**УДК 519.852:519.876**
**DOI: 10.15587/1729-4061.2019.157521**

# Розвинення технології довгої арифметики при побудові алгоритмів дослідження лінійних систем

**В. І. Кудін, В. В. Оноцький, Алі Аль-Амморі, Л. О. Шкварчук**

*Розвинуто застосування алгоритмів методу базисних матриць, які оснащені технологією довгої арифметики для покращення точності виконання основних операцій при дослідженні погано обумовлених лінійних систем, зокрема, систем лінійних алгебраїчних рівнянь (СЛАР). Встановлення факту поганої обумовленості системи є досить трудомісткою обчислювальною процедурою. Закладено проведення контролю входження обчислень в стан некоректності та унеможливлення накопичення похибок обчислень, що є бажаною властивістю методів та алгоритмів розв'язання практичних задач.*

*В сучасних ЕОМ, як правило, використовуються стандарті типи цілих чисел, розмір яких не перевищує 64 байта. Було подолано це апаратне обмеження програмним шляхом, а саме, розробкою власного типу даних у вигляді спеціальної бібліотеки Longnut мовою C++ з використанням стандартної бібліотеки шаблонів STL(Standard Template Library). Програмна реалізація була розвинута на проведення обчислень за методами базисних матриць (МБМ) та Гауса, тобто використано довгу арифметику для моделей з раціональними елементами. Запропоновано алгоритми та комп'ютерну реалізацію методів типу Гауса та штучних базисних матриць (варіант методу базисних матриць) в середовищах Matlab та Visual C++ з використанням технології точних обчислень елементів методів, в першу чергу, для погано обумовлених систем різної розмірності. Розроблено бібліотеку Longnut з типами довгих цілих чисел (longint3) та раціональних чисел (longrat3) із чисельником та знаменником типу longint3. Арифметичні операції над довгими цілими числами реалізовано на основі сучасних методів: зокрема, методу Штрасена множення. Наведено результати обчислювального експерименту за згаданими методами, в якому тестові моделі систем генерувались, зокрема, на основі матриць Гільберта різної розмірності, які характеризуються як "незручні"*

*Ключові слова: метод базисних матриць, точні обчислення, погано обумовлена система лінійних рівнянь*

## 1. Вступ

Відомо, що математичне моделювання процесів різної природи приводить до необхідності досліджувати нелінійні рівняння та системи різної складності (математичні моделі). В багатьох випадках, розв'язання їх проводиться введенням певних спрощень в постановках, переходом, зокрема, до різницевих аналогів (дискретний варіант) та врешті-решт до систем лінійних алгебраїчних рівнянь (СЛАР) різної розмірності, частіше, з квадратною матрицею обмежень. Методи та алгоритми СЛАР (як базові) стали предметом дослідження багатьох

вчених. Це в свою чергу обумовило розробку (на даний час) багатьох десятків точних методів та, напевне, сотні ітераційних методів. Виключне становище в ланцюгу розв'язання початкової задачі моделювання обумовило виключні вимоги до обчислювальних властивостей методів та алгоритмів розв'язання СЛАР. Природна складність розв'язання таких задач проявляється в некоректності та поганій обумовленості. Зусилля обчислювачів спрямовані на подолання (чи послаблення) прояву поганої обумовленості та некоректності в ході розв'язання задачі та (накопиченню похибок обчислень). З'являються нові підходи розв'язання відомих задач. Один з перспективних, на думку авторів, є розвиток відомих методів (зокрема алгоритмів методу базисних матриць), оснащенням технологією довгої арифметики при дослідженні погано обумовлених лінійних систем. Звичайно, остання технологія (довгої арифметики) передбачає використання і інших технологій, про що буде сказано нижче, зокрема, введення нового класу даних. Саме вдале поєднання ряду нових технологій організації алгоритмів обчислень може надати нові позитивні можливості.

