УДК 004.4:003.26:681.326.3

Трутнєв С.Г. (ВА м.Одеса)

DOI: http://doi.org/10.17721/2519-481X/2024/84-12

ПІДХОДИ ЩОДО ЗМЕНШЕННЯ ПОХИБОК ПРИ ОБЧИСЛЕННІ МАТЕМАТИЧНИХ МОДЕЛЕЙ БОЙОВИХ ДІЙ В КОМП'ЮТЕРНИХ СИСТЕМАХ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

Наукова стаття присвячена дослідженню та аналізу сучасних методів зменшення похибок у чисельних обчисленнях, що є критично важливими для підвищення точності та надійності імітаційних моделей бойових дій. Досліджено кілька підходів, які можуть бути застосовані для досягнення цієї мети.

Перший з розглянутих методів — це метод Рунге-Кутта четвертого порядку, який є одним з найпоширеніших методів для чисельного розв'язання диференційних рівнянь. Цей метод дозволяє отримати високу точність розв'язків при відносно невеликій кількості обчислень, що робить його ефективним для використання в реальному часі в комп'ютерних системах імітаційного моделювання.

Другий підхід, що розглядається в статті, — це використання алгоритму Кахана для точного сумування. Алгоритм Кахана дозволяє значно зменшити похибки, що виникають при сумуванні великої кількості чисел з плаваючою комою, що є особливо актуальним у випадках, коли необхідно обробляти великі обсяги даних з високою точністю.

Крім того, у статті обговорюється застосування високоточної арифметики, яка дозволяє виконувати обчислення з більшою кількістю значущих цифр, ніж це можливо при використанні стандартних типів даних з плаваючою комою. Це досягається за рахунок подання чисел у вигляді масивів, що дозволяє зберігати та обробляти додаткові біти точності.

Усі зазначені підходи реалізовані на мові програмування Python, що забезпечує їх доступність та легкість інтеграції в існуючі системи імітаційного моделювання. Python, завдяки своїй простоті та широкому набору бібліотек для наукових обчислень, є ідеальним вибором для реалізації таких алгоритмів.

Таким чином, стаття надає комплексний огляд методів зменшення похибок у чисельних обчисленнях, що можуть бути застосовані для підвищення точності математичних моделей бойових дій. Запропоновані підходи можуть бути корисними для розробників імітаційних систем, які прагнуть підвищити точність та надійність своїх моделей, а також для дослідників, які займаються чисельними методами та їх застосуванням у різних галузях.

Ключові слова: *системи імітаційного моделювання, точність обрахунків двійкових чисел з плаваючою комою, підвищення точності систем імітаційного моделювання, системи імітаційного моделювання бойових дій.*

Вступ. Імітаційне моделювання бойових дій є важливим інструментом у військовій науці та практиці, що дозволяє аналізувати складні сценарії та приймати обґрунтовані рішення. З розвитком комп'ютерних технологій, можливості моделювання значно розширилися, що дозволяє створювати більш реалістичні та детальні моделі [1]. Однак, точність цих моделей значною мірою залежить від математичних розрахунків, які можуть бути піддані різним видам похибок [2]. Зменшення цих похибок є критично важливим для підвищення надійності та достовірності результатів моделювання, що, в свою чергу, впливає на ефективність військових операцій та прийняття рішень [3].

Проблема зменшення похибок у математичних моделях бойових дій є актуальною як з наукової, так і з практичної точки зору. Науково ця проблема пов'язана з розвитком методів чисельного аналізу та оптимізації, які дозволяють підвищити точність розрахунків.

Практично, зменшення похибок у моделях сприяє більш точному прогнозуванню результатів бойових дій, що є критично важливим для прийняття рішень у реальних умовах.

Завданням даного дослідження є розробка підходів, які дозволять зменшити похибки при обчисленні математичних моделей бойових дій. Це включає в себе аналіз існуючих

методів, їх реалізації на мовах програмування високого рівня, розробка підходу що дозволить зменшити значення похибки під час машинного обрахунку двійкових чисел із плаваючою комою. Успішне вирішення цієї задачі сприятиме підвищенню ефективності імітаційного моделювання та, як наслідок, покращенню якості військового планування та управління.

Аналіз останніх досліджень. У статті [4] авторами розглядається використання штучного інтелекту для покращення військових навчальних симуляцій в НАТО. Штучний інтелект дозволяє створювати більш реалістичні та адаптивні сценарії, що допомагає солдатам швидше навчатися, гнучкіше мислити та краще адаптуватися до нових ситуацій. Також штучний інтелект дозволяє швидко моделювати нові загрози, що робить навчання більш актуальним та ефективним.

