationally less demanding. In terms of computational time, the algorithms implemented in MATLAB cannot compete with the optimized implementation in commercial software.

In Figs. 2 and 3, the distributions of equivalent element strains are displayed on a deflected insole in a true scale. The distributions obtained by different algorithms are visually indistinguishable. Although the maximal equivalent strain is almost 8 %, such high strains

appear only locally on a few elements. In most elements, the equivalent strain is lower than 3 %, which can be considered a small strain. Therefore, the CR algorithm provides very similar results to the NL one.

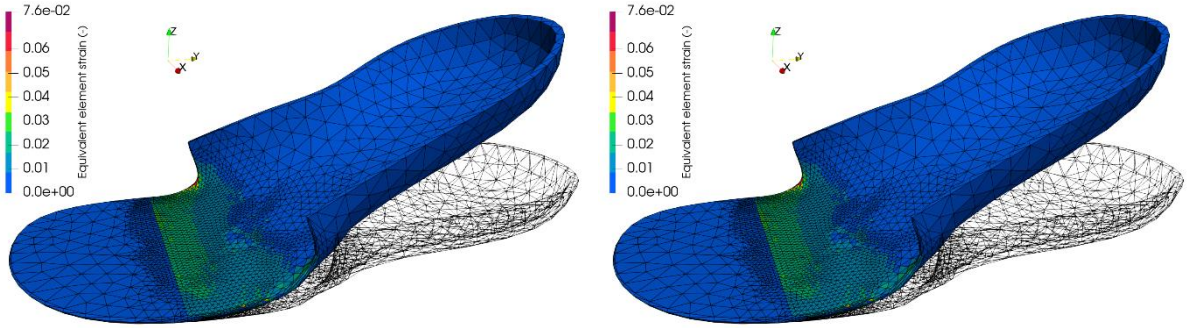

Fig. 3. Equivalent element strain (-) by NL (left) and CR (right) algorithm

Table 2. Comparison of the CR and NL algorithm results and verification with commercial software results

Algorithm	**NL**	**CR**	**Commercial**
Max. total displacement [m]	0.043170	0.043135	0.042643
Displacement relative error [%]	1.24	1.15	-
Max. eqv. element strain [-]	0.075667	0.075170	0.074267
Eqv. strain relative error [%]	1.89	1.22	-
Max. eqv. element stress [MPa]	94.583	93.963	92.834
Eqv. stress relative error [%]	1.88	1.22	-
Computational time [s]	1748	1475	(11)
Number of NR iterations [-]	9	10	(7)

4. Conclusion

The CR and NL algorithm were implemented and numerically tested on a practical example of an orthopaedic shoe insole undergoing large displacements. The insole was discretized by tetrahedral finite elements with linear shape functions. The resulting CR and NL displacements, element strains and stresses are almost the same, and they differ from the commercial software results by less than 2 %. The CR algorithm is faster than the NL one because it contains less computationally demanding mathematical operations. Although the CR formulation is generally limited by a small strain, it was demonstrated that the large local strain over a few elements does not noticeably damage the results, because the large local strains do not influence the final deflected shape significantly.

Acknowledgements

The presented work was supported by: the Specific Reasearch „Application of Modern Computational and Experimental Approaches in Applied Mechanics“ (SP2023/027); the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254); the science and research support in the Moravian-Silesian region (RRC/12/2022).

References

[1] Bhatti, M. A., Advanced topics in finite element analysis of structures with Mathematica and MATLAB computations. Wiley, New York, 2006.
[2] Moita, G. F., Crisfield, M. A., A finite element formulation for 3-D continua using the co-rotational technique, International journal for numerical methods in engineering 39 (1996) 3775-3792.
[3] Páleník, R., Molčan, M., Poruba, Z., Comparison of co-rotational and nonlinear finite element simulations with geometric nonlinearities, Proceedings of the 27/28th International Conference Engineering Mechanics, Milovy, 2022, pp. 301-304.