УДК 519.651:004.222.3

О.Я. НИКОНОВ, д.т.н., проф., зав. каф. ХНАДУ, Харьков,
О.В. МНУШКА, ассистент ХНАДУ, Харьков,
В.Н. САВЧЕНКО, ст. преп. УИПА, Харьков

ОЦЕНКА ТОЧНОСТИ ВЫЧИСЛЕНИЙ СПЕЦИАЛЬНЫХ ФУНКЦИЙ ПРИ РАЗРАБОТКЕ КОМПЬЮТЕРНЫХ ПРОГРАММ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Проведен анализ формата вещественных чисел IEEE-754 как составляющей дополнительной погрешности при вычислении специальных функций. Проведено сравнение и показано, что применение арифметики с произвольной точностью (mpfr_erf(), MPFR) и стандартной функции (erf(), C99) в программах на C/C++ снижает быстродействие на порядок при той же точности. Ил.: 2. Библиогр.: 8 назв.

Ключевые слова: формат IEEE-754, произвольная точность, точность вичислений специальніх функций.

Постановка проблемы. Применение статистических методов в математическом моделировании сложных технических систем, например, спутниковых систем связи [1], позволяет значительно снизить временные и материальные затраты на их разработку, оценить воздействие различных влияющих факторов на поведение системы в реальных условиях эксплуатации. Многократное повторение вычислений в процессе моделирования может приводить к значительным погрешностям результатов, обусловленных ограниченным числом разрядов представления вещественных чисел в памяти компьютера. Специфика многих задач не позволяет пользоваться универсальными пакетами моделирования, требует реализации на одном из универсальных языков программирования и оценки точности вычислений, и адекватности полученных численных результатов.

Анализ литературы. Представление вещественных данных в памяти компьютера регламентируется стандартом IEEE754-2008 [2], в котором определены четыре формата двоичных чисел – *binary16, binary32, binary64, binary128,* и три формата десятичных чисел – *decimal32, decimal64, decimal128,* а также способы расширения базовых форматов для повышения точности вычислений. Точность представления чисел в определяемых стандартом форматах составляет соответственно 7, 16, 34 десятичных разрядов после запятой. Существенным отличием данной редакции стандарта от предыдущей [3] является отсутствие формата *extended* с размером данных 80 бит, который жестко привязан к размерности регистров сопроцессора x87. Способы реализация стандарта

для языка C++ предлагаются в [4], а сама поддержка стандарта в разной степени реализована в разных компиляторах языка C/C++. Таким образом, требуется оценка кода, получаемого при помощи различных компиляторов, для учета дополнительной погрешности результатов вычислений, обусловленной реализацией алгоритмов вычислений.

Цель статьи – оценка дополнительной погрешности результатов моделирования численными и статистическими методами, возникающей за счет приближенного представления данных в памяти компьютера и форматов вещественных данных языка программирования C/C++.

Основная часть. Основными преимуществами языка C/C++ для научных вычислений является его универсальность, кроссплатформенность и переносимость кода. При этом, если для вычислений с целыми типами данных существует более или менее неплохая совместимость между различными компиляторами и вычислительными платформами, то в случае с вещественными типами данных существует ряд нестыковок, которые могут приводить к неоднозначной трактовке результата и даже потере данных.

В статистических методах математического моделирования часто возникает задача вычисления функции ошибок, в частности, при построении информационных карт вероятности ошибки систем спутниковой мобильной связи. Функция ошибки определена как [4]

$$erf\ x = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \ \text{ или } \ erfc\, x = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt\,. \qquad (1)$$

Для функций (1) не существует аналитического решения, поэтому для ее вычисления с высокой точностью используют различные приближения, от выбора которых зависит не только точность вычислений, но и быстродействие программы моделирования.