В основі покращення військових навчальних симуляцій за допомогою штучного інтилекту, описаних у статті, базуються на принципах: реалістичності, адаптивності, персоналізації, ефективності та актуальності.

У роботі [5] авторами розглядаються сучасні виклики та можливості в області моделювання бойових дій. Проаналізовано, як зміни у військовій справі, технологіях та геополітичній обстановці впливають на актуальність та ефективність існуючих методів моделювання та розкриваються аспекти щодо впливу нових технологій, зміни характеру сучасних конфліктів, потреба в більшій реалістичності моделей, впровадження інноваційних підходів до моделювання.

Метою статті є надання рекомендацій щодо розвитку та вдосконалення бойових симуляцій для більш ефективного планування, аналізу та прийняття рішень в умовах сучасного світу.

У досліджені [6] автори розглядають застосування методів машинного навчання для вирішення задач стохастичного управління та теорії ігор. Робота зосереджується на методах глибокого навчання, які відкрили можливість розв’язувати проблеми, навіть у великих розмірах або коли структура дуже складна, що перевищує те, що можна досягти традиційними чисельними методами. Розглядається переважно безперервний час і безперервний простір. Багато нових підходів базуються на методах на основі нейронної мережі для розв’язування диференціальних рівнянь із частковими похідними великої розмірності чи зворотних стохастичних диференціальних рівнянь, або на безмодельному навчанні з підкріпленням для Марківських процесів прийняття рішень. Ця стаття містить вступ до методів машинного навчання і підсумовує сучасні роботи на перехресті машинного навчання та стохастичного керування та ігор.

Стаття [7] представляє систему на основі машинного навчання для прогнозування результатів морських битв. Автори використовують метод випадкового лісу для аналізу різних факторів, що впливають на результат бою, таких як характеристики кораблів (броня, вогнева міць, швидкість), їх розташування та стратегія.

Система моделює бій та враховує різні сценарії, щоб передбачити ймовірність перемоги кожної сторони. Дослідження демонструє потенціал використання машинного навчання для військового моделювання та планування, дозволяючи оцінити ефективність різних стратегій та тактик. Результати показують, що точність симуляцій досягається за рахунок використання ансамблевого методу, врахування технічних характеристик (особливостей) об’єктів, моделювання великої кількості різноманітних сценаріїв та проведення оптимізації моделі під необхідні параметри моделювання.

У роботі [8] досліджується зростаюча роль штучного інтелекту (ШІ) у сфері моделювання та симуляції. Автори розглядають, як методи ШІ, такі як машинне навчання, глибоке навчання та еволюційні алгоритми, можуть бути використані для покращення різних аспектів моделювання та симуляції.

У статті розглядається використання ШІ для створення моделей, калібрування та валідації моделей, оптимізації експериментів, аналізу результатів симуляції та управління складними системами.

Автори також описують виклики та можливості, пов'язані з використанням ШІ в моделюванні та симуляції, та пропонують напрямки для майбутніх досліджень. Стаття підкреслює, що інтеграція ШІ та моделювання та симуляції має значний потенціал для вирішення складних проблем у різних галузях.

Із наведеного аналізу останніх досліджень ми бачимо, що основна наукова діяльність спрямована на досягнення базової точності систем імітаційного моделювання за рахунок використання емпіричних даних, розробки адаптивних моделей та впровадження технологічних інновацій, хоча математичні обчислення і чисельні методи відіграють критично важливу роль у забезпеченні точності та надійності моделей [9, 10].

Метою роботи є розробка методу підвищення точності математичних розрахунків в комп'ютерних системах імітаційного моделювання бойових дій.

Виклад основного матеріалу дослідження

Як правило, математичні моделі, що використовуються у комп'ютерних системах імітаційного моделювання бойових дій описуються диференційними рівняннями [11]. Величина похибки обрахунків рівнянь напряму залежить від точності методу який використовується.

Для комп'ютеризованих систем імітаційного моделювання бойових дій, де потрібна висока точність, найкраще підходить Метод Рунге-Кутта Четвертого Порядку [12].

Переваги методу Рунге-Кутта Четвертого Порядку:

1. Висока точність. Це особливо важливо для моделювання складних систем, де навіть невеликі похибки можуть призвести до значних відхилень у результатах.
2. Стабільність. Є стабільним для багатьох типів задач, особливо коли крок інтегрування обрано правильно.
3. Універсальність. Підходить для широкого спектра задач, включаючи нелінійні та нестаціонарні системи, які часто зустрічаються в моделюванні бойових дій.
4. Простота Реалізації. Незважаючи на високу точність, метод є зручним для інтеграції в комп'ютеризовані системи.
5. Ефективність. Забезпечує хороший баланс між точністю та обчислювальними витратами, що важливо для реального часу або великих симуляцій.

