УДК 517.9

ПРОБЛЕМИ ТОЧНОСТІ КОМП'ЮТЕРНИХ ОБРАХУНКІВ

О. Ю. Мудренко, О. С. Вєтров

Анотація. Проблема точності комп'ютерних обчислень є однією з фундаментальних проблем Computer Science, оскільки неврахування межі достовірність комп'ютерних обрахунків може призвести до непередбачуваних результатів. Фізичні обмеження на обсяг пам'яті, що виділяється комп'ютерною технікою для обробки числа, накладає принципові обмеження на можливості та логіку організації комп'ютерних обрахунків, що також потребує залучення специфічних математичних методів дослідження.

Ключові слова: числа з плаваючою комою, стандарт IEEE-754, точність комп'ютерних обрахунків.

Вступ. Арифметика з плаваючою комою сьогодні є найпоширенішим способом апроксимації арифметики дійсних чисел для виконання числових обчислень на сучасних комп'ютерах. Загальне подання арифметики з плаваючою комою виглядає так: число x має вигляд добутку s, m та e^{β}, де s позначає знак числа, m – мантиса та e^{β} – показник степеня (експонента), тобто:

$$x = s \times m \times e^{\beta}.$$

Основний розділ. Пропонована реалізація арифметики з плаваючою комою має відповідати критеріям надійності, швидкодії та портативності. Перша сучасна реалізація концепції арифметики з плаваючою комою з'явилася в комп'ютері Конрада Цузе Z3 з тактовою частотою 5,33 Гц [1].

Ефективне представлення дійсних чисел та маніпулювання ними є необхідним у багатьох галузях науки, техніки, фінансів тощо. В історії було запроваджено багато різних способів наближеного представлення дійсних чисел на комп'ютерах. У [1] наводяться приклади подібних реалізацій: арифметика з фіксованою комою, логарифмічна та напівлогарифмічна системи числення, неперервні дроби, раціональні числа та нескінченні ряди раціональних чисел, системи числення з індексом рівня, системи числення з фіксованою та плаваючою косою рискою, 2-адичні числа тощо.

Моделювання нескінченної неперервної множини (дійсних чисел) за допомогою скінченної множини («машинних чисел») не є простим завданням: необхідно знайти розумні компроміси між ними, наприклад, швидкістю, точністю, динамічним діапазоном, простотою використання і реалізації та вартістю пам'яті. Виявляється, що арифметика з плаваючою комою, з адекватно підібраними параметрами (радіус, точність, екстремальні показники тощо) є дуже хорошим компромісом для більшості числових застосувань.

В обчисленнях арифметика з плаваючою комою – це комп'ютерно реалізована арифметика, яка представляє підмножини дійсних чисел за допомогою цілого числа з фіксованою точністю, що називається знаменником, масштабованого за допомогою цілої експоненти з фіксованою основою. Числа такої форми називаються числами з плаваючою комою.

Арифметичні операції з плаваючою комою, як-от додавання і ділення, наближають відповідні арифметичні операції з дійсними числами шляхом округлення будь-якого результату, який не є числом з плаваючою комою, до найближчого числа з плаваючою комою. Представлення числа визначає певний спосіб кодування числа, зазвичай у вигляді рядка цифр.

Існує декілька механізмів, за допомогою яких рядки цифр можуть представляти числа. У стандартній математичній системі числення рядок цифр може бути довільної довжини, а місцезнаходження розрядної точки вказується за допомогою спеціального символу «.» (у різних національно локалізованих нотаціях це може бути крапка або кома). Якщо розрядну точку не вказано, то рядок неявно представляє ціле число, а невстановлена радіусна точка буде знаходитися в правому кінці рядка, поруч із найменшою значимою цифрою. У системах числення з фіксованою комою для крапки-радіуса вказується позиція у рядку.

