# Верификация программных систем геопространственного обеспечения ВМФ

## Verification of program systems of geospatial maintenance of the navy

**Ключевые слова:** точные вычисления – exact calculations.

Приводится обоснование необходимости функциональной верификации программных систем навигационно-гидрографического и гидрометеорологического обеспечения (НГГМО).

The substantiation of necessity of functional verification of program systems is resulted.

**ЖУКОВ / ZHUKOV Y.**
**Юрий Николаевич**
(zhukov50@mail.ru)
доктор технических наук, старший научный сотрудник
ОАО «Государственный научно-исследовательский навигационно-гидрографический институт»,
Санкт-Петербург

Широкое внедрение ЭВМ в систему навигационно-гидрографического обеспечения ВМФ и судовождения предполагает повышение эффективности решения практических задач флота и безопасности судовождения. В настоящее время рынок программного обеспечения предлагает широкий спектр судовых навигационных, геопространственных и гидрографических автоматизированных систем. Эти системы решают различный набор задач, определенный в спецификации на конкретную программную систему. Потребителю представляется техническая документация, в которой указаны спецификации, однако отсутствует документ независимой экспертизы, подтверждающей, что программная система действительно функционирует согласно своей спецификации. Наличие в работе программных средств навигационно-гидрографического и гидрометеорологического обеспечения (НГГМО) несоответствий спецификации (по осознанным или неосознанным причинам) представляет чрезвычайную опасность для военной безопасности страны и, в частности, для мореплавания.

В военной области ошибки часто приводят к трагическим последствиям. В 70–80-х годах прошлого века мир был на грани ядерной войны из-за ошибок советской и американской систем раннего предупреждения. В 1991 году во время войны в Заливе (США против Ирака) батарея американских зенитных ракет «Пэтриот» не смогла перехватить запущенную иракцами ракету «Scud» советского производства. Ракета попала в казарму американских солдат, при этом погибло 28 человек. Причиной была погрешность вычисления времени. При вычислении необходимо было умножать время, задаваемое тактовым генератором компьютера (оно измерялось в десятых долях секунды), на 1/10, но это десятичное число невозможно точно представить в двоичном виде. Для хранения данной константы использовался 24-разрядный регистр. Разница между точным значением 1/10 и ее неточным двоичным представлением составляет в двоичном виде 0,0000000000 0000000000000011001100... или около 0,000000095 в десятичном виде. Компьютер был включен около 100 часов, за это время накопилась ошибка в измерении времени в 0,34 с. Скорость ракеты «Scud» составляла примерно 1700 м/с, т.е. за это время она прошла более 500 м, и этого хватило, чтобы зенитные ракеты не смогли ее перехватить [1].

Одним из основных источников нарушения функциональных спецификаций программных систем является отсутствие при их программировании учета специфических особенностей арифметических вычислений на ЭВМ. Широко распространено мнение, что ЭВМ выполняет арифметические операции приближенно, с некоторой известной погрешностью (приближенные вычисления – давний и почтенный раздел математики [2]). Это опасное заблуждение. В действительности арифметические операции на ЭВМ выполняются с ошибкой.

В частности, для формата чисел с плавающей точкой арифметические операции выполняются всегда ошибочно. В подавляющем большинстве существующих программных систем НГГМО используется формат плавающей точки для хранения исходных данных и при выполнении

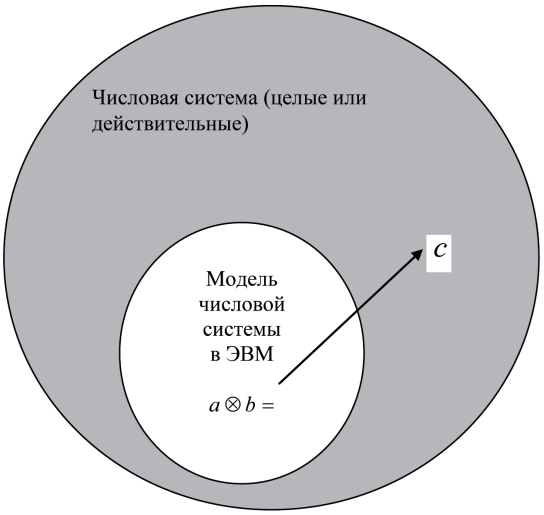


**Рис. 1.** Результат произведения двух чисел может выходить за множество чисел, поддерживаемое в конкретной ЭВМ

арифметических операций. Это обосновывается тем, что при решении научно-технических задач более удобно представление чисел с плавающей запятой, поскольку в них, как правило, приходится иметь дело с числовыми данными из очень широкого диапазона. Однако, несмотря на то, что каждая из отдельных ошибок округления очень мала, в совокупности они могут оказывать значительное влияние на точность результата, получаемого в процессе реализации алгоритма, особенно если число выполняемых операций достаточно велико. Более того, из-за ошибок округления некоторые алгоритмы просто нельзя реализовать. Поэтому при решении серьезных задач оценке влияния ошибок округления уделяется особое внимание.