Розглянемо розв'язок звичайного диференційного рівняння методом Рунге-Кутта [13] Четвертого Порядку та Аналітичним методом на прикладі:

$$\frac{dy}{dx} = -2y \tag{1}$$

Метод Рунге-Кутта четвертого порядку обчислює наступне значення *y* за формулами:

1. Обчислення проміжних значень:

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_{n+}\frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\ k_3 &= f\left(t_{n+}\frac{h}{2}, y_n + \frac{h}{2}k_2\right), \\ k_4 &= f(t_{n+}h, y_n + hk_3), \end{aligned} \tag{2}$$

2. Оновлення значення *y:*

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3}$$

Дослід: Порівняння методу Рунге-Кутта четвертого порядку з Аналітичним Розв'язком.

Розв'яжемо диференційне рівняння $\frac{dy}{dx} = -2y$ з початковою умовою y(0) = 1 аналітичним методом.

Це рівняння є лінійним і може бути розв'язане методом розділення змінних:

1. Розділимо змінні:

$$\frac{dy}{y} = -2dx;$$

2. Інтегруємо обидві частини:

$$\int \frac{dy}{y} = \int -2dx$$
$$ln|y| = -2x + C$$

3. Знайдемо y:

Піднесемо обидві частини до степеня e для усунення логарифма:

$$y = e^{-2x+C} = e^{C} \cdot e^{-2x}$$

Позначемо e^C як нову константу C_1:

$$y = C_1 e^{-2x}$$

4. Використовуємо початкову умову $y(0) = 1$:

$$1 = C_1 e^{-2 \cdot 0} \Rightarrow C_1 = 1$$

Отже, аналітичний розв'язок рівняння:

$$y(x) = e^{-2x}$$

Цей розв'язок описує експоненційне зменшення функції *y* з ростом *x*, і він є точним для будь-якого значення *x*.

Щоб обчислити чисельний розв'язок диференційного рівняння $\frac{dy}{dx} = -2y$ з початковою умовою y(0) = 1 методом Рунге-Кутта четвертого порядку на інтервалі $x = 0$ до $x = 1$ з кроком $h = 0.1$, виконаємо наступні обчислення:

Для кожного кроку n, обчислюємо:

1. $k_1 = h \cdot f(x_n, y_n) = h \cdot (-2y_n)$
2. $k_2 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) = h \cdot (-2(y_n + \frac{k_1}{2}))$
3. $k_3 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) = h \cdot \left(-2(y_n + \frac{k_2}{2})\right)$
4. $k_4 = h \cdot f(x_n + h, y_n + k_3) = h \cdot (-2(y_n + k_3))$

Оновлюємо значення *y*:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Обчислення для кожного кроку

Крок 1: $x_0 = 0$, $y_0 = 1$

$k_1 = 0.1 \cdot (-2 \cdot 1) = -0.2$

$k_2 = 0.1 \cdot (-2 \cdot (1-0.1)) = -0.18$
$k_3 = 0.1 \cdot (-2 \cdot (1-0.9)) = -0.182$
$k_4 = 0.1 \cdot (-2 \cdot (1-0.182)) = -0.1636$

$$y_2 = 1 + \frac{1}{6}(-0.2 + 2(-0.18) + 2(-0.182) + (-0.1636) \approx 0.8187$$

Крок 2: $x_1 = 0.1$, $y_1 \approx 0.8187$
$k_1 = 0.1 \cdot (-2 \cdot 0.8187) = -1.16374$
$k_2 = 0.1 \cdot (-2 \cdot (0.8187-0.08187)) = -0.147366$
$k_3 = 0.1 \cdot (-2 \cdot (0.8187-0.073683)) = -0.148003$
$k_4 = 0.1 \cdot (-2 \cdot (0.8187-0.148003)) = -0.135139$

$$y_2 = 0.8187 + \frac{1}{6}(-0.16374 + 2(-0.147366) + 2(-0.148003) + (-0.135139) \approx 0.6703$$

Цей процес повторюється для кожного наступного кроку до $x = 1$. Кінцевий результат буде наближеним до аналітичного розв'язку $y(x) = e^{-2x}$ на кожному кроці. Чисельні значення можуть мати незначне відхилення у зв'язку з округленням, але у загальному вигляді будуть близькими до аналітичного розв'язку.