У роботі наведені деякі приклади некоректності комп'ютерних обчислень, які виникають через обмеження на обсяг пам'яті, що виділяється для роботи з числами з плаваючою комою. Дійсні числа, що є фундаментальним об'єктом математики, перетворюються на числа з плаваю-

чою комою згідно з наявними стандартами. Неврахування принципів двійкової арифметики (стандарт IEEE-754) може суттєво вплинути на коректність результатів під час розробки прикладних програм.

Розглянемо просту програму, реалізовану за допомогою мови програмування Python, що реалізує результат елементарних арифметичних обрахунків.

Лістинг 1

```
from numpy import float32
a = float32(0.9871)
b = float32(0.2781)
c = float32(1.2680)
print(a*c + b*c == (a+b) * c)
```

Очевидним результатом є True, але якщо ми змінимо в коді значення змінної a на 0,0001 (a = 0.9872), то результатом комп'ютерних обчислень буде вже False. Продовжуючи експеримент, при значенні c = 1.2681 результат виконання програми на лістингу 1 знову буде True.

Порушення очевидних нам правил роботи з дійсними числами продиктоване тим, що математичний об'єкт «дійсне число» і модель «floating point number» принципово розрізняються між собою, оскільки математичний аналіз трактує дійсне число як нескінченний об'єкт, а множину дійсних чисел (дійсну пряму) – виходячи з іншої абстракції неперервності, a floating point number – об'єкт за своєю суттю дискретний, а реальна комп'ютерна реалізація множини floating point numbers – множина скінчена, оскільки реально ми маємо можливість працювати лише з обмеженим обсягом виділеної пам'яті.

Обмежимось демонстрацією представлення чисел з плаваючою комою з описаного вище прикладу у форматах float та double (float32 та float64 з бібліотеки numpy в Python).

Формат float – під число виділяється 32 біти: старший біт знаковий (0 означає додатне число, 1 – число є від'ємним), далі 8 біт виділяється під експоненту, і останні 23 біти під нормалізовану мантису. Схематично це можна зобразити на рис. 1.

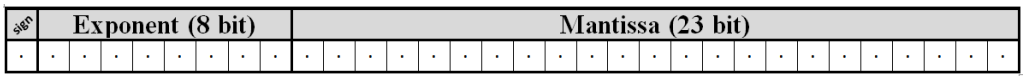

Рис. 1. Бінарне представлення числа типу float

Запишемо для прикладу числа з раніше розглянутого прикладу у бінарному коді:

$$(0.9871)_{10} = (00111111\ 01111100\ 10110010\ 10010110)_2,$$

$$(0.2781)_{10} = (00111110\ 10001110\ 01100011\ 00100000)_2,$$

$$(1.2680)_{10} = (00111111\ 10100010\ 01001101\ 11010011)_2.$$

Збільшуючи точність, можна розглянути і представлення числа типу double (рис. 2).

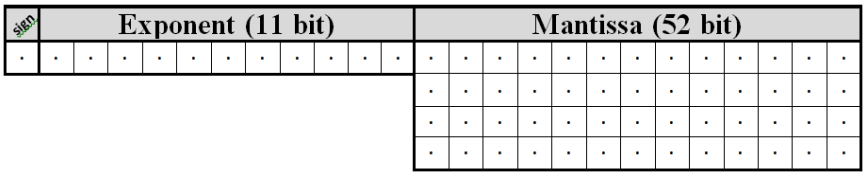

Рис. 2. Бінарне представлення числа типу double

Ті ж самі числа запишуться:

$$(0.9871)_{10} = (00111111\ 11101111\ 10010110\ 01010010\ 10111101\ 00111100\ 00110110\ 00010001)_2,$$

$$(0.2781)_{10} = (00111111\ 11010001\ 11001100\ 01100011\ 11110001\ 01000001\ 00100000\ 01011100)_2,$$

$$(1.2680)_{10} = (00111111\ 11110100\ 01001001\ 10111010\ 01011110\ 00110101\ 00111111\ 01111101)_2.$$

Оскільки відповідно до стандарту ми відсікаємо молодші біти, під час зворотної конвертації до десяткової системи числення отримаємо:

$$float: \quad (0.9871)_{10} \to (...)_2 \to (0.98710000514984130859375)_{10}$$

$$double: \quad (0.9871)_{10} \to (...)_2 \to (\ 0.98709999999999997744026813961 7)_{10}$$

Очевидно, результати різні.

