Code Generation Using Computer Algebra Systems

Michael Wester
Cotopaxi, Albuquerque, New Mexico
`wester@math.unm.edu`
`http://math.unm.edu/~wester/`
and
John K. Prentice
Quetzal Computational Associates, Inc.
`john@quetzalcoatl.com`
`http://www.quetzalcoatl.com`

Abstract

The strength of computer algebra systems (CASs) is their ability to symbolically manipulate mathematics. CASs can also numerically evaluate symbolic expressions. However, compared to compiled programming languages such as Fortran or C, CASs are very slow at numerical evaluation.[1] If the numerical evaluations are simple or only being done occasionally, the speed of evaluation may not be of importance. However, if the evaluations are complicated or have to be repeated on large data sets, the speed of evaluation can become of paramount importance. In such cases, the general approach is to generate Fortran or C expressions that are equivalent to the symbolic expression and use these translated expressions as part of a larger Fortran or C program. Conversion of symbolic expressions or procedures into Fortran or C is also valuable when the CAS is being used to derive symbolic results for the specific purpose of including these results in Fortran or C programs. In scientific research, the generation of Fortran or C is in fact often one of the major uses of CASs. To simplify this task of converting symbolic results into Fortran or C, all major general purpose CASs provide one or more utilities for translating symbolic expressions, and in some cases, procedures, into these computer languages.

In this talk, we will examine the code generation utilities provided with the Axiom, Macsyma, Maple, Mathematica, MuPAD, and REDUCE computer algebra systems. These six systems were chosen for examination because they represent the vast majority of the CASs in use today; hence, they are representative of the capabilities of CASs in general. We will discuss the capabilities of the code generation utilities provided with these CASs, critique the code that is generated, and finally discuss some numerical dangers associated with the naive use of the Fortran and C produced by these utilities. The intent is not to provide a comprehensive or rigorous comparison of the capabilities of these six systems, but rather to give the listener a feel for the code generation capabilities of general purpose CASs, using examples from these six systems. The hope is that this will be valuable to users in deciding whether and how to use CASs to translate symbolic expressions or procedures into Fortran and C code.

[1] As an example, using Maple V Release 5, it takes 2.9 seconds to perform 100,000 evaluations of the exponential of a real number on a 266 MHz WinBook XL laptop running Windows 95. By comparison, it takes less than 0.16 seconds to do the same calculation in Fortran 90 on the same computer using Digital's Visual Fortran version 5.0 compiler.