11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
July 20–25, 2014, Barcelona, Spain

# INTERACTIVE DEBUGGING OF AUTOMATICALLY GENERATED NUMERICAL CODES

**Jože Korelc**

University of Ljubljana, Faculty of Civil and Geodetic Engineering,
Jamova cesta 2, 1000 Ljubljana, Slovenia,
E-mail: joze.korelc@fgg.uni-lj.si , web page: http://www3.fgg.uni-lj.si

**Key words:** *Automatic coding, Symbolic approach, Finite elements, Debugging*

Symbolic algebra systems such as Mathematica are general and very powerful tools for the manipulation of formulae and for performing various mathematical operations by computer. However, in the case of complex numerical models, direct use of these systems is not possible. Two reasons are responsible for this fact: a) during the development stage the symbolic derivation of formulae leads to uncontrollable growth of expressions and consequently redundant operations and inefficient programs, b) for numerical implementation computer algebra systems can not keep up with the run-time efficiency of programming languages like FORTRAN and C and by no means with highly problem oriented and efficient numerical environments used for finite element (FE) analysis. However, large FE systems can be awkward for the development and testing of new numerical procedures. The basic tests which are performed on a single finite element or on a small patch of elements can be done most efficiently by using the general symbolic-numeric environments. It is well known that many design flaws, such as element instabilities or poor convergence properties, can be easily identified if we are able to investigate element quantities on a symbolic level. In order to assess element performances under real conditions we have to perform large scale tests using numerically efficient programming languages. In order to meet all this demands in an optimal way, a hybrid system for multi-language and multi-environment generation of numerical codes is needed.

The paper presents some aspects of interactive debugging and analyzing of automatically generated numerical codes. An important question arises: how to understand the automatically generated formulas? The automatically generated code should not act like a "black" box. For example, after using the automatic differentiation tools we have no insight in the actual structure of the derivatives. One my argue that this is not needed, however if the finite element code are to be derived with the efficiency comparable with the manually written codes then an efficient debugging procedures become a necessity. The standard way in which symbolic tools work is to evaluate a single expression completely, which can be then analyzed and printed in a required format. If the problem is complex with potentially hundreds of expressions and it includes loops and branching, then more sophisticated procedures for analyzing and interactive run time debugging are required. Additionally, when optimization of the derived numerical code is performed simultaneously with the derivation of formulas, the explicit form of the expressions is lost.

Some approaches which can help to overcome presented problems were implemented into the Mathematica packages AceGen (www.fgg.uni-lj.si/symech) that is used for automatic generation of numerical codes and AceFEM finite element environment (www.fgg.uni-lj.si/symech). First, the AceGen extends the collections of forms in which Mathematica can represent expressions with expression browser, where automatically generated auxiliary variables are represented as active areas (buttons) which can then be further explored.

During the simulation run time the current values of the auxiliary variables can be presented instead of the formulas. For an efficient interactive debugging it is important where the debugging actually takes place. One possibility is to run simulation code under whatever debugger is provided for particular compiled language. However, this approach is not applicable for automatically generated codes, since the structure of the automatically generated codes is rather "unreadable", thus difficult to debug directly. The approach where the generated code is not debugged directly, but we look at representation of the expressions and their current values in Mathematica, is essential for debugging of automatically generated codes. The procedure has been implemented in AceGen/AceFEM system and will be presented on several examples.