Ось реалізація методу Рунге-Кутта четвертого порядку для розв'язання диференційного рівняння $\frac{dy}{dx} = -2y$ з початковою умовою y(0) = 1 на мові програмування Python:

```
def runge_kutta_4th_order(f, y0, x0, x_end, h):
    """
    f: функція, що описує диференційне рівняння dy/dx = f(x, y)
    y0: початкове значення y
    x0: початкове значення x
    x_end: кінцеве значення x
    h: крок інтегрування
    """
    # Ініціалізація
    x = x0
    y = y0
    results = [(x, y)]
    # Ітерація по кроках
    while x < x_end:
        k1 = h * f(x, y)
        k2 = h * f(x + h / 2, y + k1 / 2)
        k3 = h * f(x + h / 2, y + k2 / 2)
        k4 = h * f(x + h, y + k3)
        # Оновлення значення y
        y += (k1 + 2 * k2 + 2 * k3 + k4) / 6
        # Оновлення значення x
        x += h
        # Збереження результату
        results.append((x, y))
    return results
# Визначення функції для диференційного рівняння
def dydx(x, y):
    return -2 * y
```

```
# Початкові умови
y0 = 1
x0 = 0
x_end = 1
h = 0.1
# Виконання методу Рунге-Кутта
solution = runge_kutta_4th_order(dydx, y0, x0, x_end, h)
# Виведення результатів
for x, y in solution:
    print(f"x = {x:.1f}, y = {y:.4f}")
```

Пояснення:

Функція runge_kutta_4th_order: Реалізує метод Рунге-Кутта четвертого порядку. Вона приймає функцію f, яка описує диференційне рівняння, початкові умови y0 і x0, кінцеве значення x_end, і крок інтегрування h.

Функція dydx: Визначає праву частину диференційного рівняння $\frac{dy}{dx} = -2y$

Результати: Зберігаються у списку results і виводяться на екран.

Цей код обчислює значення y на кожному кроці x від 0 до 1 з кроком 0.1, використовуючи метод Рунге-Кутта четвертого порядку.

Наступний метод який дозволяє підвищити точність обрахунків це компенсоване сумування (Алгоритм Кахана) [12], який призначений для зменшення похибок округлення при додаванні великої кількості чисел з плаваючою комою. Він працює шляхом збереження невеликої компенсації, яка коригує суму на кожному кроці. Це особливо корисно, коли додаються числа з дуже різними порядками величин.

Принцип роботи алгоритму Кахана

1. Ініціалізація: Початкова сума (total) встановлюється на 0, а компенсація (compensation) також встановлюється на 0.

2. Ітерація по числах: для кожного числа в списку:

Обчислюється кориговане значення (y) шляхом віднімання компенсації від поточного числа.

Додається кориговане значення до поточної суми, отримуючи тимчасову суму (t).

Компенсація оновлюється, щоб врахувати похибку, яка виникла при додаванні.

Оновлюється загальна сума.

3. Результат: Після проходження всіх чисел, total містить суму з мінімальною похибкою округлення.

Алгоритм Кахана є ефективним методом для зменшення похибок округлення при додаванні чисел з плаваючою комою, особливо коли числа мають різні порядки величин. Це робить його корисним у чисельних обчисленнях, де точність є критично важливою.

Порівняємо результати звичайного додавання і алгоритму Кахана для прикладу з числами з плаваючою комою:

Числа

a = 1.0000001

b = 1 x 10^{-7}

c = -1.0000001

Звичайне додавання

1. Додавання a і b:

1.0000001 + 1 1 x 10^{-7}= 1.0000002

Через обмежену точність, результат буде округлений до 1.0000001, втрачаючи b.

2. Додавання c:

1.0000001 - 1.0000001 = 0.0

Результат звичайного додавання: 0.0

Компенсоване сумування (Алгоритм Кахана)

1. Ініціалізація:

total = 0.0

compensation = 0.0

2. Додавання a:

y = a - compensation = 1.0000001 - 0.0 = 1.0000001

t = total + y = 0.0 + 1.0000001 = 1.0000001

compensation = (t - total) - y = (1.0000001 - 0.0) - 1.0000001 = 0.0

total = 1.0000001

3. Додавання b:

y = b - compensation = $1x10^{-7}$ - 0.0 = $1x10^{-7}$

t = total + y = 1.0000001 + $1x10^{-7}$ = 1.0000002

compensation = (t - total) - y = (1.0000002 - 1.0000001) - $1x10^{-7}$ = 0.0

total = 1.0000002

4. Додавання c:

y = c - compensation = -1.0000001 - 0.0 = -1.0000001

t = total + y = 1.0000002 - 1.0000001 = $1x10^{-7}$

compensation = (t - total) - y = ($1x10^{-7}$ - 1.0000002) - (-1.0000001) = 0.0

total = $1x10^{-7}$

Результат компенсованого сумування: $1x10^{-7}$

Цей приклад демонструє, як алгоритм Кахана може значно покращити точність обчислень, коли додаються числа з різними порядками величин, зменшуючи похибки округлення.