Задачі моделювання процесів різної природи часто описуються класом оптимальних задач (на екстремум), зокрема і СЛАР. Точні методи розв'язання СЛАР в ряді згаданих задач є основоположними, оскільки дослідження початкових більш складних математичних моделей "зводиться" (після спрощень) саме до аналізу таких лінійних систем.

Відомо, що:

– переважна більшість математичних постановок задач (в перше чергу, по дослідженню властивостей процесів) за своєю природою є нелінійними, тобто не мають адекватного подання в класі лінійних моделей (матричних структур) – систем лінійних алгебраїчних рівнянь, нерівностей, задач лінійного програмування тощо;

– важливі параметри процесу (в ході моделювання та спрощень) знаходять своє відображення ("переходять") до окремих елементів, рядків, стовпців та блоків ("підматриць") матриці обмежень тощо, тобто зазнають змін та уточнень. А тому врахування впливу змін у моделі в результаті уточнень (без перерозв'язання) є бажаною характеристикою методів моделювання.

Наявна властивість поганої обумовленості та некоректності в задачі обумовлює на всіх стадіях моделювання. Зокрема, на стадії комп'ютерного подання моделі – заокруглення, усікання, обмеженість довжини мантиси спричиняє накопичення помилок, похибок, неточностей подання моделі, тощо. Це породжує неадекватність досліджуваного процесу та моделі. Слід зазначити, що часто в таких ситуаціях навіть малі кількісні неточності у поданні моделі можуть зумовити значні відхилення (похибку) розв'язку. Оснащення алгоритмів технологією, яка перешкоджає накопиченню похибок на стадіях моделювання, зокрема і в ході ітерацій алгоритму методу, є актуальним напрямком досліджень.

**2. Аналіз літературних даних та постановка проблеми**

Станом на сьогодні розроблено досить багато точних та ітераційних методів розв'язання систем лінійних алгебраїчних рівнянь. Серед них, зокрема, наведені в [3–4] та багато інших. Можна констатувати, що питання розробки уні-

версального високоточного методу розв'язання широкого класу лінійних задач "не знялося". Це можна пояснити об'єктивними складностями в задачах, що виникають, насамперед, при моделюванні процесів різної природи. Математичні моделі, досить часто, подаються як великої розмірності, є некоректними (чутливими до неточностей), погано обумовленими, тощо [1, 2]. Технологічні досягнення сьогодення, такі як збільшення розрядності процесорів до 64 та обсягів різних типів пам'яті, разом із підвищенням швидкодії виконання операцій надають додаткові можливості при представленні довгих чисел. Однак вони не розв'язують проблему представлення чисел та реалізації операцій для чисел з розрядністю більше за 64 і вона лишається відкритою. В зв'язку з цим, для різних видів поганої обумовленості чи структури матриць обмежень виникає необхідність адаптувати алгоритм розв'язку для забезпечення якості обчислень [3–5].

Звичайно, серед них особливе місце займають алгоритмічні схеми, що базуються на методі Гауса. Відомо, що за своїми структурними властивостями моделі СЛАР, нерідко, можуть бути некоректними (природна складність). Однією з проявів некоректності є властивість поганої обумовленості (число обумовленості $M_A = \|A\| \times \left\|A^{-1}\right\|$ набуває великих значень, що впливає на похибки виконання основних операцій) [1–5]. Можна переконатись, що інформація про значення числа обумовленості (чи його оцінка), як фактор контролю коректності обчислень, наразі залишаються в полі зору подальших досліджень [3–5]. Зокрема, наявність контролю обумовленості системи, накопичення похибок обчислень при розв'язанні таких незручних задач є важливою необхідною і невід'ємною складовою обчислювального процесу. З точки зору класичних методів типу Гауса, наприклад, процедура побудови оцінювача числа обумовленості може бути додана якби "зовні". В схемі таких методів відсутні складові знаходження чи оцінки обумовленості $M_A = \|A\| \times \left\|A^{-1}\right\|$ (оскільки, обернена матриця невідома) і сама по собі є обчислювально трудомісткою процедурою. Органічно «вбудувати» так процедуру в алгоритм є окремою задачею. Особливо проблема оцінювання числа обумовленості заявила про себе при накопиченні значного досвіду розв'язання практичних задач. Наявність оцінки числа обумовленості в ході обчислень є важливою складовою, оскільки вона "дає сигнал" (вказує) на коректність обчислень. Як додатковий "захід", в вирішенні проблеми обумовленості (покращення обумовленості) може бути представленим у вигляді направленого перетворення початкової задачі (передобумовлення), наприклад, множенням зліва та справа СЛАР на спеціальні матриці [5, 6].