Особливе місце поданню чисел з плаваючою комою у пам'яті комп'ютера відведено стандарту IEEE-754 (1985–1987, 2008, 2019). Головні переваги чисел із плаваючою крапкою, як відомо, полягають у тому, що вони дають змогу здійснювати обчислення у великому діапазоні значень, і водночас обчислення організовують інструментарієм двійкової арифметики, яку легко реалізувати на обчислювальному пристрої. Однак остання обставина приховує в собі певні неочевидні проблеми, які є причиною того, що розрахунки, зроблені з використанням цього формату, можуть призводити до некоректних результатів.

Стандарт IEEE-754 визначає всі фундаментальні правила та формати роботи з числами з плаваючою комою [2]:

- арифметичні формати: набори двійкових і десяткових даних з плаваючою комою, які складаються зі скінченних чисел (включно зі знаковими нулями і субнормальними числами), нескінченностей і спеціальних значень «не число» (NaNs);
- формати обміну: кодування (бітові рядки), які можна використовувати для обміну даними з плаваючою комою в ефективній та компактній формі;
- правила округлення: властивості, яких треба дотримуватися під час округлення чисел в арифметичних діях та перетвореннях;
- операції: арифметичні та інші операції (наприклад, тригонометричні функції) над арифметичними форматами;
- обробка виключних ситуацій: ознаки виключних ситуацій (наприклад, ділення на нуль, переповнення тощо);

Визначення арифметики з плаваючою комою (формати, механізми роботи операторів тощо) вимагає від нас пошуку компромісів між вимогами, які рідко бувають повністю сумісними. Серед різних властивостей, які є бажаними, можна навести такі:

- швидкість: обрахунки мають здійснюватися за осмислений людиною час;
- точність: навіть якщо швидкість важлива, отримати неправильний результат просто зараз гірше, ніж отримати правильний результат занадто пізно;
- діапазон значень, що можуть бути представлені: нам може знадобитися представляти як великі, так і малі числа. Дуже малі числа – проблема точності, дуже великі – проблема переповнення розрядної сітки (overflow problem);
- переносимість: реалізовані програми повинні працювати на різних машинах, не вимагаючи модифікацій;
- простота реалізації та використання: задана арифметика має бути прийнятною під час використання широкими класами користувачів.

Щодо точності, то найточніші сучасні фізичні вимірювання дають змогу перевірити деякі передбачення квантової механіки з відносною точністю до 10^{-15}. Це, звичайно, означає, що в деяких випадках ми повинні мати можливість представляти числові дані з подібною точністю (що легко зробити, використовуючи формати, які реалізовані майже на всіх сучасних платформах). Але це також означає, що іноді ми можемо проводити обчислення, які в підсумку повинні мати відносну похибку менше або дорівнює 10^{-15}, що набагато складніше.

У роботі були представлені результати дослідження автора, пов'язані з аналізом проблематики коректності комп'ютерних обчислень, їх стійкості, достовірності та збіжності. Продемонстровані відповідні приклади некоректної роботи програм: від елементарних до прикладів, що є предметом окремих досліджень точності обрахунків.

Робота є логічним продовженням досліджень [3, 4], в яких із математичного та технічного поглядів розглядались відомі приклади некоректності комп'ютерних обрахунків – приклад Рампа та послідовність Мюллера, і способи та алгоритми їх подолання.

