УДК 004.891.3

Гайденко О. С.,
(асистент кафедри «Автоматизація та комп'ютерно-інтегровані технології транспорту» Київського інституту залізничного транспорту Державного університету інфраструктури та технологій, м. Київ)
Голуб Г. М., к.т.н.,
(доцент кафедри «Автоматизація та комп'ютерно-інтегровані технології транспорту» Київського інституту залізничного транспорту Державного університету інфраструктури та технологій, м. Київ)
Галан О. В., к.т.н.,
(доцент кафедри «Автоматизація та комп'ютерно-інтегровані технології транспорту» Київського інституту залізничного транспорту Державного університету інфраструктури та технологій, м. Київ)

ДОСЛІДЖЕННЯ ПРОБЛЕМИ «КОМП'ЮТЕРНОЇ НЕТОЧНОСТІ» В АВТОМАТИЗАВАНИХ СИСТЕМАХ ТРАНСПОРТУ

Розглянуто причину виникнення обчислювальних помилок при проведенні розрахунків на комп'ютері. Проведено аналіз відомих спроб вирішення проблеми «комп'ютерної неточності», висвітлено їх недоліки. Проведено експериментальне дослідження типу даних BigInt у JavaScript, в результаті якого на конкретних прикладах показано проблеми, через які на сьогоднішній день немає остаточного рішення проблеми «комп'ютерної неточності».

Ключові слова: *програмування, програмні засоби, зберігання інформації, стандарт IEEE 754, обчислювальна помилка.*

Вступ. Сьогодні еволюція прикладних інформаційних систем поступово зміщується в сторону їх інтелектуалізації, на сучасному етапі розвитку інформаційного суспільства наявність різноманітного програмного забезпечення визначає ступінь і якість обробки даних та виконання на комп'ютері користувацьких завдань [1]. Значущим завданням при цьому залишається підвищення точності обчислення.

Розрахунки з похибкою призводять до накопичення проміжних обчислювальних помилок, які в певних задачах можуть повністю спотворити достовірність остаточних результатів [2].

Аналіз останніх досліджень і постановка проблеми. Питання достовірності та точності обчислень стає все більш актуальним, так як для більшості розрахунків, що використовуються сьогодні в автоматизованих технологічних процесах виникнення та накопичення похибок може привести до різного роду наслідків, зокрема і стати причиною появи аварійних режимів роботи [9-15]. Питання

DOI: 10.32703/2617-9040-2019-33-2-9

похибок у результатах при обчисленні комп'ютерною технікою порушено в роботах [2-6, 8]. Підсумовуючи проаналізовані джерела, розглянемо першопричину виникнення неточності, відомі шляхи її усунення, недоліки таких рішень та практичні способи обходу проблеми.

Мета роботи. Метою написання роботи є висвітлення причин виникнення проблеми «комп'ютерної неточності», тенденцій щодо її усунення та пошук практичних шляхів її обходу.

Наукова новизна отриманих результатів: запропоновано алгебраїчні методи послаблення впливу проблеми «комп'ютерної неточності», перевагою яких є широка доступність та простота застосування.

Матеріали та методи дослідження. В обчислювальних системах використовуються два формати зберігання чисел – цілі та числа з плаваючою комою.

Найчастіше використовується подання дійсних чисел у вигляді двійкових чисел з плаваючою комою, описане в стандарті IEEE 754 (IEEE Standard for Binary Floating-Point Arithmetic, ANSI / IEEE Std 754-1985). Цей найбільш поширений стандарт для обчислень з плаваючою комою, використовується багатьма сучасними апаратними та програмними засобами ЕОМ [2].

Даний формат представлення чисел з плаваючою комою двох видів: одинарної точності (32 біта) і подвійний точності (64 біта).

32-бітові контейнери, що відповідають стандарту IEEE 754 призначені для зберігання та виконання операцій над апроксимаціями дійсних чисел. 32 біта розділені на три частини (рис. 1) [7]:

- S (1 бит) для зберігання знака;
- E (8 бит) для експоненти;
- M (23 бита) для мантиси.