Для проведення перевірки та контролю властивостей алгоритмів розв'язання (тестування даного класу задач) розроблено ряд алгоритмів, програм та модельних задач з погано обумовленими матрицями обмежень, наприклад, підпрограми лінійної алгебри BLAS (Basic Linear Algebra Subroutines) [6]. Відомо, що до таких тестувальних матриць належить і матриця Гільберта. В даній роботі для проведення тестування алгоритмів вона активно застосована. "Наріжним каменем" при проведенні моделювання залишається дослідження впливу збурень (зокрема, похибок обчислень) на властивості системи. Ця проблема розглядається в ряді наукових праць, зокрема, [1, 2, 6, 7]. Один з напрям-

ків, що активно розвивається останнім часом, пов'язаний з розробкою математичних методів, алгоритмів та програмного забезпечення виконання основних операцій, зокрема, введенням нових типів даних (для дій з раціональними числами), що унеможливлює накопичення похибок. Такі підходи розглянуті в [8–11]. Одним із недоліків такого підходу є додаткове обчислювальне навантаження в алгоритмі, яке спричиняє уповільнення обчислень та накладає обмеження на розмірність розв'язуваних задач.

Але згадані підходи не закривають проблеми, тому доцільно розвинути проведення обчислень використанням технології довгої арифметики на моделі з раціональними елементами, один з варіантів якої реалізовано у вигляді бібліотеки Longnum на мові C++ [23]. Застосування раціональної арифметики для прямих методів розв'язання СЛАР усуває обчислювальну похибку і дає можливість зосередитися на властивостях власне моделі, наприклад [7].

В сучасних ЕОМ, як правило, використовуються стандарті типи цілих чисел, розмір яких не перевищує 64 байта. Подолання такого апаратного обмеження можна вирішити програмним шляхом, а саме, розробкою власного типу даних. Відомими прикладами реалізації такого підходу є бібліотеки GMP [8], MPI [9], LIP [10], OpenSSL [11] та LibTomMath [12]. Бібліотеки GMP та LIP громісткі та не достатньо зручні у використанні; OpenSSL та LibTomMath несуть криптографічне призначення; бібліотека MPI наразі не розвивається [9]. Важливо розробити бібліотеку, що не залежить від сторонніх розробок крім стандартних бібліотек C++ та може гнучко змінюватися та підлаштовуватися під конкретні дослідження.

Звичайно, згадані вище прийоми інтенсифікації процедур обчислень мають свої сильні та слабкі сторони. Часто, вони застосовуються розрізнено.
Бачиться доцільною розробка оригінальної технології обчислень, в якій би були раціонально використані сильні сторони названих підходів.

Запропонований у [14] метод володіє рядом структурних властивостей, зокрема, аналізувати та розв'язувати наряду зі СЛАР також задачі лінійного програмування (ЗЛП), він поширений на слабко нелінійні задачі, містить в елементах методу складові для обчислення та оцінювання обумовленості системи в ході ітерацій [15–18], може бути оснащеним технологією довгих чисел. Наведені властивості методу та алгоритмів базисних матриць при оснащенні технологією роботи з довгими чисел набувають рис універсальності для застосування при розв'язанні широкого класу задач.

Аналіз літературних джерел та "виділення" основних проблем в організації обчислень вказують, що підняття точності виконання основних операцій може бути досягнуте включенням та вдалим поєднанням додаткових процедур: передобумовлення, знаходження оцінки та числа обумовлення, високоточного проведення основних операцій. Саме на розробку технології проведення обчислень з такими властивостями (з елементами універсальності) мають бути направлені зусилля дослідників.

