Journal of Symbolic Computation 60 (2014) 120–136

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Automated simplification of large symbolic expressions

David H. Bailey[a,1], Jonathan M. Borwein[b], Alexander D. Kaiser[c]

[a] *Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States*
[b] *Centre for Computer Assisted Research Mathematics and its Applications (CARMA), Laureate Professor, Univ. of Newcastle, Callaghan, NSW 2308, Australia*
[c] *Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, United States*

A R T I C L E   I N F O

Article history:
Received 17 July 2013
Accepted 13 September 2013
Available online 24 September 2013

Keywords:
Simplification
Computer algebra systems
Experimental mathematics
Error correction

A B S T R A C T

We present a set of algorithms for automated simplification of symbolic constants of the form $\sum_i \alpha_i x_i$ with α_i rational and x_i complex. The included algorithms, called `SimplifySum`[2] and implemented in *Mathematica*, remove redundant terms, attempt to make terms and the full expression real, and remove terms using repeated application of the multipair PSLQ integer relation detection algorithm. Also included are facilities for making substitutions according to user-specified identities. We illustrate this toolset by giving some real-world examples of its usage, including one, for instance, where the tool reduced a symbolic expression of approximately 100 000 characters in size enough to enable manual manipulation to one with just four simple terms.

© 2013 Published by Elsevier B.V.

1. Introduction

A common occurrence for many researchers who engage in computational mathematics is that the result of a Computer Algebra System (CAS) operation, in say *Mathematica* or *Maple*, is a very

E-mail addresses: dhbailey@lbl.gov (D.H. Bailey), jonathan.borwein@newcastle.edu.au (J.M. Borwein), kaiser@cims.nyu.edu (A.D. Kaiser).
URLs: http://crd.lbl.gov/~dhbailey/ (D.H. Bailey), http://carma.newcastle.edu.au/jon/ (J.M. Borwein), http://cims.nyu.edu/~kaiser (A.D. Kaiser).

[1] Supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231.

[2] Available from https://github.com/alexkaiser/SimplifySum.

0747-7171/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.jsc.2013.09.001

long, complicated expression, which although technically correct, is not very helpful; only later do these researchers discover, often indirectly, that in fact the complicated expression they produced further simplifies, sometimes dramatically, to something much more elegant and useful. With some frequency the CAS will provide no answer and may well 'hang'. Such events are to be expected, given the limitations of any symbolic computing package, for any of a number of reasons, including the difficulty of recognizing when a given subexpression is zero.

Such instances are closely related to the problem of recognizing a numerical value as a closed-form expression. In this case, researchers have used integer relation-finding algorithms, such as the PSLQ algorithm and its variants (Bailey and Broadhurst, 2000), to express the given numerical value as a linear sum of constants or terms. In both instances, researchers seek as simple a closed-form expression as possible. Such simplified closed-form expressions are highly desirable, both in mathematical research and in problems, say, from mathematical physics. The various definitions and importance of closed forms is described in Borwein and Crandall (2013), Chow (1999). Examples of such work are described in Bailey et al. (2010b) and Borwein et al. (2010).

We present herein a software package `SimplifySum` for the simplification of symbolic constants of the form $\sum_i \alpha_i x_i$, where each α_i is rational and each x_i real or complex. Such constants frequently arise in looking for closed forms for integrals or sums, and are frequently large and machine-generated by symbolic mathematics software such as *Mathematica* or *Maple*.

Implemented in *Mathematica*, our package includes a focused set of tools for simplification of such constants. The package is able to remove redundant constants, opposites and conjugates, symbolically, numerically or both. It can simplify complex terms, which is useful if some x_i are complex yet the whole constant is real. The package uses symbolic algebra to repeatedly apply the multipair variant of the PSLQ integer relation detection algorithm, and reduces expressions using exact, rational number arithmetic. It also contains code to apply substitutions, so the user may specify identities or substitutions they would like performed. These tools can be accessed through a convenient, simple function interface. Users also have access to the individual functions that can thence be customized.

The criterion to decide which version of an expression is simpler is straightforward — a sum that has fewer terms is simpler.[2] All the algorithms (with the exception of substitution) only process the rational coefficients α_i, so an expression $\sum_{i=1}^m \alpha_i x_i$ is simpler than $\sum_{i=1}^n \beta_i x_i$ if $m < n$. This is in contrast to the general case discussed in Carette (2004), where the question of which version of a general expression is considered and formalized. Because of the restriction to sums, our definition is nearly always consistent with Carette's formalism. In general, the package does not alter x_i in simplification, it only removes terms and alters the rational coefficients. Unless the new α_i are very complex, the resulting expression is simpler. Note that if the user requests substitutions (as described in Section 3) then the substitution will be made regardless of whether this reduces or increases the complexity.

The tools have proven quite effective. Many computer-generated constants have instances of the simple redundancies described above. The techniques using integer-relations are general and reliable, provided numerical results are used with appropriate caution. The substitutions allow the user to apply specific identities automatically. This will allow them to use identities that arise repeatedly in particular work, but are not in *Mathematica*.

Note also that the restriction to sums is very general — no limit is placed on each x_i, only that it must evaluate to a real or complex number, so each x_i may be arbitrarily complicated. Each term will be treated as a single constant by all parts of the code, with the exception of substitutions.

The remainder of the paper is structured as follows. In Section 2 existing literature on simplification and simplification in current CAS is discussed. In Section 3 the general structure of `SimplifySum` is described. Precise descriptions of the package are relegated to Appendix A. In Section 4 we give a variety of illustrative examples, then conclude in Section 5 with timing and results on some large research constants.

All tests were performed on a stock 2012 MacBook Pro with a 2.9 GHz Intel Core i7 processor and 8 GB of RAM using *Mathematica 7.01*.

[2] For short expressions this may occasionally lead to less elegant presentation; for longer ones it seems highly desirable.