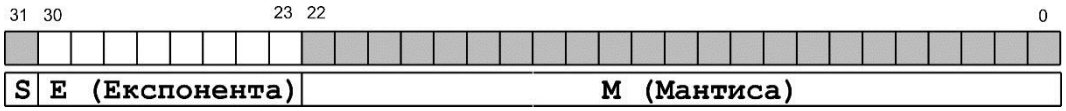

Рис.1. **Три частини числа з плаваючою комою одинарної точності**

Розглянемо причини виникнення помилок при проведенні обчислень з дробовими числами на комп'ютері. Навіть, якщо взяти стандарт IEEE 754 подвійної точності (double), у ньому на число виділяється рівно 8 байт (64 біта), не більше і не менше. З 64 біт, відведених на число, самі цифри числа займають до 52 біт (M), один біт зберігає знак (S), а решта 11 – позицію десяткового дробу (E). При цьому деякі десяткові дроби у двійковій системі числення є нескінченними дробами. Двійкове значення нескінченних дробів зберігається лише до певного знака, який залежить від обмежень пам'яті. Під час виконання коду десяткові значення чисел переводяться в їх двійкові еквіваленти і це є першопричиною виникнення обчислювальних помилок. Звідси береться неточність.

Проблема стосується популярних мов програмування, таких як Java, C, PHP, JavaScript, Ruby, Perl, так як усі вони працюють у стандарті IEEE 754. Дану проблему називають «комп'ютерною неточністю».

Які кроки у напрямку усунення «комп'ютерної неточності» було зроблено?

У 1991 році ЦП не мав апаратного пристрою для обчислень із плаваючою комою. Проте на ринку з'явилися апаратні модулі операцій з плаваючою комою («математичні співпроцесори»), в той час вони представляли інтерес лише для вчених і навіть з часом не знайшли широкого застосування.

З 2008 року у стандарті IEEE 754 з'явився новий 128–бітний формат розширеної точності з 15 бітною експонентою і 112 бітною мантисою. Очевидно, що його застосування позитивно впливає на точність обчислювальних процесів, проте не є остаточним вирішенням проблеми. Крім того не у всіх мовах програмування, які працюють у стандарті IEEE 754, реалізовано підтримку формату розширеної точності. Збільшення розміру контейнера має суттєвий негативний вплив на швидкодію обчислювальних процесів.

Набувають популярності спеціалізовані бібліотеки, в яких реалізовано точні обчислення. Механізм функціонування яких, оснований на здійсненні певних перетворень двійкового подання десяткових чисел до проведення арифметичних операцій.

Найбільш ефективним рішенням в плані масового розповсюдження за весь час стало впровадження в мовах програмування додаткових числових типів даних.

Так, наприклад, одним із кроків по усуненні проблеми «неточності», зроблених розробниками ECMAScript стало введення (поки що чорнового варіанту) в 2018 році нового для JavaScript числового типу даних «BigInt», який дозволяє без помилок працювати з цілими числами, кількість цифер яких перевищує 253. Межі для безпечної роботи з цілими числами не застосовні до BigInt, тому з BigInt ми можемо застосовувати довгу арифметику не турбуючись про втрату точності.

Проте такі впровадження ведуть за собою до ряду проблем, які не можливо вирішити не втрутившись в усталений механізм функціонування мови програмування, що в свою чергу не можна зробити, так як це призведе до втрати працездатності

Проведемо **експериментальні дослідження** в JavaScript.

```
var a = BigInt (158);
alert (+a);
```

Такий код видає помилку «Uncaught TypeError: Cannot convert a BigInt value to a number». Унарний «+» не підтримується в BigInt, тому що він порушить код asm.js, який передбачає, що +a завжди буде повертати або Number, або виключення. При цьому унарний «–» працює як і з Number:

```
var a = BigInt (158);
alert (-a);
```

Код виведе «–158».