Реалізація алгоритму Кахана на Python для прикладу з числами 1.0000001, -0.0000001, $1x10^{-7}$:

```
def kahan_sum(numbers):
    total = 0.0
    compensation = 0.0
    for number in numbers:
        # Коригування числа з урахуванням компенсації
        y = number - compensation
        # Тимчасова сума
        t = total + y
        # Оновлення компенсації
        compensation = (t - total) - y
        # Оновлення загальної суми
        total = t
    return total
# Список чисел
numbers = [1.0000001, 1e-7, -1.0000001]
# Виконання компенсованого сумування
result = kahan_sum(numbers)
# Виведення результату
print(f"Компенсоване сумування (Кахан): {result}")
```

Пояснення:

Ініціалізація:

Total – початкова сума, встановлена на 0.0.

Compensation – початкова компенсація, встановлена на 0.0.

Цикл обчислення:

Для кожного числа в списку:

Коригування числа: обчислюється y як різниця між поточним числом і компенсацією.

Тимчасова сума: обчислюється t як сума total і y.

Оновлення компенсації: обчислюється нова компенсація для коригування похибок округлення.

Оновлення загальної суми: total оновлюється до значення t.

Результат:

Після виконання алгоритму Кахана, результатом є більш точна сума $1x10^{-7}$, яка враховує всі додані числа, включаючи дуже мале b.

Цей код демонструє, як алгоритм Кахана може значно покращити точність обчислень, коли додаються числа з різними порядками величин.

Наступний метод це використання високоточної арифметики, що може бути критично важливою для забезпечення точності моделювання складних сценаріїв, таких як траєкторії снарядів, витрати ресурсів, або логістичні розрахунки. Використання високоточної арифметики дозволяє зменшити похибки, які можуть вплинути на результати моделювання.

Приклад: Імітація розрахунку траєкторії снаряда

У цьому прикладі ми використовуємо модуль decimal для моделювання траєкторії снаряда з високою точністю, що може бути важливим для симуляцій, де навіть невеликі похибки можуть суттєво вплинути на результат.

```
from decimal import Decimal, getcontext
import math
# Встановлення точності для всіх обчислень
getcontext().prec = 20
# Функція для обчислення траєкторії снаряда
def calculate_trajectory(initial_velocity, angle_degrees, time):
    # Перетворення кута в радіани
    angle_radians = Decimal(math.radians(angle_degrees))
    # Обчислення компонент швидкості
    velocity_x = initial_velocity * Decimal(math.cos(angle_radians))
    velocity_y = initial_velocity * Decimal(math.sin(angle_radians))
    # Гравітаційне прискорення
    g = Decimal('9.81')  # м/с²
    # Обчислення положення снаряда в момент часу
    x = velocity_x * time
    y = velocity_y * time - (g * time**2) / 2
    return x, y
# Ініціалізація параметрів
initial_velocity = Decimal('500.0')  # м/с
angle_degrees = Decimal('45.0')  # градуси
time = Decimal('10.0')  # секунди
# Обчислення траєкторії
position_x, position_y = calculate_trajectory(initial_velocity, angle_degrees, time)
# Виведення результату
print(f"Положення снаряда через {time} секунд: x = {position_x}, y = {position_y}")
```

Пояснення

Модуль decimal: використовується для забезпечення високоточної арифметики, що дозволяє уникати похибок округлення, які можуть вплинути на точність моделювання.

Обчислення траєкторії: функція calculate_trajectory обчислює положення снаряда в просторі через заданий час, використовуючи початкову швидкість і кут запуску.

Гравітація: враховується вплив гравітації на траєкторію снаряда.

Застосування

Бойові симуляції: для моделювання траєкторій снарядів, ракет та інших об'єктів з високою точністю.

Логістичні розрахунки: для моделювання витрат ресурсів і часу в бойових умовах.

Аналіз сценаріїв: для оцінки ефективності різних стратегій і тактик.

Цей приклад демонструє, як високоточна арифметика може бути використана в системах імітаційного моделювання бойових дій для забезпечення точності і надійності обчислень.

У системах імітаційного моделювання бойових дій, де можуть бути необхідні обчислення з дуже великими числами [16], наприклад, для моделювання великих масштабів або тривалих періодів часу, використання строкового подання або подання у вигляді масивів може бути корисним. Це дозволяє зберігати і обробляти великі числа, які виходять за межі стандартних типів даних.

