Applications of Computer Algebra in Scientific Computing[1]

Hans-Joachim Bungartz
FORTWIHR — The Bavarian Consortium
for High Performance Scientific Computing
c/o Institut für Informatik der
Technischen Universität München
D-80290 München, Germany
bungartz@informatik.tu-muenchen.de

Summary

In the field of scientific computing, the methods and tools of classical numerical analysis are traditionally much more common than the use of computer algebra systems, which is mainly due to the fact that computer algebra languages can't keep up with the run-time efficiency of programming languages like FORTRAN, C, or C++. Nevertheless, modern computer algebra systems can support both the development, the theoretical analysis, and the efficient use of algorithms and codes in the different areas of application of scientific computing. Therefore, it turns out to be more and more important to bring together the two worlds of numerical and symbolic computation.

We report a couple of situations where computer algebra tools, recently, have been successfully integrated in scientific computing projects. Here, the areas of application range from support in proving approximation properties of discretization schemes to data compression for purposes of visualization and multimedia applications.

keywords: scientific computing, partial differential equations, sparse grids, rapid prototyping, higher order methods, data compression;

1. Introduction: Computer Algebra and Scientific Computing

At first glance, scientific computing and computer algebra don't seem to have a great deal in common: The first discipline deals with numerical computations and, consequently, most of the time with approximations based on a fixed precision. The

[1] Extended abstract of the talk given at the IMACS Conference on Applications of Computer Algebra, May, 16–19, 1995, Albuquerque, New Mexico, USA.

focus of computer algebra, however, are symbolic (i. e. exact, especially) calculations with numerical mathematics appearing as late as possible and with a theoretically arbitrary precision. In spite of these putative contrasts, there are a lot of starting-points where methods and tools of computer algebra can support and accelerate developments in scientific computing:

- support in numerical analysis (both in proving convergence properties and in in the verification of algorithms, e. g.);
- educational aspects (possibility of dealing with problems beyond simple model problems even in introductory courses);
- rapid prototyping (possibility of quick test implementations to evaluate the potential of new algorithmic ideas, e. g.).

In fact, rapid prototyping turns out to be very helpful during the process of getting first numerical results with a new idea or a new algorithm. It is one of the main disadvantages of standard programming languages like FORTRAN, C, or C++ that the overhead due to decisions concerning data structures, due to definitions, declarations, and, last but not least, to the syntax of the respective language, is very significant. Therefore, in order to decide whether your specific idea is worth implementing in a standard and run-time efficient language (with all the overhead mentioned above), you may even have to do a big part of the implementation, which is a prime example for slow prototyping. In contrast to that, computer algebra languages very often allow a quick and straightforward implementation of a test version, based on which a decision of the further proceeding can be taken.

2. Proving Convergence Properties of Sparse Grid Methods

Sparse grid methods are a very efficient approach to the discretization of partial differential equations (for a detailed introduction to sparse grids, see [6] and [1], e. g.). In addition to the direct application of finite element methods to sparse grids, the combination technique, an approach closely related to extrapolation techniques, has been developed (see [5]). Based on first results presented in [2], some principal convergence properties of the combination technique could be shown with the help of an extended interactive use of MAPLE [3, 4]. Here, computer algebra provided decisive support in finding appropriate substitutions for large terms, in identifying quantities without asymptotic relevance, and, finally, in determining the asymptotic order of the error of the combination solution.

3. Generating Hierarchical Polynomial Bases for Sparse Grids

As we can see in [1], e. g., sparse grids allow a very efficient representation of functions of d variables or of d-dimensional data, respectively. Up to now, only piecewise d-linear basis functions have been used to construct a sparse grid interpolant or a sparse grid finite element approximation to the solution of a partial differential equation. Therefore, the aim of defining and experimenting with hierarchical polynomial bases of higher order suitable for sparse grid methods has become the focus. There is a large variety of applications of such an approach that have been dealt with so far, ranging from data compression for image processing, video conferences, or contour line models used in flight simulators, to p-version-type finite element algorithms for sparse grids.

In order to make experiments with higher order hierarchical bases easier, a computer algebra environment that allows the fully automatic generation of such polynomial bases in four steps has been developed. First, you have to specify the desired type of hierarchical basis (i. e., the polynomial degree p and the way of fixing the degrees of freedom: Lagrange interpolation, Hermite interpolation, and so on). Here, we choose a hierarchical Lagrange interpolation: The basis function corresponding to a grid point i takes the value of 1 in i and is 0 in certain hierarchical ancestors of i. This leads to the generation of a set of prototypes of the basis functions on the coarsest level. Second, the corresponding hierarchical basis functions and their supports are constructed for each grid point i of the finer levels. Thus, the hierarchical basis is complete. In a third step, the operators for basis transformation (switching from the hierarchical basis to a nodal point representation and vice versa) are calculated, followed by the fourth step of an explicit construction of the corresponding nodal point basis.

4. Generating Stiffness Matrices for FEM on Sparse Grids

In addition to the four steps of the scheme described in the previous section, the explicit knowledge of the hierarchical basis can be used as input for the symbolic computation of stiffness matrices for given partial differential equations in a finite element context. Such a construction turns out to be helpful for both the analysis of the convergence behaviour of the underlying sparse grid finite element method (condition of the stiffness matrix, spectral radius of the iteration matrix), for the analysis of the sparsity of the matrix itself, for rapid prototyping in a pde context (test some small examples and study the behaviour of your new algorithm / your new higher order basis), and, finally, for the verification of sophisticated and optimized (C-) codes implemented afterwards.

Thus, the whole process from the principles of construction of a certain hierarchical basis suitable for sparse grid methods to the generation of the resulting sparse grid finite element stiffness matrix has been realized symbolically with the help of computer algebra tools. Such an approach has turned out to be very advantageous for the development of efficient algorithms in scientific computing.

REFERENCES

[1] H.-J. Bungartz, *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimen-*

sionalen Poisson-Gleichung, PhD thesis, Institut für Informatik, TU München, June 1992.

[2] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger, *A proof of convergence for the combination technique for the Laplace equation using tools of symbolic computation*, in Proceedings of the International IMACS Symposium on Symbolic Computation, Lille, 14.–17. 6. 1993, G. Jacob, N. Oussous, and S. Steinberg, eds., IMACS/USTL, 1993, pp. 56–61.

[3] ———, *Pointwise convergence of the combination technique for the Laplace equation*, East-West Journal of Numerical Mathematics, 2 (1994), pp. 21–45.

[4] ———, *Two proofs of convergence for the combination technique for the efficient solution of sparse grid problems*, in Domain Decomposition Methods in Scientific and Engineering Computing (Contemporary Mathematics, Vol. 180), Proceedings of the 7^{th} International Conference on Domain Decomposition Methods, 27.–30. 10. 1993, The Pennsylvania State University, D. Keyes and J. Xu, eds., American Mathematical Society, Providence, 1994, pp. 15–20.

[5] M. Griebel, M. Schneider, and C. Zenger, *A combination technique for the solution of sparse grid problems*, in Proceedings of the IMACS International Symposium on Iterative Methods in Linear Algebra, Brussels, 2.4.–4.4. 1991, P. de Groen and R. Beauwens, eds., Elsevier, Amsterdam, 1992.

[6] C. Zenger, *Sparse grids*, in Parallel Algorithms for Partial Differential Equations: Proceedings of the Sixth GAMM-Seminar, Kiel, 19.1.–21.1. 1990, Notes on Numerical Fluid Mechanics (Vol. 31), W. Hackbusch, ed., Vieweg, Braunschweig, 1991.