Проведемо арифметичну операцію з BigInt та Number:

```
var a = BigInt (158);
alert (128  + a);
```

У результаті отримаємо «Uncaught TypeError: Cannot mix BigInt and other types, use explicit conversions». Операції з BigInt та Number проводити не можна за винятком операторів порівняння:

```
128 + 158n; // → TypeError
128 < 158n; // → true
```

Записавши в BigInt дробове число, отримаємо помилку:

```
BigInt(1.5); // → RangeError
```

Зважаючи на особливості застосування BigInt можна припустити, що передбачається його використання на вибір із звичним типом Number у тих нішевих випадках, де він необхідний. Отже, BigInt вирішує проблему обмеження запису чисел, проте залишає відкритою проблему втрати точності для дробових чисел, яка має ту ж причину виникнення, до того ж має більше розповсюдження, так як числа довжиною більші за допустимі в типі Number застосовуються лиш у певних галузях, а проблема з точністю чисел із плаваючою комою може стосується кожного розробника, наприклад:

```
alert (0.1  + 0.2); // 0.30000000000000004
```

Як видно на 17 порядку числа у прикладі з'явилась похибка. Вона надзвичайно низька і на практиці рідко коли має вагоме значення. Проте, якщо робота програми потребує дотримання заданого рівня точності обчислень, проблему «комп'ютерної неточності» можна пом'якшити за допомогою написання функції округлення, яку реалізовано на основі вбудованого математичного методу Math.trunc(), виклик якого повертає цілу частину числа, повністю відкидаючи дробову:

```
function truncated(num){
    return Math.trunc(num * 1000) / 1000;
}
```

Перемножуючи аргумент «num» на 1000 до застосування методу Math.trunc() та ділячи після застосування, ми зберігаємо йому три знаки після коми. Таким методом, застосовуючи інші методи округлення та множник, можна задавати й інші рівні точності.

Якщо на практиці при обчисленнях чисел з плаваючою комою проблему неточності можна пом'якшити округленням, то небезпечною стороною залишається застосування таких чисел як аргументів умови, наприклад:

```
for (var i = 0.1; i != 1; i += 0.1){ alert (i); }
```

Такий цикл зациклиться. Тому програмісту залишається обдумано писати умови й уникати чисел в якості аргументів.

Висновки. У результаті аналізу відомих спроб вирішення проблеми «комп'ютерної неточності» можна зробити висновок, що головним їх недоліком є відсутність розповсюдженості того чи іншого рішення. Причиною низького рівня розповсюдження існуючих рішень проблеми насамперед є необхідність додаткових фінансових витрат, якщо говорити про апаратні рішення та суттєве зниження продуктивності, якщо мова йде про програмні. При цьому обидва варіанти є нераціональними для пересічного користувача. Проведено експериментальне дослідження типу даних BigInt у JavaScript, в результаті якого на конкретних прикладах показано чому на сьогоднішній день немає остаточного вирішення

проблеми «комп'ютерної неточності». Наведено доступні практичні рішення проблеми, які на сьогоднішній день застосовуються в програмуванні.

ЛІТЕРАТУРА

1. *Шаров С.В.* Сучасний стан розвитку інтелектуальних інформаційних систем // Вісник Чернігівського національного педагогічного університету. Серія: Педагогічні науки. 2015. Вип. 130. С. 111–114.

2. *Жульковская И.И.* Жульковский О.А., Шаганенко Р.Г. Вычисление граничных значений субнормальных чисел в IEEE-стандарте // Мат. мод. № 1 (32). 2015. С.41–44.

3. *Петров Ю.П.* Обеспечение надежности и достоверности компьютерных расчетов // СПб: БХВ–Петербург, 2008. 160 с.

4. *Brisebarre N.*, Dinechin F., Jeannerod C., Lefevre V., Melquiond G., Muller J., Revol N., Stehle D., Torres S. Floating-Point Arithmetic // Laboratoire LIP, Projet Arenaire CNRS, INRIA, Ecole Normale Superieure de Lyon. France, 2009. 16 p.

5. *Loh E.*, Walster G. Rump's Example Revisited // Reliable Computing 8. 2002. p. 245–248.

6. *Аноприенко А.Я.*, Иваница С.В. Тетралогика, тетравычисления и ноокомпьютинг. Исследования 2010–2012 // Донецк: ДонНТУ, Технопарк ДонНТУ УНИТЕХ. 2012. 308 с.

7. *Fabien Sanglard* Game Engine Black Book: Wolfenstein 3D // Software Wizards. 2017. 315 p.

8. *Stasiuk A.I.*, Goncharova L.L., Golub G.M. Method for assessing cybersecurity of distributed computer networks for control of electricity consumption of power supply distances // Journal of Automation and Information Sciences. 2017. V.49, Issue 7. PP. 48–57.