Вибір МБМ, як базового, при розробці технології довгих чисел, ґрунтувався на наявності такої унікальної властивості як контролювати входження обчислень в стан некоректності. Зокрема, інформація про пряму та обернену матриці,

як складові обчислення числа обумовленості. Побудова оцінювача числа обумовленості (та передобумовлювача) та його властивості (з експериментами) на основі МБМ розглянуто в [18].

Це вказує на доцільність застосування МБМ та його алгоритмів, як такі, що пройшли апробацію [14–18], для "надбудови" процедурами високоточного проведення основних операцій (технологією довгої арифметики) при дослідженні лінійних систем.

**3. Ціль та задачі дослідження**

Метою дослідження є розвинення технології довгої арифметики, що підвищує точність виконання основних операцій алгоритмів дослідження і розв'язку СЛАР і мінімізує модуль величини похибок на стадії комп'ютерного моделювання.

Для досягнення мети були поставлені такі завдання:

– розробити алгоритм розв'язку СЛАР (лінійна система) з елементами аналізу та контролю обумовленості та накопичення похибок в ході ітерацій (технологія довгої арифметики) при реалізації технології довгої арифметики;

– реалізувати типи довгих цілих та відповідних раціональних чисел із швидкими операціями множення, ділення, побудованих на сучасних алгоритмах та прискорити операції множення та ділення операцій з довгими числами.

**4. Технології довгої арифметики прискорення виконання основних операцій на основі алгоритму методу базисних матриць (МБМ)**

Перш за все хотілось би наголосити, що в статті розвинуто метод базисних матриць (перші публікації кінець 80-х років) оснащенням технологією довгої арифметики [14]. З детальним викладенням обгрунтування методу, його властивостей, результатів обчислювального експерименту, порівнянь з відомими іншими методами можна ознайомитись в [15–17].

Розглянемо СЛАР виду

$$Au=C, \qquad (1)$$

де матриця $A$ розмірності $(m\times m)$, $C=(c_1,c_2,..,c_m)^{\mathrm{T}}$ – вектор стовпець розмірності $m$, $u=(u_1,u_2,..,u_m)^{\mathrm{T}}$ – шуканий вектор розмірності $m$, $T$ – знак транспонування, $a_j=(a_{j1}, a_{j2}, \ldots, a_{jm})$, $j=1,2, \ldots m$ – рядки матриці $A$. Рівняння (1) доповнюється допоміжною СЛАР виду:

$$Iu=K, \qquad (2)$$

де $I$ – одинично-діагональна матриця розмірності $(m\times m)$, а $K=(\underbrace{1,1,....,1}_{m})^{\mathrm{T}}$ – вектор розмірності m. Слід зазначити, що система (2), зазвичай, тривіальна, з відомими властивостями виконує лише допоміжну роль – побудови початкових значень елементів МБМ, зокрема, оберненої матриці та розв'язку.

Тобто, в основу побудови алгоритму розв'язання СЛАР покладено метод базисних матриць, оскільки в ньому згідно [14] закладена здатність:

– знаходити величину рангу матриці обмежень системи (1) ;

– знаходити розв'язок СЛАР (1) ;

– контролювати обумовленість системи;

– аналізувати вплив змін у моделі (1) в результаті уточнень (без перерозв'язання);

– оснащення технологією, що перешкоджає накопиченню похибок;

– будувати початкові розв'язки задач на основі тривіальних базисних матриць (2), що виключає трудомісткі початкові обчислення

– застосовувати схему аналізу для задач, що передбачають багатокроковість або багаторазовість розрахунків на моделях з незначними змінами.

Коротко нагадаємо [14], що основою запропонованого методу штучних базисних матриць (МШБМ) є ідея порядкової базисної матриці. Базисні матриці в ході ітерацій послідовно змінюються вводом-виводом із неї рядків-нормалей обмежень задачі.