Приклад: Додавання великих чисел для моделювання ресурсів

```
def add_large_numbers(num1, num2):
    # Перетворення рядків у масиви цифр
    num1 = [int(digit) for digit in num1]
    num2 = [int(digit) for digit in num2]
        # Вирівнювання довжини масивів
    max_len = max(len(num1), len(num2))
    num1 = [0] * (max_len - len(num1)) + num1
    num2 = [0] * (max_len - len(num2)) + num2
    carry = 0
    result = []
    # Додавання цифр з кінця
    for i in range(max_len - 1, -1, -1):
        total = num1[i] + num2[i] + carry
        carry = total // 10
        result.append(total % 10)
    # Додавання залишкового переносу
    if carry:
        result.append(carry)
    # Перетворення результату в рядок
    result.reverse()
    return ''.join(map(str, result))
# Приклад використання для моделювання ресурсів
# Наприклад, загальна кількість боєприпасів або пального
resources1 = "12345678901234567890123456789O"
resources2 = "98765432109876543210987654321O"
total_resources = add_large_numbers(resources1, resources2)
print(f"Загальна кількість ресурсів: {total_resources}")
```

Пояснення

Перетворення в масиви: кожна цифра великого числа зберігається в окремому елементі масиву, що дозволяє виконувати арифметичні операції з великими числами.

Вирівнювання довжини: масиви вирівнюються по довжині, додаючи нулі на початок, щоб забезпечити коректне додавання.

Арифметичні операції: реалізоване додавання з урахуванням переносу, що дозволяє точно обчислювати суму великих чисел.

Застосування в моделюванні бойових дій: цей підхід дозволяє ефективно працювати з дуже великими числами в системах імітаційного моделювання бойових дій, забезпечуючи необхідну точність і діапазон для складних обчислень.

Як результат для досягти високої точності в комп’ютеризованих системах імітаційного моделювання бойових дій об'єднаємо методи Рунге-Кутта четвертого порядку, компенсованого сумування (алгоритм Кахана), високоточної арифметики, та подання чисел у вигляді масивів.

Розглянемо програмну реалізацію з комбінованим підходом щодо збільшення точності математичних операцій з двійковими числами з плаваючою комою на прикладі звичайного диференційного рівняння:

$$\frac{dy}{dx} = -2y$$

Реалізуємо чисельне розв'язання диференційного рівняння $\frac{dy}{dx} = -2y$ за допомогою методу Рунге-Кутта четвертого порядку, компенсованого сумування (алгоритм Кахана), високоточної арифметики та подання чисел у вигляді масивів, наступним чином.

Реалізація в Python

```
from decimal import Decimal, getcontext
# Встановлення високої точності для обчислень
getcontext().prec = 50
# Функція для алгоритму Кахана, що зменшує похибки округлення
def kahan_sum(input_list):
    total = Decimal('0.0')
    c = Decimal('0.0')  # Компенсація
    for number in input_list:
        y = Decimal(number) - c
        t = total + y
        c = (t - total) - y
        total = t
    return total
# Метод Рунге-Кутта четвертого порядку для чисельного розв'язання диференційних рівнянь
def runge_kutta_4th_order(f, y0, x0, x_end, h):
    n = int((x_end - x0) / h)
    y = Decimal(y0)
    x = Decimal(x0)
    results = []
    for _ in range(n):
        k1 = h * f(x, y)
        k2 = h * f(x + h / 2, y + k1 / 2)
        k3 = h * f(x + h / 2, y + k2 / 2)
        k4 = h * f(x + h, y + k3)
        # Використання алгоритму Кахана для точного сумування
        delta_y = kahan_sum([k1, 2 * k2, 2 * k3, k4]) / 6
        y += delta_y
        x += h
        results.append((x, y))

    return results
# Диференційне рівняння: dy/dx = -2y
def differential_eq(x, y):
    return -2 * y
```

```
# Початкові умови для розв'язання
y0 = '1.0'  # Початкове значення y
x0 = '0.0'  # Початкове значення x
x_end = '5.0'  # Кінцеве значення x
h = '0.1'  # Крок інтегрування
# Розв'язання диференційного рівняння
solution = runge_kutta_4th_order(differential_eq, y0, x0, x_end, h)
# Виведення результатів
for x, y in solution:
    print(f"x = {x}, y = {y}")
```

Висновки. Комбінований підхід дозволяє ефективно розв'язувати диференційні рівняння з високою точністю, що є важливим для моделювання складних процесів у системах імітаційного моделювання бойових дій. Метод Рунге-Кутта четвертого порядку забезпечує точність чисельного інтегрування, алгоритм Кахана зменшує похибки округлення, а високоточна арифметика та подання чисел у вигляді масивів дозволяють працювати з великими числами.

Отримані результати дозволять у подальшому проводити дослідження щодо стійкості чисельних методів до похибок вхідних даних або змін параметрів моделі, а також розробити метод для оцінки надійності та верифікації чисельних моделей у бойових сценаріях.