9. *Кульбовський І.І.,* Голуб Г.М., Гайденко О.С. Моделювання інформаційних потоків комп'ютерного моніторингу мереж електропостачання транспорту // Металургическая и горнорудная промышленость. 2018. 4 (313). С. 94–98.

10. *Golub G.M.* Reliability control of failure-free operation of power supply system of railroad and its components by methods of intellectualization and informatization. // Metallurgical and Mining Industry. 2017. № 5. С. 8–13.

11. *Стасюк О.І.,* Гончарова Л.Л., Максимчук В.Ф., Голуб Г.М. Методи комп'ютерної інтелектуалізації режимів функціонування тягових мереж залізниць // Інформаційно-керуючі системи на залізничному транспорті. 2013. № 5. С. 29–35.

12. *Кульбовський І. І.,* Голуб Г. М. Аналіз нормативно-технічної бази впровадження інтелектуальних енергетичних систем на основі технологій Smart Grid // Інформаційно-керуючі системи на залізничному транспорті. 2016. № 3. С. 50–57.

13. *Opanasenko V.N.*, Kryvyi S.L. Partitioning the full range of boolean functions based on the threshold and threshold relation // Cybernetics and Systems Analysis. 2012. Vol. 48, No.3. PP. 459 – 468.

14. *Opanasenko V.N.*, Kryvyi S.L. Synthesis of Adaptive Logical Networks on the Basis of Zhegalkin Polynomials // Cybernetics and Systems Analysis. 2015. Vol. 51, No 6. PP. 969–977.

15. *Стасюк О.І.,* Гончарова Л.Л. Математичні моделі і методи комп'ютерного керування електропостачанням залізниць на основі диференційних перетворень Пухова // Электронное моделирование. 2016. Т.38, №4. С. 127–139.

REFERENCES

1. Sharov S.V. (2015). *Suchasnuy stan rozvytku intelectualnih informatsiynyh system* [*Pedahohichni nauky]*. Visnyk Chernigivskoho natsionalnoho pedahohichnoho universytetu, 130, 111-114.

2. Zhulkovskaya I.I., Zhulkovskiy O.A, Shaganenko R.G. (2015). *Vychislenie granichnyh znacheniy subnormalnyh chisel v IEEE-standarte* (rus) [Mat. mod] 1(32), 41-44.

3. Petrov U.P. (2008). Obespechenie nadegnosti I dostovernosti komputernyh raschetov. (rus) SPb:BHV-Peterburg, 160.

4. Brisebarre N., Dinechin F., Jeannerod C., Lefevre V., Melquiond G., Muller J., Revol N., Stehle D., Torres S. (2009). *Floating-Point Arifmatic.* Laboratoire LIP, Projet Arenaire CNRS, INRIA, Ecole Normale Superieure de Lyon, France. 16. (in English)

5. Loh E., Walster G. (2002) *Rump's Example Revisited*. Reliable Computing 8, 245-248. (in English)

6. Anoprienko A.Ya., Ivanitsa S.V. (2012) *Tetralogika, tetravychisleniya i nookomputing*. [*Issledovaniya 2010-2012*].(rus) Donetsk: DonNTU,Technopark DonNTU UNITEX, 308.

7. Fabien Sanglard (2017) *Game Engine Black Book: Wolfenstein 3D* [*Software Wizards*], 315. (in English)

8. Stasiuk A.I., Goncharova L.L., Golub G.M. (2017) *Method for assessing cybersecurity of distributed computer networks for control of electricity consumption of power supply distances.* Journal of Automation and Information Sciences. V.49, Issue 7, 48-57. (in English)

9. Kulbovskiy I.I., Holub H.M., Haydenko O.S. (2018) *Modeluvannya informatsiynyh potokiv komputernoho monitorynhu merezh electropostachannya transportu.* Metalurgicheskaya I gornorudnaya promyshlennost 4 (313), 94–98.

10. Golub G. M. (2017) *Reliability control of failure-free operation of power supply system of railroad and its components by methods of intellectualization and informatization.* Metallurgical and Mining Industry 5, 8 – 13. (in English)

11. Stasyuk O.I. Honcharova L.L., Maksymchuk V.F., Holub H.M. (2013) Metody komputernoyi intelectualizatsiyi rezhymiv funktsionuvannya tyahovyuh merezh zaliznyts. Informatsiyno-keruyuchi systemy na zaliznychnomu transporti, 5, 29-35.