Підматрицю $A_б$, складену із $m$ лінійно незалежних рядків-нормалей ($i_1$, $i_2$, …, $i_m$) обмежень, будемо називати штучною базисною, а розв язок $u_0$ відповідної їм системи рівнянь $A_б u=C^0$, де $C^0=(c_{i_1}, c_{i_2},..., c_{i_m})^T$ штучним базисним.

Нехай: $e_{ri}$ – елементи матриці $A_б^{-1}$, оберненої до $A_б$; $u_0=(u_{01}, u_{01}, ..., u_{0m})^{\mathrm{T}}$ – базисний розв'язок; $\alpha_r=(\alpha_{r1}, \alpha_{r2}, …, \alpha_{rm})$ – вектор розвинення вектору-нормалі обмеження $\alpha_r u \le c_r$ за рядками базисної матриці $A_б$; $\Delta_r=\alpha_r u_0 - c_r$ – нев'язка $r$-го обмеження (1) в вершині. Всі введені елементи в новій базисній матриці $\overline{A}_б$, відмінній від $A_б$ одним рядком, будемо позначати рискою зверху.

Згідно Теореми 1 [14], між коефіцієнтами розвинення нормалей обмежень, елементами обернених матриць, базисними розв'язками, нев'язками обмежень в двох суміжних базисних матрицях встановлено відповідні співвідношення.

На основі них будується схема визначення рангу системи (1) та розв'язку системи рівнянь, послідовними змінами базисних матриць та відповідних штучних розв язків.

Попередні результати застосування технології довгої арифметики у поєднанні з методом базисних матриць для аналізу властивостей СЛАР (з реалізацією) розглядались в [22]. "Довга арифметика" представлення раціональних чисел при виконанні основних операцій виключає накопичення похибок, що у поєднанні з контролем обумовленості в ході обчислень є ефективною, особливо при дослідженні погано обумовлених систем. Для останніх, вплив навіть малих похибок у поданні моделі та обчисленням є суттєвим.

В подальших версіях технологія неодноразово зазнала удосконалень і програмно, і алгоритмічно було проведено відповідні обчислювальні експерименти.

Предметом дослідження будуть властивості нового алгоритму та комп'ютерної реалізації методу штучних базисних матриць [14] для моделі (1) в середовищі Visual C++ з використанням раціональної арифметики, в якій

множення довгих чисел здійснюється за допомогою методу Штрасена, що базується на швидкому алгоритмі дискретного перетворення Фур'є [19].

*Концепція представлення цілих чисел.* В сучасних ЕОМ, як правило, використовуються стандарті типи цілих чисел, розмір яких не перевищує 64 байта. Подолання такого апаратного обмеження можна вирішити програмним шляхом, а саме, розробкою власного типу даних [22]. В розробленій на мові C++ бібліотеці Longnum реалізовані типи довгих цілих чисел longint3 та відповідних раціональних чисел longrat3 із швидкими операціями множення, ділення, побудованих на сучасних алгоритмах.

Для реалізації довгих цілих чисел довільного розміру та відповідних раціональних чисел в C++ був використаний *об'єктно-орієнтований підхід* (ООП) [21]. Він полягає у наступному: реальному об'єкту предметної області ставиться у відповідність так званий *об'єктний тип* або *клас* об'єктів, що є узагальненням структурного типу. Скористаємось означенням [21].

*Означення 1. Клас* в C++ – це програмна конструкція, яка складається з даних (*елементів даних* або *полів*) та підпрограм, що оперують над цими полями та описують властивості відповідного об'єкту предметної області, що моделюється.

Загальна концепція довгої або точної арифметики (multiple precision arithmetic), поняття довгого цілого числа описано, зокрема, в [12]. Але для ясності введемо поняття:

*Довгим цілим числом або цілим числом довільного розміру* назвемо ціле число, яке не обмежується діапазонами стандартних комп'ютерних типів. Наприклад, стосовно мови C++, такі числа принципово неможливо зберігати в змінній типу __int64 або unsigned __int64, які обмежуються діапазонами від $-2^{63}$ до $2^{63}$ та від 0 до $2^{64}$-1 відповідно.