Висновки. Окрім класичних математичних обґрунтувань, що є необхідними для розуміння та реалізації пропонованої теми, в роботі наведені специфічні програмні можливості контролю за точністю складних обчислень. Із цією метою були наведені приклади ефективного використання функціоналу модулів *decimal* та *fraction* у Python 3.x, один із можливих легко доступних способів зменшення проблеми неточності програмних обрахунків. Перспективним видається проведення складних обрахунків у Python із використанням специфічних бібліотек інтервальних обчислень, що набувають популярності останнім часом.

Abstract. The problem of the accuracy of computer calculations is one of the fundamental problems of Computer Science, since failure to take into account the limit of reliability of computer calculations can lead to unpredictable results. Physical limitations on the amount of memory allocated by computer hardware for number processing impose fundamental restrictions on the capabilities and logic of organizing computer calculations, which also requires the use of specific mathematical research methods.

Keywords: floating point numbers, IEEE-754 standard, accuracy of computer calculations.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Handbook of Floating-Point Arithmetic. Birkhäuser Basel / J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod etc. 2018. 627 p.

2. IEEE Standard for Floating-Point Arithmetic, (Revision of IEEE 754-2008). *IEEE Std 754-2019.* 2019. P. 1–84. DOI: 10.1109/IEEESTD.2019.8766229.

3. Vietrov O., Bilous R. Special methods of increasing the accuracy of computer calculations. *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*. 2022. P. 1–5. DOI: 10.1109/KhPIWeek57572.2022.9916383.

4. Vietrov O., Bilous R. Study of the Convergence of Muller's Sequence Computer Calculations. *2021 IEEE 3rd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*. 2021. P. 547–551. DOI: 10.1109/UKRCON53503.2021.9575546.

УДК 004.8:[004.774.6:7.012

НАЙКРАЩІ ШІ-ІНСТРУМЕНТИ ДЛЯ ВЕБДИЗАЙНЕРА

М. С. Назаренко, К. О. Якубич

Анотація. У статті розглядаються найкращі ШІ-інструменти, доступні для вебдизайнерів, і їх застосування в різних аспектах вебдизайну, яке охоплює використання штучного інтелекту для створення прототипів сайтів, наповнення вмісту, автоматизації рутинних завдань та аналізу даних. Кожен засіб детально описано, підкреслено його особливості та переваги. Використовуючи можливості штучного інтелекту, вебдизайнери можуть оптимізувати свій робочий процес, підвищити ефективність і результативність процесів проєктування.

Ключові слова: вебдизайн, штучний інтелект, аналіз даних, ШІ-інструменти.

Вступ. У сучасну цифрову епоху штучний інтелект (ШІ) став трансформаційною силою, вплив якої так чи інакше відображується у змінах індустрії, зокрема в галузі дизайну та послуг. Завдяки швидкому розвитку інтелектуальних технологій вебдизайнери можуть використовувати різноманітні інструменти на основі штучного інтелекту, які дають їм змогу створювати більш ефектні та креативні проєкти. Серед різноманіття дизайнерських сервісів, пропонованих на ринку, найбільшим питанням є особливість їх використання в тому чи іншому напрямі, а також відповідність цих ресурсів заявленим характеристикам.

Метою статті є проведення аналізу наявних серед продуктової лінійки інструментів штучного інтелекту для вебдизайну в різних напрямах, функціонал яких призначений для підвищення процесів прийняття рішень у середовищі проєктування вебсайтів та вебдодатків, а також виявлення їх особливостей та основних переваг щодо конкретних дизайнерських проєктів.

Інструменти, описані в цій статті, є лише верхівкою айсберга. Вони вказують на початок нової ери, коли дизайнери за допомогою штучного інтелекту будуть спроможні розширювати межі творчості та інновацій. Використання цих інструментів – це не просто можливість, це стане необхідністю для тих, хто прагне залишатися попереду в дизайні, що постійно розвивається.

Основний розділ. Незважаючи на застереження, що штучний інтелект може становити потенційну загрозу, в сучасному вебдизайні він пропонує безліч переваг, даючи змогу оптимі-