12. Kulbovskiy I.I., Holub H.M. (2016) Analiz normatuvno-tehnichnoyi bazy vprovadzhennya intelektualnyh enerhetychnyh system na osnovi tehnolohiy Smart Grid. Informatsiyno-keruyuchi systemy na zaliznychnomu transporti, 3, 50-57.

13. Opanasenko V.N., Kryvyi S.L. (2012) *Partitioning the full range of boolean functions based on the threshold and threshold relation.* Cybernetics and Systems Analysis. 48/3, 459–468. (In English)

14. Opanasenko V.N., Kryvyi S.L. (2015) *Synthesis of Adaptive Logical Networks on the Basis of Zhegalkin Polynomials.*Cybernetics and Systems Analysis. 51/6, 969–977. (in English)

15. Stasyuk O.I., Honcharova L.L. (2016) Matematyuchni modeli ta metody komputernoho keruvannya electropostachannyam zaliznyuts na osnovi dyferentsiynyh peretvoren Puhova. Electronnoe modelirovanie, 38/4, 127-139.

Гайденко О. С.,

(асистент кафедры «Автоматизация и компьютерно-интегрированные технологии транспорта» Киевского института железнодорожного транспорта Государственного университета инфраструктуры и технологий, г. Киев)

Голуб Г. Н., к.т.н.,

(доцент кафедры «Автоматизация и компьютерно-интегрированные технологии транспорта» Киевского института железнодорожного транспорта Государственного университета инфраструктуры и технологий, г. Киев)

Галан О. В., к.т.н., доц.,

(доцент кафедры «Автоматизация и компьютерно-интегрированные технологии транспорта» Киевского института железнодорожного транспорта Государственного университета инфраструктуры и технологий, г. Киев)

ИССЛЕДОВАНИЯ ПРОБЛЕМЫ «КОМПЬЮТЕРНОЙ НЕТОЧНОСТИ» В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ ТРАНСПОРТА

Рассмотрены причины возникновения вычислительных ошибок при проведении расчетов на компьютере. Проведен анализ известных попыток решения проблемы «компьютерной неточности», освещены их недостатки. Проведено экспериментальное исследование типа данных BigInt в JavaScript, в результате которого на конкретных примерах показано проблемы, через которые на сегодняшний день нет окончательного решения проблемы «компьютерной неточности».

Ключевые слова: *программирования, программные средства, хранения информации, стандарт IEEE 754, вычислительная ошибка.*

Oles Haidenko
(Assistant of department «Automation and Computer-Integrated Technologies of Transport department», State University for Infrastructure and Technologies)
Halyna Holub, PhD
(Associate Professor of department «Automation and Computer-Integrated Technologies of Transport department», State University for Infrastructure and Technologies)
Olha Halan, PhD
(Associate Professor of department «Automation and Computer-Integrated Technologies of Transport department», State University for Infrastructure and Technologies)

STUDY OF THE PROBLEM OF «COMPUTER INACCURACY» IN AUTOMATIC TRANSPORT SYSTEMS

The question of reliability and accuracy of calculations has been considered, which becomes urgent, since for most calculations used today in automated technological processes of occurrence and accumulation of errors can lead to various kinds of consequences, in particular, to cause the occurrence of emergency modes of work. Also, there have been considered the reason for the calculation errors occurring during computing on computer, which is related to popular programming languages such as Java, C, PHP, JavaScript, Ruby, Perl, working in the IEEE 754 standard, the trends of their elimination and the search for practical ways of bypassing them. The analysis of known attempts to solve the problem of "computer inaccuracy" has been carried out; as a result of which it is determined that their main disadvantage is the absence of the prevalence of a decision. An experimental study of the type of BigInt data in JavaScript has been carried out, which resulted in specific examples of problems that, due to this, there is no definitive solution to the problem of "computer inaccuracy". There have been depicted available practical solutions to the problems that are currently being used in programming.

Keywords*: programming, software, information storage, IEEE 754 standard, computational error.*