ЛІТЕРАТУРА:

1. Шинкарук, О., Михайлишин, О. Окремі аспекти застосування імітаційного моделювання у підготовці складових сектору безпеки і оборони України / Збірник наукових праць Національної академії Державної прикордонної служби України 2019. №80(2). С. 227-241 URL: https://doi.org/10.32453/3.v80i2.201

2. Oberkampf, W., DeLand, S., Rutherford, B., Diegert, K., Alvin K. Error and uncertainty in modeling and simulation / Reliability Engineering & System Safety 2002. №75(3). pp. 333-357 URL: https://doi.org/10.1016/S0951-8320(01)00120-X

3. Robinette, S. Testing simulation platforms to accelerate optimal military decision-making in a platoon-formation task / Naval postgraduate school, 2021. 103 p. URL: https://apps.dtic.mil/sti/trecms/pdf/AD1151125.pdf

4. Ortmann, M. Artificial Intelligence in combat simulations: How AI is changing NATO soldier training / Defense magazine 2024 URL: https://www.defensemagazine.com/article/artificial-intelligence-in-combat-simulations-how-ai-is-changing-nato-soldier-training

5. Cox, C., Dellinger, J., Sackett, S., White, A., Gillespie, S. Modeling combat simulations in an evolving world – analysis to provide innovative recommendations for combat simulations / Proceedings of the Annual General Donald R. Keith Memorial Conference 2024 URL: https://www.ieworldconference.org/content/WP2024/Papers/GDRKMCC24_56.pdf

6. Laurière, M. Recent developments in machine learning methods for stochastic control and games / American Institute of Matematical Sciences 2024. №14(3). pp. 435-525 URL:https://doi.org/10.3934/naco.2024031

7. Intizhami, N., Husodo, A., Jatmiko, W. Warfare Simulation:Predicting Battleship Winner Using Random Forest / IEEE International Conference on Communication, Networks and Satellite (ComNetSat) 2019. pp. 30-34 URL:https://doi.org/10.1109/COMNETSAT.2019.8844049

8. Fachada, N., David, N. Artificial Intelligence in Modeling and Simulation / Algorithms №17(6). 2024 URL:https://doi.org/10.3390/a17060265

9. Шаріпова, І., Трутнєв С., Левченко А., Головко О. Витоки помилок прогнозування ситуацій в комп'ютерних системах імітаційного моделювання / Збірник наукових праць Військової академії (м. Одеса) 2020. № 2(14). С. 41 - 50 URL: https://doi.org/10.37129/2313-7509.2020.14.2.41-50

10. Левченко, А., Войтенков, Р. Граничні точності обчислень в інформаційних системах з представленням чисел із плаваючою комою / Збірник наукових праць Військова академія (м. Одеса). Одеса. 2014. № 2(2). С. 157-160 URL:http://vaodessa.org.ua/images/zbirnyk_2/22.PDF

11. Введення в моделювання динамічних систем: Навчальний посібник / Хусаінов, Д., Харченко, І., Шатирко, А. 2010. 132 с. URL:https://www.csc.knu.ua/en/library/books/khusainov-17.pdf

12. Zheng, L., Zhang, X. Numerical Methods / Modeling and Analysis of Modern Fluid Problems, 2017. pp. 361-455 URL: https://doi.org/10.1016/B978-0-12-811753-8.00008-6

13. Dmitruk, B., Stpiczynski, P. Improving accuracy of summation using parallel vectorized Kahan's and Gill-Moller algorithms / Concurrency and Computation: Practice and Experience №35(23). 2023 URL: https://doi.org/10.1002/cpe.7763

REFERENCES:

1. Shynkaruk, O. and Mykhaylyshyn, O. (2019), "Okremi aspekty zastosuvannya imitatsiynoho modelyuvannya u pidhotovtsi skladovykh sektoru bezpeky i oborony Ukrayiny" [Some aspects of simulation modeling usage in the training of the components in the security and defense sector of Ukraine], *Collection of Scientific Works of the National Academy of the State Border Service of Ukraine* No. 80(2), pp. 227-241 URL:https://doi.org/10.32453/3.v80i2.201

2. Oberkampf, W., DeLand, S., Rutherford, B., Diegert, K. and Alvin, K. (2002), "Error and uncertainty in modeling and simulation", *Reliability Engineering & System Safety* No. 75(3), pp. 333-357 URL:https://doi.org/10.1016/S0951-8320(01)00120-X

3. Robinette, S. (2021) *Testing simulation platforms to accelerate optimal military decision-making in a platoon-formation task,"* Naval postgraduate school, Monterey, California 103 p. URL: https://apps.dtic.mil/sti/trecms/pdf/AD1151125.pdf

4. Ortmann, M. (2024) "Artificial Intelligence in combat simulations: How AI is changing NATO soldier training", *Defense magazine.* URL:https://www.defensemagazine.com/article/artificial-intelligence-in-combat-simulations-how-ai-is-changing-nato-soldier-training