*Довга або точна арифметика* – це арифметика довгих цілих чисел.

Ціле число довільного розміру програмно представлено у вигляді класу longint3. Раціональне число як пара (чисельник класу longint3, знаменник класу longint3) запрограмоване у вигляді класу longrat3. Ціле число зберігається у динамічному масиві, максимальна довжина якого 4096. Елементами вектора є 16-бітні беззнакові числа стандартного типу C++ *unsigned __int16* і є, фактично, "цифрами" в системі числення з основою $BASE=2^{16}$. Так, наприклад, ціле число з діапазону від $2^{32}$ до $2^{64}$-1 буде зберігатися у масиві розміру 3.

Масив "цифр", знак числа та операції над числами такої структури оформлені у вигляді класу longint3. Зокрема, в класі longint3 реалізовані такі операції з цілими числами: додавання, віднімання, множення та ділення з остачею, порівняння, конвертація у рядок символів типу char. При розробці бібліотеки Longnum [22] використано елементи стандартної бібліотеки шаблонів STL(Standard Template Library) [13].

*Оголошення класу longint3*

```
class longint3
{
public:
unsigned __int16 *s;
```

```
unsigned int l;
longint3();
longint3(unsigned n);
longint3(unsigned __int16 *p,int n);
longint3(const char *s1);
longint3(const char *s1,int n);
longint3(const longint3 &a);
longint3 operator=(const longint3 &a);
~longint3();
char* text(void);
friend longint3 operator+(longint3 &a,longint3 &b);
friend longint3 operator-(longint3 &a,longint3 &b);
friend longint3 operator*(longint3 &a,longint3 &b);
friend longint3 operator/(longint3 &a,longint3 &b);
friend ostream& operator<<(ostream &os,const longint3 &a);
friend istream& operator>>(istream &is,longint &a);
friend longint3 karatsuba2(longint3 &u1,longint3 &u2,unsigned int n);
friend longint3 tomakuka(longint3 &u1,longint3 &u2);
friend longint3 fastmul2(longint3 &A, longint3 &B);
friend __int8 divmod(longint3 &A, longint3 &B, longint3 &Q, longint3 &R) ;
void mult2(int n);
void mult(int m);
void div(int m);
friend int longcmp2(longint3 &a,longint3 &b);
unsigned int toint();
void print();
void operator+=(longint3 &b);
void operator-=(longint3 &b);
void operator*=(longint3 &b);
friend void savebint2(FILE *f,longint3& a);
friend void loadbint2(FILE *f,longint3& a);
friend longint3 ncd(longint3 a,longint3 b);
};
```

*Представлення раціональних чисел.* Раціональне число представляється як пара: чисельник та знаменник типу longint3 і оформлено у вигляді класу longrat3. Даний клас також оснащується операціями додавання, віднімання, множення, ділення, скорочення, порівняння, представлення у вигляді десяткового числа заданої точності, конвертація у тип double, конвертація у рядок символів типу char. Операції ‘+’, ’-’, ’*’, ’/’ побудовані з використанням відповідних операцій над числами типу longint3. Данні класи були відкомпільовані у вигляді динамічної бібліотеки та протестовані для точного розв’язування систем лінійних алгебраїчних рівнянь.

*Оголошення класу longrat3*

```
class longrat3
{
```

```
public:
  __int8 sign;
  longint3 num;
  longint3 denom;
  longrat3():num("0"),denom("1"){sign=0;}
  longrat3(char *s);
  longrat3(double d);
  longrat3(char *s1,char *s2);
  longrat3(const longrat3 &l1):num(l1.num),denom(l1.denom),sign(l1.sign){}
  longrat3& operator=(const longrat3 &a);
  friend ostream& operator<<(ostream &f,const longrat3 &a);
  void text(char *s);//return s as 'p/q';
  friend void savebrat3(FILE *f,longrat3& a);
  friend void loadbrat3(FILE *f,longrat3& a);
  friend longrat3 operator+(longrat3 &a,longrat3 &b);
  friend longrat3 operator-(longrat3 &a,longrat3 &b);
  friend longrat3 operator*(longrat3 &a,longrat3 &b);
  friend longrat3 operator/(longrat3 &a,longrat3 &b);
  void operator+=(longrat3 &a);
  void operator-=(longrat3 &a);
  void operator*=(longrat3 &a);
  void operator/=(longrat3 &a);
  char* decimal(unsigned __int32 size);
  operator double();
  friend __int8 longrabscmp(longrat3 &a,longrat3 &b);
  ~longrat3(){};
  void reduce(void);
};
```