Наиболее часто функция *erf*() аппроксимируется рядами [5] или различными минимаксными полиномиальными приближениями, например, рациональной чебышевской аппроксимацией [6], для которой существует стандартная библиотечная функция на фортране [7]. Недостатком такого подхода является фиксированная точность получаемого результата, что не всегда оправдано с точки зрения физической формулировки задачи и дополнительных накладных вычислительных расходов. Разложение в ряд может быть использовано для построения алгоритмов вычисления с произвольной точностью результата. На таком принципе основаны широко распространенные библиотеки численных методов, такие как GMP (GNU библиотека

арифметики с произвольной точностью) и ее производные: MPFR (библиотека языка Си для вычислений с произвольной точностью и корректным округлением результата), MPC (библиотека для вычислений с комплексными числами с произвольной точностью) и т.д.

Снижение погрешности результатов моделирования статистическими методами требует большого числа испытаний, что может приводить к значительным затратам времени и накоплению ошибки, обусловленной недостаточной точностью промежуточных результатов. С другой стороны, необходимо удерживать большое количество значащих цифр, что требует оценки целевой вычислительной платформы на предмет способности хранить эти знаки.

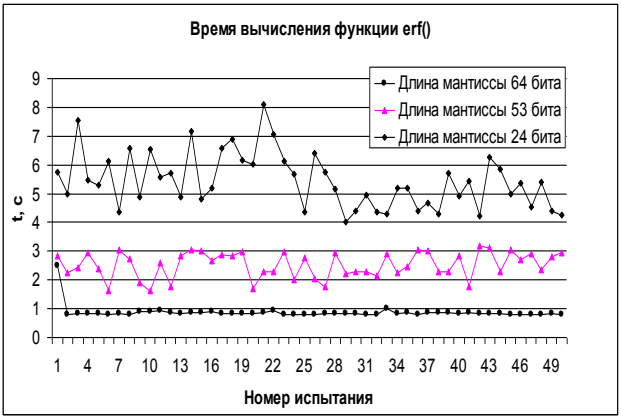
В исследовании использовались компиляторы: Microsoft Visual C++ из состава Microsoft Visual Studio 2008(2010); gcc 4.x (Debian 4.4.5-8) и 4.5.0 (mingw); Intel C Compiler 11.1.054 (Windows), 11.1.064 (Debian amd64), 11.1.074 (Debian x86). Авторами была составлена программа на языке C++ для сравнения эффективности работы различных алгоритмов вычисления интеграла ошибки. Программа позволяет провести указанные вычисления с использованием функции mpfr_erf() из пакета MPFR, erf() и erfc() из стандартной библиотеки C/C++, а также функций на основе аппроксимации, предложенной Cody [6, 7] и аппроксимации рядами по формуле [(7.1.26), 5].

На рис. 1, а показано время, затраченное функцией mpfr_erf(), которая вычисляла интеграл ошибки с точностью 32 (7-8 десятичных разрядов), 53 (15-16 десятичных разрядов) и 64 (19-20 десятичных разрядов) бита, что соответствует базовым числовым типам данных языка C/C++. Было проведено 50 испытаний, в каждом из которых вычислялось значение функции ошибки в цикле 100000 раз. Увеличение длины мантиссы от 23 до 64 разрядов увеличивает среднее время вычислений приблизительно в 5 раз (рис. 1, а).

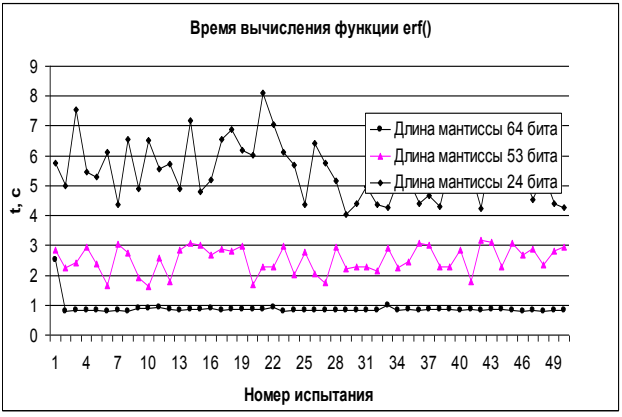
Проведено сравнение работы функции на основе аппроксимации, предложенной Cody [6, 7] и стандартной, определенной стандартом языка C99 (рис. 1, б). Для оценки времени выполнения количество повторений цикла было увеличено в 10 раз, время необходимое стандартному алгоритму оказалось пренебрежимо малым, функция на основе аппроксимации [6, 7] также оказалась значительно быстрее функции с произвольной точностью. Следует отметить, что при компиляции программы не производилась оптимизация под используемую вычислительную платформу.