Как техническое устройство, ЭВМ хранит, обрабатывает и визуализирует информацию только в дискретном виде. Это принципиальное условие. Причина ошибочности вычислений проста – технические особенности конструкции ЭВМ таковы, что не позволяют непосредственно поддерживать ни одной числовой системы: ни системы целых, ни рациональных, ни действительных чисел. Конечное число разрядности процессора приводит к тому, что не выполняется требование замкнутости арифметических операций ни для одной числовой системы (рис. 1). Действительные числа моделируются на ЭВМ форматом плавающей точки. Для этого формата на ЭВМ существует наименьшее (рис. 2) и наибольшее число, что противоречит свойствам системы действительных чисел. Дело в том, что в ЭВМ для приближения вещественных чисел используется всего лишь конечное множество чисел с плавающей точкой. Множество чисел $F$ с плавающей точкой характеризуется основанием счисления $\beta$, точностью $t$ и областью значений экспоненты $[L,U]$, где параметры $\beta$, $t$, $L$, $U$ явным образом зависят от компьютера. Каждое число с плавающей точкой $f$ из множества $F$ может быть представлено в виде

$$f = \pm\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t}\right)\beta^e,$$

где целые числа $d_i, i=1,2,\ldots,t$, удовлетворяют неравенствам $0 \le d_i \le \beta 1$ и $L \le e \le U$; если мы потребуем, чтобы $d_1 \neq 0$ для всех $f \in F, f \neq 0$, будем иметь дело с нормализованными числами с плавающей точкой.

Проблемы, возникающие из такого приближения вещественных чисел числами с плавающей точкой, состоят в следующем. Прежде всего, множество $F$ не является непрерывным или даже бесконечным множеством. Во множестве $F$ существует в точности $2(\beta-1)\beta^{t-1}(U-L=1)=1$ нормализованных чисел с плавающей точкой (включая нуль). Более того, эти числа распределены равномерно не на всей области значений, а только между последовательными степенями $\beta$.

Из сказанного следует, что сумма (или произведение) данных чисел $f_1$ и $f_2$ из множества $F$ не может не принадлежать $F$ и должна быть приближена числом с плавающей точкой. Разность между истинным и приближенным значением суммы (или произведения) является ошибкой округления. Следует также отметить, что операции сложения и умножения в $F$ не являются ассоциативными и закон дистри-


**Рис. 2.** Положение числовых точек на числовой оси вблизи нуля для числовой модели действительных чисел в 7-битном стандарте IEEE для чисел с плавающей точкой

бутивности не выполняется. Ошибки округления встречаются не только при использовании чисел с плавающей точкой. Они могут возникнуть и при работе с целыми числами, например – в случае, когда мы хотим вычислить произведение двух *s*-значных чисел в компьютере, который не может обрабатывать числа, содержащие больше *s* цифр.

При проектировании программной системы обычно предполагается, что все арифметические преобразования будут точно выполнены на ЭВМ. Однако в ЭВМ на представление любого числа отводится только конечное, строго фиксированное число разрядов, поэтому после выполнения каждой операции результат «обрезается» до нужной длины. Эта процедура вносит в результат ошибку, которая называется ошибкой округления. Сами по себе ошибки округления отдельных операций очень малы. Малы настолько, что часто возникает соблазн не учитывать их влияние на общий результат. К этому подталкивает и то обстоятельство, что во всех языках программирования любые формульные выражения записываются как математические без какого-либо указания на наличие ошибок округления. Более того, отметим как факт, что ни одна используемая на практике стандартная программная среда не имеет инструментальных средств для контроля за распространением этих ошибок. А это распространение происходит. Важнейшим фактором, объясняющим влияние ошибок округления на окончательный результат, является радикальное изменение свойств арифметических операций. Именно на множестве чисел, представленных в компьютере в форме с плавающей запятой, все операции перестают обладать свойствами коммутативности, ассоциативности и дистрибутивности. Аналогичная потеря свойств происходит и для чисел с фиксированной запятой, но в несколько меньшей мере. Следует исключительно важный вывод: записывая в программах математически эквивалентные выражения, мы не должны поддаваться иллюзии, что эти программы будут давать на компьютере хотя бы похожие результаты. Известно, что даже такая простая операция, как перестановка слагаемых в суммах чисел может привести из-за ошибок округления к катастрофически большим различиям. Имеется много других задач, аспектов и вопросов, связанных с заменой одних формульных выражений другими, математически эквивалентными. Обратим внимание на следующие моменты. Во-первых, сфера замен формульных выражений исключительно обширна. Во-вторых, в теории и практике осуществления замен остается очень много белых пятен. И наконец, даже если замены делаются математически эквивалентными, это еще не гарантирует, что на практике мы не встретимся с большими неожиданностями. Учет указанных арифметических свойств вычислений на ЭВМ имеет особую значимость при разработках автоматизированных навигационных систем. Это напрямую связано с безопасностью мореплавания и эффективностью выполнения боевых задач ВМФ. Тем более что основные задачи таких систем лежат в области решения картометрических и геометрических задач. К такому типу задач относятся, например, задача нахождения точки на акватории моря или суши, принадлежности точки прямой и т.д. Такой тип задач особенно чувствителен к арифметическим ошибкам. Более того, в формате с плавающей точкой указанные задачи не вычислимы. Это значит, что встречаются ситуации, когда нельзя получить результат вычисления в формате с плавающей точкой. Несмотря на то, что эти ситуации довольно редки, их появление при решении боевой задачи приведет к трагическим последствиям.