Операції ‘+’ та ’-’ реалізовані за класичними алгоритмами додавання та віднімання; множення та ділення суттєво покращені в порівнянні з традиційними алгоритмами множення та ділення в стовпчик і оптимізовані як за часом, так і за використанням ресурсів ЕОМ.

*Прискорення операцій множення та ділення довгих чисел*

Арифметичні операції з точними раціональними числами базуються на множенні довгих цілих чисел. Для здійснення скорочення раціональних дробів реалізовано ділення довгого цілого числа на довге ціле число.

*Множення довгих цілих чисел з використанням метода Штрасена, що базується на дискретному перетворення Фур’є.* Добуток цілих чисел

$A=a_0+a_1BASE+\ldots+a_{n-1}BASE^{n-1}$ та $B=b_0+b_1BASE+\ldots+b_{m-1}BASE^{m-1}$

в системі числення з основою $BASE$ можна інтерпретувати як добуток багаточленів

$$(a_0+a_1x+\ldots+a_{n-1}x^{n-1})(b_0+b_1x+\ldots+b_{m-1}x^{m-1})=c_0+c_1x+\ldots+c_{n+m-1}x^{n+m-1},$$

де $c_i$ – компоненти вектора – згортки векторів $(a_0, a_1, \ldots, a_{n-1})$ та $(b_0, b_1, \ldots, b_{m-1})$.

Скористаємося означеннями [18].

*Означення 2.* Циклічною згорткою векторів $a$ і $b$ називається вектор $c=a\times b$ із координатами

$$c_i = \sum_{k+l\equiv i \,(\mathrm{mod}\, m)} a_k b_l.$$

*Означення 3.* Дискретне перетворення Фур'є (ДПФ) вектора $(a_0, a_1,\ldots, a_{N-1})$ визначається як комплексний вектор з координатами $(y_0, y_1, \ldots, y_{N-1})$:

$$y_k = \sum_{j=0}^{N-1} a_j \omega^{kj},$$

де $\omega$ – головний комплексний корінь $N$-го степеня з одиниці,

$$\omega = \cos\frac{2\pi}{N} + i\sin\frac{2\pi}{N}.$$

*Зауваження.* Обернене дискретне перетворення Фур'є ($ДПФ^{-1}$) можна обчислити за формулою:

$$a_k = \frac{1}{N}\sum_{j=0}^{N-1} y_j \omega^{-kj}.$$

В основі методу Штрасена [20] множення довгих чисел лежить

*Теорема про згортку [20].* Перетворення Фур'є від циклічної згортки двох векторів є скалярний добуток Фур'є-образів цих векторів:

$$c = a\times b \Leftrightarrow F(c) = F(a) * F(b),$$

а тоді

$$c=F^{-1}(F(a)*F(b)).$$

Тут * означає покомпонентне множення векторів.

Отже, для множення довгих цілих чисел $A$ та $B$ достатньо:

1. Обчислити коефіцієнти згортки $c_i$, $i = \overline{0, n+m-1}$.

2. Зробити переноси, щоб всі коефіцієнти були менше $BASE$.

Теорема про згортку дозволяє побудувати ефективний алгоритм обчислити коефіцієнтів згортки на кроці 1 за допомогою ДПФ:

1. 1. Обчислити $F(a)$ і $F(b)$.

1. 2. Покомпонентно перемножити отримані вектори.