Учитывая высокие скоростные показатели работы стандартных библиотечных функций, возникает необходимость оценки точности

представления вещественных чисел в памяти компьютера и их интерпретации компиляторами C/C++.



а)



б)

Рис.1. Время вычисления функции ошибок: а) с произвольным числом разрядов; б) на основе реализации алгоритма Cody и стандартного (C99)

Авторами проведен анализ поддержки форматов вещественных чисел наиболее распространенными компиляторами языка C/C++. В зависимости от реализации компилятора и операционной системы вещественные типы бывают: float (binary32), double (binary64), long

double и занимают 10 (extended [3]), 12 или 16 (binary128) байт в памяти. Анализ полученных данных показывает, что: для данных в формате long double компилятором выделяется различное количество байт для хранения одного экземпляра переменной в памяти; неоднозначная интерпретация компиляторами C/C++ этого формата требует аккуратности и обоснованности его применения; большее число байт должно обеспечивать большую точность представления чисел.

Для оценки точности вычислений в различных форматах данных была составлена программа вычисления $\pi = 4 \cdot arctg(1.0)$. Анализ полученных данных позволяет сделать вывод о том, что: 1) переход от формата double к long double позволяет получить дополнительно 3 точных цифры результата; 2) переменные формата long double компиляторами icc и gcc/g++ преобразуются в формат extended (по IEEE754-1985), а MS VC++ в формат double; результат может содержать "информационный мусор", который отображается на экране и может быть неверно истолкован; 3) заголовочный файл cmath для компилятора MS VC++ отличается от аналогичного для компилятора gcc/g++ отсутствием ряда часто используемых функций, а также не определяет способов работы с форматами long double размером больше 8 байт; 4) только арифметические операции для переменных типа long double гарантированно выполняются; переменные типа long double (больше 8 байт) могут некорректно выводиться как стандартным оператором (printf()), так и потоковым (cout) в ОС Windows. Результат, который будет отображаться на экране, в данном случае непредсказуем; 5) работа с форматом long double в ОС Linux достаточно корректна. Определен набор функций для переменных этого формата, при этом ввод-вывод также работает корректно, единственным недостатком следует считать "информационный мусор", который отображается после 18 десятичного разряда (а для формата double – после 15).

Была проверена погрешность вычисления erf() по формуле [(7.1.26), 5] с аппроксимацией Cody. Было установлено, что первая дает погрешность по сравнению с аппроксимацией Cody 1e-7 (рис. 2, а) в диапазоне x = [0, ..., 3], погрешность реализации аппроксимации Cody и стандартной C99 дает погрешность не более чем 1e-17 (рис. 2, б) по сравнению с "точным" значением. Уменьшение погрешности вычислений (рис. 2, а) объясняется недостатком разрядов для хранения числа. Дальнейшее повышение точности вычислений возможно только с использованием специальных приемов [8].