5. Cox, C., Dellinger, J., Sackett, S., White, A. and Gillespie, S. (2024) "Modeling combat simulations in an evolving world – analysis to provide innovative recommendations for combat simulations", *Proceedings of the Annual General Donald R. Keith Memorial Conference.* URL: https://www.ieworldconference.org/content/WP2024/Papers/GDRKMCC24_56.pdf

6. Lauriere, M. (2024) "Recent developments in machine learning methods for stochastic control and games", *American Institute of Matematical Sciences* No.14(3), pp. 435-525 URL:https://doi.org/10.3934/naco.2024031

7. Intizhami, N., Husodo, A. and Jatmiko, W. (2019) "Warfare Simulation:Predicting Battleship Winner Using Random Forest", *IEEE International Conference on Communication, Networks and Satellite (ComNetSat), pp. 30-34* URL:https://doi.org/10.1109/COMNETSAT.2019.8844049

8. Fachada, N. and David, N. (2024) "Artificial Intelligence in Modeling and Simulation". *Algorithms* No.17(6) URL:https://doi.org/10.3390/a17060265

9. Sharipova, I., Trutniev, S., Levchenko, A. and Holovko, O. (2020) "Vytoky pomylok prohnozuvannya sytuatsiy v kompyuternykh systemakh imitatsiynoho modelyuvannya" [Sources of situation forecasting errors in computer simulation systems], *Collection of scientific works of the Odesa Military Academy*, No.2(14), pp. 41- 50 URL: https://doi.org/10.37129/2313-7509.2020.14.2.41-50

10. Levchenko A. and Voytenkov R. (2014) "Hranychni tochnosti obchyslen v informatsiynykh systemakh z predstavlennyam chysel iz plavayuchoyu komoyu" [Limiting accuracy of the calculations in information system with numeration with sailling comma] *Collection of scientific works of the Odesa Military Academy*, No.2(2), pp. 157-160 URL:http://vaodessa.org.ua/images/zbirnyk_2/22.PDF

11. Khusainov, D., Kharchenko, I. and Shatyrko A. (2010) "*Vvedennya v modelyuvannya dynamichnykh system" [Introduction to modeling of dynamic systems]*, Study book, Kyiv 132 p. URL:https://www.csc.knu.ua/en/library/books/khusainov-17.pdf

12.Zheng, L. and Zhang, X. (2017) "Numerical Methods. Modeling and Analysis of Modern Fluid Problems", *pp. 361-455* URL:https://www.sciencedirect.com/topics/mathematics/runge-kutta-method

13. Dmitruk, B. and Stpiczynski, P. (2023) "Improving accuracy of summation using parallel vectorized Kahans and Gill-Moller algorithms", *Concurrency and Computation: Practice and Experience*, No.35(23). URL: https://doi.org/10.1002/cpe.7763

Trutniev S.G.

APPROACHES TO THE REDUCTION OF ERRORS IN THE CALCULATION OF MATHEMATICAL MODELS OF COMBAT ACTIONS IN COMPUTER SIMULATION SYSTEMS

The scientific article is dedicated to the study and analysis of modern methods for reducing errors in numerical calculations, which are critically important for enhancing the accuracy and reliability of simulation models of combat operations. Several approaches have been investigated that can be applied to achieve this goal.

The first method discussed is the fourth-order Runge-Kutta method, which is one of the most common methods for numerically solving differential equations. This method allows for high accuracy of solutions with a relatively small number of calculations, making it effective for real-time use in computer simulation systems.

The second approach considered in the article is the use of the Kahan algorithm for precise summation. The Kahan algorithm significantly reduces errors that occur when summing a large number of floating-point numbers, which is particularly relevant in cases where large volumes of data need to be processed with high precision.

Additionally, the article discusses the application of high-precision arithmetic, which allows calculations to be performed with more significant digits than is possible with standard floating-point data types. This is achieved by representing numbers as arrays, allowing for the storage and processing of additional bits of precision.

All the mentioned approaches are implemented in the Python programming language, ensuring their accessibility and ease of integration into existing simulation systems. Python, with its simplicity and wide range of libraries for scientific computing, is an ideal choice for implementing such algorithms.

Thus, the article provides a comprehensive overview of methods for reducing errors in numerical calculations that can be applied to improve the accuracy of mathematical models of combat operations. The proposed approaches can be useful for developers of simulation systems who aim to enhance the accuracy and reliability of their models, as well as for researchers engaged in numerical methods and their applications in various fields.

Keywords: simulation modeling systems, floating-point binary number calculation accuracy, improving the accuracy of simulation modeling systems, combat simulation modeling systems