

## Automation of Computational Modeling by Automatic Differentiation

J. Korelc

*University of Ljubljana, Faculty of Civil and Geodetic Engineering, Slovenia, jkorelc@fgg.uni-lj.si*

---

The advances in reliability, generality and interdisciplinary nature of the new computational methods derived in recent years are primary result of a holistic approach to computational modeling. The use of advanced software technologies is playing a central role in the process that leads to the ultimate goal, i.e. a complete automation of computational modeling. The problem of automation of computational methods has been explored by researches from the fields of mathematics, computer science and computational mechanics, resulting in a variety of approaches (e.g. a hybrid object-oriented approach [1] and a hybrid symbolic-numeric approach [5]) and available software tools (e.g. computer algebra systems, automatic differentiation tools [2], problem solving environments and numerical libraries). Automation can address all steps of the finite element solution procedure from the strong form of boundary-value problem to the presentation of results [6], but more often it is used only for the automation of the selected steps of the whole procedure. The paper presents a hybrid symbolic-numeric approach to automation of primal as well as sensitivity analysis [3]. The hybrid symbolic-numeric approach employs general-purpose automatic code generator [7] to derive and code characteristic finite element quantities (e.g. residual vector and stiffness matrix) at the level of individual finite element and a general-purpose finite element environment to solve the global problem. The automatic code generation approach presented combines a symbolic system Mathematica, an automatic differentiation technique with the simultaneous expression optimisation and an automatic generation of program code in a selected compiled language.

The automation of the finite element methods should not be restricted only to the repetition of the same procedures that are normally done manually on a sheet of paper. The true advantages of automation become apparent only if the description of the problem, the notation and the mathematical apparatus used are changed as well. It is demonstrated in the paper that this can be achieved using the automatic differentiation technique. The automatic differentiation technique is based on the fact that every computer program executes a sequence of elementary operations with known derivatives, thus allowing the evaluation of exact derivatives via the chain rule for an arbitrary complex formulation [2]. Therefore, the automatic differentiation is a method to evaluate the derivative of a function specified by a computer program and represents an alternative to the classical manual derivation of derivatives. Recent advances in development of automatic differentiation technique, especially the backward mode implementation of the code-to-code approach to automatic differentiation, have rejected the traditional assumption that automatic differentiation is impracticable and that the automatic differentiation based numerical codes are intrinsically too slow for large-scale numerical computations. However, as powerful as automatic differentiation technique is, the results of the automatic differentiation procedure might not automatically correspond to the specific mathematical formalism used to describe the mechanical problem. The essential feature of the proposed approach is that it extends the classical formulation of automatic differentiation technique by additional operators defining exceptions in automatic differentiation procedure. Based on these operators a new notation is introduced, representing a bridge between the classical mathematical notation of computational models and the actual algorithmic implementation of finite elements. Thus, the basis for the proposed automation of computational modeling is an automatic differentiation based form or ADB form of basic equations used to describe the problem. The introduced notation does not only simplify the derivation of the corresponding equations, but also reflects much more closely the actual algorithmic

implementation. In this way, the mathematical formulation and computer implementation become indistinguishable.

The automatically generated code is numerically efficient if the number of functions to differentiate and the number of calls to automatic differentiation procedure are kept to a minimum. One of the consequences of this rule is that, in general, formulations where the element residual vector is derived as a gradient of scalar function, e.g. variational potential, lead to a more efficient numerical code than those based on weak form of the equilibrium equations where the variation of the kinematic quantities, e.g. strain tensor or tangential gap vector, requires differentiation of several scalar functions. Thus, the possibility of transforming the weak form into the “pseudo-potential” scalar function is worth exploring. The pseudo-potential has to be formed in a way that automatic differentiation of the pseudo-potential accompanied with the proper automatic differentiation exceptions leads to the correct equations of the problem.

The paper demonstrates on several examples direct consequences of using automatic differentiation exceptions on how the mechanical problem and a corresponding numerical model are postulated and solved. The ADB form of the problem description is rather straightforward for e.g. total-Lagrangian displacement-based finite element formulation of hyperelastic problems. However, it can be nontrivial for e.g. spatial formulation of finite strain elasto-plastic finite elements. The ADB form is presented for a general formulation of primal and sensitivity analysis of coupled transient problems; total-Lagrangian and spatial formulation of elastic, hyperelastic, small and finite strain elasto-plastic problems; nonconservative forces; derivation of geometric tangent matrix needed within the solution of linear-buckling problems; frictional contact problems with penalty and augmented Lagrangian constraints and extended system model for direct determination of bifurcation and limit points. It is shown that the numerical efficiency of automatically derived program codes based on ADB notation is comparable to that of the manually coded finite elements.

## References

- [1] D. Eyheramendy, Th. Zimmermann, *Object-oriented symbolic derivation and automatic programming of finite elements in mechanics*, Engineering with Computers, 15, 1, 12-36 (1999)
- [2] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Philadelphia, PA: SIAM (2000)
- [3] J. Korelc, *Automation of primal and sensitivity analysis of transient coupled problems*, Computational mechanics, 44:631-649 (2009)
- [4] J. Korelc, *Automation of the finite element method*. WRIGGERS, Peter. Nonlinear finite element methods. Berlin Springer, 483-508 (2008)
- [5] J. Korelc, *Multi-language and multi-environment generation of nonlinear finite element codes*. Eng. comput., 18: 312-327 (2002)
- [6] A. Logg, *Automating the Finite Element Method*. Archives of Computational Methods in Engineering, 14:93-138 (2007)
- [7] J. Korelc, *AceGen and AceFEM user manual*, [www.fgg.uni-lj.si/symech/](http://www.fgg.uni-lj.si/symech/)