Symbolic Computing

Akinwale M. Oteniya, Matthew N. O. Sadiku, and Sarhan M. Musa
Roy G. Perry College of Engineering
Prairie View A&M University
Prairie View, TX 77446
Email: tennymatt@gmail.com; sadiku@ieee.org; smmusa@pvamu.edu

DOI: 10.29322/IJSRP.10.03.2020.p9914
http://dx.doi.org/10.29322/IJSRP.10.03.2020.p9914

ABSTRACT

Symbolic computation refers to using machines or computers to manipulate mathematical equations and expressions in symbolic form, as opposed to numerical manipulation. Using symbolic computing to solve mathematical problems involves manipulations of symbolic objects, rules or programs, with the main goal of being exact.

This is unlike most numeric calculations where computations use approximate floating point arithmetic. This paper provides a brief introduction to symbolic computing, its applications, benefits, and challenges.

Key words: symbolic computing, algebraic computing, computer algebra systems

INTRODUCTION

Conducting scientific research via numerical computation is an established practice today. Computing is powerfully impacting the way that modern science and engineering are carried out. Symbolic computing (or algebraic computing) is one of fastest growing areas of scientific computing. Since its earliest development in 1953, symbolic computing has increased in popularity and has been used in science, engineering, and other disciplines.

Symbolic computing is mainly concerned with the representation of information in symbolic form and how that information can be processed using computer systems. It allows solutions to be given exactly. Symbolic computing simplifies and streamlines calculation and reduces the potential for arithmetic error [1].

In contrast with numeric or conventional representation, symbolic computing deals with the representation and manipulation of information in symbolic form. Examples of numeric and symbolic computations are:

- Numeric multiplication: F = 6 * 5 = 30.
- Symbolic multiplication: $F = (2+j)*x_1*x_2*x_3 + x_1 + x_2 + 5*\sin(2*x_4)$

Symbolic computing can be used for symbolic integration or differentiation, substitution of one expression into another, simplification of an expression, etc. Symbolic computation requires large computational resources. It entails developing an infrastructure built upon massively distributed computational environments

CONCEPT OF SYMBOLIC COMPUTING

In a nutshell, symbolic computing is a computer algorithm that requires input from the user in either numerical values or exact terms such as fractions, radicals, and symbols and then performs mathematical operations on the input. Both the input and output are symbolic. The tools for symbolic computations are standalone systems that require human interpretation and control. A symbolic computer has three major components: architectures, languages, and algorithms. Parallel computing requires coordinated evolution of these three components.

A key requirement in symbolic computation is to efficiently combine computer algebra systems (CASs) in order to be able to solve complex problems. CAS has become the main tool for symbolic computations. MACSYM, developed at MIT, was one of the oldest CASs. Common examples of CAS are the Symbolic Computation Software Composability Protocol (SCSCP) and SymPy [2]. A computer algebra system stores mathematical expressions as data structures.

Languages commonly used for symbolic computing include PORTRAN, C, C++, Lisp, ML, Prolog, Python, and Java.

. Examples of symbolic algorithms include sorting, compiling, symbolic algebra, procedure calling, expert systems, and artificial intelligence systems.

EXAMPLE OF SYMBOLIC COMPUTING

One of the major successes in symbolic computation research has been the development of software systems. A representative example of software capable of performing symbolic computing is SymPy. SymPy is a computer algebra system (CAS) written in the Python, which is programming language that has a focus on ease of use and readability. It is a robust CAS which provides a wide spectrum of features in a wide range of scientific disciplines. Unlike several other CASs, SymPy does not attempt to develop and use its own programming language. Rather, it exclusively uses Python both as the internal language and the user language. This makes it easier for people already familiar with Python to use SymPy. All operations are performed symbolically. Symbolic variables or simply symbols, must be defined and assigned to Python variables. SymPy also supports matrices with symbolic dimension values. Computations on matrices with symbolic entries are necessary for many algorithms in SymPy. SymPy has equation solvers that can handle ordinary differential equations, recurrence relationships, and algebraic equations [3].

SymPy supports a wide range of mathematical facilities such as discrete math, concrete math, plotting, geometry, statistics, sets, series, vectors, combinatorics, group theory, code generation, tensors, simplifying expressions, performing common calculus operations, manipulating polynomials, pretty printing expressions, solving equations, precision numerics, and representing symbolic matrices. It also supports several higher mathematical functions such as gamma functions, Bessel functions, orthogonal polynomials, elliptic functions and integrals, zeta and polylogarithm functions, and generalized hypergeometric function are supported [3]. The fact that SymPy is readable, free, and open source makes it attractive in spite of its limitations.