а)

б)

Рис. 2. Погрешность вычислений функции ошибок:
а) для типа float; б) для типа long double

Выводы. 1. Достижимая погрешность вычислений, которая обеспечивается средствами стандартной библиотеки языка C++ не превышает 18 десятичных разрядов, для достижения более высокой точности следует применять библиотеки вычислений с произвольной точностью – MPFR, интервальную арифметику – MPFI и др., при этом значительно увеличиваются объем требуемой памяти и время вычислений (на порядок и более по сравнению с функциями стандартной библиотеки). Компиляторы gcc / g++ и icc поддерживают представление чисел в формате Binary128 (Decimal128), но в настоящий момент времени не обеспечивают средствами работы с ними. 2. Увеличение быстродействия библиотек типа MPFR должно базироваться на особенности формул, лежащих в их основе, которые должны поддаваться распараллеливанию и использованию SIMD – расширений процессора

Intel. При этом требуется как разработка соответствующих алгоритмов, так и поддержка со стороны компилятора языка программирования.

Перспективами дальнейших исследований является разработка более эффективных алгоритмов вычисления значения специальных функций, таких как erf(), с произвольной точностью и увеличенным быстродействием, на основе использования алгоритмов параллельных и многопоточных вычислений, с целью повышения точности и ускорения вычислений при моделировании сложных технических систем.

Список литературы: 1. *Мазманишвили А.С.* Визуализация информационных характеристик электромагнитной обстановкив системах спутниковой связи / *А.С. Мазманишвили, О.Я. Никонов* // Вісник СумДУ. Серія "Технічні науки". – 2008. – № 4. – С. 30-37. **2.** IEEE Standard for Floating-Point Arithmetic. – New York, 2008.– 70 P. **3.** IEEE Standard for Binary Floating-Point Arithmetic. – New York, 1985. – 23 P. **4.** Information Technology – Extension for the programming language C++ to support decimal floating-point arithmetic [Электронный ресурс]. – Режим доступа: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2009/n2849.pdf. **5.** Справочник по специальным функциям с формулами, графиками и математическими таблицами / [*М. Абрамовщ и др*.]. – М. : Наука, 1979. – 832 с. **6.** *Cody W.J.* Rational Chebyshev approximations for the error function / *W.J. Cody* // Math. Comp. – 1969. – No.23. – P. 631-637. **7.** SUBROUTINE CALERF(ARG,RESULT,JINT) [Электронный ресурс] / Режим доступа : http://www.netlib.org/specfun/erf. **8.** *Chevillard S.* Computation of the error function erf in arbitrary precision with correct rounding / *S. Chevillard, N. Revol.* – Proc. 8th Conference on Real Numbers and Computers, July 2008. – P. 27-36.

УДК 519.651:004.222.3

Оцінка точності обчислень спеціальних функцій при розробці комп'ютерних програм математичного моделювання / Ніконов О.Я., Мнушка О.В., Савченко В.М. // Вісник НТУ "ХПІ". Тематичний випуск: Інформатика і моделювання. – Харків: НТУ "ХПІ". – 2011. – № 17. – С. 115 – 121.

Проведений аналіз формату дійсних чисел IEEE-754 як складової додаткової похибки при обчисленні спеціальних функцій. Проведено порівняння й показано, що застосування арифметики з довільною точністю (mpfr_erf(), MPFR) і стандартної функції (erf(), C99) у програмах на C/C++ знижує швидкодію на порядок при тій же точності. Іл.: 2. Бібліогр.: 8 назв.

Ключові слова: формат IEEE-754, довільна точність, точність обчислень спеціальних фунцій.

UDC 519.651:004.222.3

Estimation of accuracy of special functions computation in the development of mathematical simulation software / Nikonov O.Ya., Mnushka O., Savchenko V.N. // Herald of the National Technical University "KhPI". Subject issue: Information Science and Modelling. – Kharkov: NTU "KhPI". – 2011. – №. 17. – P. 115 – 121.

Format reals in IEEE-754 as a component of the additional error in the calculation of special functions have been analysed. A comparison is shown that the use of arithmetic with arbitrary precision (mpfr_erf (), MPFR) and the standard function (erf (), C99) in programs for C / C + + reduces the speed of the order with the same accuracy. Figs.: 2. Refs.: 8 titles.

Keywords: IEEE-754 format, an arbitrary precision, accuracy of special functions computation.

Поступила в редакцию 04.02.2011