Приведем некоторые примеры потери вычислительной устойчивости в алгоритмах решения задач вычислительной геометрии, которые лежат в основе всех алгоритмов ГИС.

### Задача проверки совпадения двух заданных точек

Эта проблема является особенно актуальной в случае использования вещественной арифметики с плавающей точкой. Как известно, сравнение плавающих вещественных чисел на равенство производится всегда с заданной точностью $\varepsilon$. Здесь определяющим является выбор значения $\varepsilon$. Несмотря на кажущуюся простоту, данная проблема имеет далеко идущие последствия. Как известно, в силу своей ограниченной точности обычные вещественные вычисления на компьютерах не обладают многими свойствами истинно вещественных чисел. Например, если мы используем числа, хранящиеся в памяти компьютера с точностью до 3 значащих цифр, то результат вычисления следующих выражений может не совпадать, т.е. нарушается свойство ассоциативности:

$$(100 + 0{,}5) + 0{,}5 \cong 100 \neq 101 \cong 100 + (0{,}5 + 0{,}5) \cdot$$

### Задача проверки взаимного расположения двух точек относительно прямой, проходящей через две заданные точки

Данная задача обычно очень просто решается методами аналитической геометрии. Записываем уравнение прямой, проходящей через две заданные точки – $(x_1, y_1)$ и $(x_2, y_2)$:

$$(x_1 - x)(y_2 - y) - (x_2 - x)(y_1 - y) = 0 \cdot$$

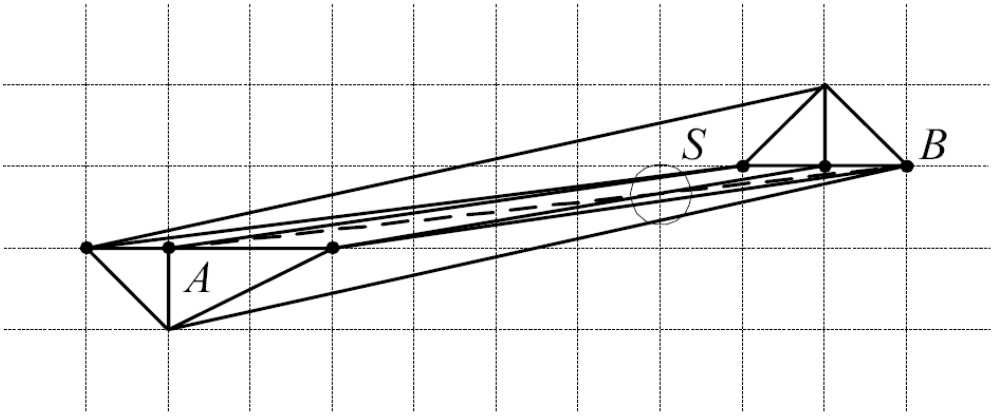


**Рис. 3.** Пример возможного «выворачивания» треугольников

Затем подставляем в это уравнение вместо $x$ и $y$ координаты тестовых точек($x_3$, $y_3$) и ($x_4$, $y_4$). Если значения выражений будут иметь одинаковый знак, то точки находятся по одну сторону от прямой, в противном случае – по разную. Результат выражения, равный нулю, будет означать попадание точки строго на прямую. Проблема заключается в потере точности промежуточных вычислений. Перемножая два $n$-значных числа, вообще говоря, получаем $2n$-значное число. На практике это обычно не учитывается, младшие разряды просто отбрасываются. Результат вычислений может показать, что тестовая точка лежит на прямой, хотя это не так. Для избавления от данного эффекта проводят сравнение с нулем с некоторой точностью. Несмотря на это, реальная точность вычислений все равно уменьшается в 2 раза, составляя не более $n/2$ исходных значащих цифр.