APPLICATIONS

Symbolic computing is useful in different areas of mathematics such as computer algebra, linear algebra, computational logic, symbolic geometric methods, solution of algebraic equations, equational theorem proving and rewriting, factorization of polynomials and operators, integration theory, and recurrence relation [4]. Whatever the application, users have a variety of symbolic processors to choose from. These include MAPLE, MATHEMATICA, MATHCAD and MAXIMA.,

Functional programming and logic programming are two common applications of symbolic computing. To functional programming, *p*ure functions are the basic unit of computation and they are entirely characterized by the results they return for a given set of arguments. To logic programming, logical predicates are the basic unit of computation and a predicate is defined as a true relationship [5].

BENEFITS AND CHALLENGES

Symbolic computation has been used extensively in developing closed form solutions to engineering problems. It has been observed that the use of symbolic computational improves the reliability of calculations as well as provides a method to shorten the time required for performing extensive and repetitive calculations [6].

Development of symbolic computing has been slow until recently due to the inadequacy of available computational resources such computer memory and processor power. Now a variety of symbolic computing software packages are available for engineering use. Also, computer algebra systems (CASs) are now capable of solving large problems such as the Grid [7]. Symbolic computations are hard to parallelize due to their complex data and control structures.

CONCLUSION

Scientific research via symbolic computation is less widespread than other techniques.

However, with the advent of time-sharing and virtual memory operating systems, research on symbolic computation have flourished. Notable, impressive results have been achieved in symbolic computation over the last two decades [8]. Symbolic computation has made mathematics more useful to other scientific fields. It is now routinely used in a number of diverse disciplines such as mathematical research, physics, engineering, education, and economic. The needs for symbolic computation have grown.

Universities should introduce symbolic computation as a tool into existing courses.

For additional information on symbolic computing, one should consult books in [9-14] and the journal exclusively devoted to it: *Journal of Symbolic Computation.*

REFERENCES

[1] A. B. Sibley et al., "Facilitating the calculation of the efficient score using symbolic computing," *The American Statistician*, vol. 72, no. 2, 2018, pp. 199-205.

[2] S.Lintona et al., "Easy composition of symbolic computation software using SCSCP: A new lingua franca for symbolic computation," *Journal of Symbolic Computation*, vol. 49, 2013, pp. 95-119.

[3] A. Meurer et al.,"SymPy: symbolic computing in Python," *PeerJ Preprints*, June 2016.
https://github.com/sympy/sympy-paper.

[4] F. Winkler, "What can symbolic computation contribute to mathematics?" *Proceedings of the 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2011, pp. 19-20.

[5] R. D. Cameron, "Introduction to symbolic computing," September 1999,
http://www.cs.sfu.ca/people/Faculty/cameron/Teaching/384/99-3/intro.html

[6] E. L. Richards, "Application of symbolic computing in analysis of modal properties of structurally coupled twin tall buildings," *M. Sc. Thesis*, Colorado State University Fort Collins, Colorado, Spring 2011.

[7] D. Petcu, "Distributed symbolic computations," *Proceedings of the Sixth International Symposium on Parallel and Distributed Computing*, July 2007

[8] A. Boyle and B. F. Caviness, "Future directions for research in symbolic computation," April , 1988
https://www.eecis.udel.edu/~caviness/wsreport.pdf

[9] Frank E. Harris, *Mathematics for Physical Science and Engineering: Symbolic Computing, Applications in Maple and Mathematica*. Academic Press, 2014.

[10] B. Harvey, *Computer Science Logo Style. Volume 1" Symbolic Computing.* Cambridge, MA: The MIT Press, 2nd ed., 1997.

[11] D. S. Touretzky, *COMMON LISP: A Gentle Introduction to Symbolic Computation.* Redwood City, CA: The Benjamin/Cummings Publishing Company, 1990.

[12] U.Langer and P. Paule (eds.), *Numerical and Symbolic Scientific Computing: Progress and Prospects.* Springer, 2012.

[13] E.G. Rajan, *Symbolic Computing : Signal and Image Processing*. Anshan, 2005.

[14] R. A. Mueller and R. L. Page, *Symbolic Computing With Lisp And Prolog*. Wiley, 1988.

ABOUT THE AUTHORS

Matthew N.O. Sadiku (sadiku@iee.org) is a professor at Prairie View A&M University, Texas. He is the author of several books and papers. He is an IEEE fellow. His research interests include computational electromagnetics and computer networks.

Akinwale M. Oteniya (tennymatt@gmail.com) received his MSc degree from Prairie View A&M University. His research interests include robotics, micro devices, smart technologies, electrical distribution, cyber security, and big data.

Sarhan M. Musa (smmusa@pvamu.edu) is a professor in the Department of Engineering Technology at Prairie View A&M University, Texas. He has been the director of Prairie View Networking Academy, Texas, since 2004. He is an LTD Sprint and Boeing Welliver Fellow.