### Задача проверки порядка обхода трех заданных точек

Здесь требуется определить, в каком заданном порядке обходятся точки: по часовой стрелке или против? Эту задачу также решаем, записывая уравнение прямой, проходящей через две заданные точки, и подставляя в уравнение координаты третьей точки. После этого анализируем знак выражения. Таким образом, если:

$$(x_1 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_3) < 0,$$

точки обходятся по часовой стрелке, а если это выражение больше нуля – против (это верно для левосторонней системы координат, для правосторонней системы все будет наоборот). В данном алгоритме возникает та же самая проблема переполнения, что и при решении предыдущих задач.

### Задача поиска точки пересечения двух прямых

В данном случае первой проблемой является то, что определяемая точка в силу ограниченности точности вычислений в большинстве случаев не лежит ни на одном из ребер (ни на ранее существовавшем, ни на новых). Возможно, что эта точка лежит даже не в смежных с ребром треугольниках, поэтому в результате разбиения старого ребра на части образуются новые «вывернутые» треугольники, разрушающие структуру триангуляции. На рисунке 3 дан пример вставки ребра $AB$ в триангуляцию, приводящей к пересечению с существующим ребром в точке $S$ (пунктирными линиями размечена дискретная координатная сетка). В результате округления точка пересечения окажется немного выше реальной – в узле дискретной координатной сетки.

Несмотря на кажущуюся экзотичность, вероятность приведенного сценария возникновения ошибки велика уже при попытке вставить в триангуляцию нескольких десятков взаимно пересекающихся структурных ребер. Вдоль структурных ребер образуются многочисленные узкие вытянутые треугольники, которые и создают указанную критическую ситуацию.

Вторая проблема в этой задаче связана непосредственно с самим способом вычислений. Здесь сложности возникают при нахождении точки пересечения двух «почти» коллинеарных отрезков. В результате потери точности возможно значительное смещение найденных координат от реального значения. Кроме того, из-за потерь точности мы можем предположить, что отрезки пересекаются, хотя это не так. Тогда попытка вычисления их пересечения может привести к значительному удалению найденной точки

от самих отрезков (для «почти» коллинеарных отрезков).

Таким образом, большинство проблем связано с потерей точности внутренних вычислений. Контролировать точность, используя стандартные вещественные типы данных, предлагаемые большинством распространенных языков программирования, весьма сложно. Почти все современные компьютеры поддерживают стандарты ANSI представления вещественных чисел, однако даже 10-байтовый тип extended позволяет хранить не более 20 значащих цифр. В то же время при описании одной из задач показано, что для корректных вычислений требуется *4n*-значная арифметика. Это означает, что реальная достижимая точность построения триангуляции Делоне составляет не более 20/4 = 5 знаков в задании координат исходных данных, т.е. значение точности для проверки совпадения двух точек, возникшее при описании первой задачи, следует установить не менее $10^{-5}$, что не всегда приемлемо на практике. Другой способ заключается в использовании в явном виде вычислений с фиксированной точкой. В этом случае можно точно контролировать все потери точности. Еще более простым является переход к целочисленному представлению координат исходных точек. Например, используя обычные 32-битные целые числа, можно обеспечить точность представления в 9 значащих цифр, что является уже приемлемым в большинстве ситуаций.

Учитывая важность обеспечения ВМФ геопространственной информацией на основе автоматизированных систем, необходим строгий контроль за реализованными алгоритмами в этих системах. Такой контроль осуществляется путем верификации проектов разработки автоматизированных систем. Верификация означает подтверждение того, что описание проекта полностью соответствует спецификации (техническому заданию) проектируемой системы. Спецификация – это документ, подробно перечисляющий условия, которым должна соответствовать изготовляемая или проектируемая система. Таким образом, процесс верификации является необходимым условием разработки надежных и точных автоматизированных систем обработки геопространственной информации.

**Литература**

1. *Грушин А.И.* Верификация в вычислительной технике // Потенциал. – 2007. – № 4 (28). – С. 31–37.

2. European Symposium on Algorithm (ESA) // Lecture Notes in Computer Science Springer Berlin. – Vol. 3133. – Heidelberg, 2007.