1. 3. Обчислити обернене ДПФ від скалярного добутку.

Множення багаточленів зводиться до скалярного добутку відповідних векторів. Передбачається, що на кожному кроці розміри векторів однакові й рівні $N$. Складність алгоритму множення має порядок $O(N\log N)$. Причому, найкращої швидкодії досягають лише варіанти ШПФ (швидкого алгоритму дискретного перетворення Фур'є), що працюють на векторах розміру $N=2^k$, тому вектори в таких ситуаціях необхідно доповнити нулями. Метод Штрасена реалізовано в класі longint3 бібліотеки Longnum.

*Ділення довгих цілих чисел.* Операція ділення довгих чисел використовується в операції скорочення раціональних чисел. За основу взятий "шкільний" метод ділення "в стовпчик", складність якого O(nm), де n та m – кількості цифр діленого та дільника відповідно.

Наведемо основні стадії алгоритмічної схеми знаходження величини рангу, початкової базисної матриці та розв'язку невиродженої системи (1) з раціональними елементами, що грунтується на технології точних обчислень. Алгоритм може бути застосованим при перерахунку оберненої матриці в ході ітераційного процесу. Такий перерахунок доцільно виконувати при накопиченні значних похибок при знаходженні елементів методу. За результатами наведеного алгоритму можна конструювати аналітичне представлення загального розв'язку відповідної системи лінійних алгебраїчних нерівностей (СЛАН) [14].

**Алгоритм МШБМ**

*На вході:* матриця коефіцієнтів $A$ розмірності $m\times m$, вектор правих частин СЛАР $C$ розмірності $m$ та одинична базисна матриця $I$ розмірності $m\times m$, що представляють собою динамічно створені масиви елементів типу longrat3. Надалі наступні кроки виконуємо з використанням операцій порівняння, додавання, віднімання, множення та ділення класу longrat3.

*Крок 1.* Проводимо симплексні ітерації по заміщенню рядків базисної матриці $I$ системи (2) нормалями обмежень системи (1), згідно співвідношень теореми 1 [14] .

Знаходимо відповідні елементи методу вектори розвинення за рядками базисних матриць обмежень (2), обернену базисну матрицю, штучні базисні розв'язки $u_0^{(k)}$, де $k$ – номер ітерації.

*Крок 2.* Перевіряємо кількість ітерацій $r$ заміщення рядків допоміжної системи рядками основної системи для яких виконуються умови невиродженості, тобто $\alpha_{lk}^{(i)} \neq 0$ – число визначає ранг основної системи, (Наслідок 2–3 теореми 1 [14]).

*Крок 3.* Якщо кількість ітерацій, для яких $\alpha_{lk}^{(i)} \neq 0,$ рівна $m$, то переходимо на наступний крок. В супротивному на передостанній крок.

*Крок 4.* Знаходимо єдиний розв'язок (наслідок 1 теореми 1 [14]). згідно співвідношення $A_6^{-1}c^0 = u^0$.

*Крок 5*. Виконання умови $r<m$ означає порушення умови єдиності розв'язку за схемою методу, тобто модель потребує уточнення та подальшого аналізу розв'язності (наслідок 1 теореми 1 [14]).

*Останній крок*. Формування вихідної інформації за результатами аналізу (1) у вигляді масиву елементів типу longrat3.

Алгоритм може бути застосованим при перерахунку оберненої матриці в ході ітераційного процесу знаходження оптимального розв'язку. Такий перерахунок доцільно виконувати при накопиченні значних похибок при знаходженні елементів методу. За результатами наведеного алгоритму можна конструювати аналітичне представлення загального розв'язку СЛАН.

## 5. Результати досліджень обчислювальних властивостей програмної реалізації методів та алгоритмів

Було розглянуто дві СЛАР з погано обумовленими матрицями, а саме: матриця Гілберта $A_1$ з елементами

$$a_{ij}^{(1)} = 1/(i+j-1),\ i=\overline{1,m};\quad j=\overline{1,m}$$

та матриця $A_2$ з елементами

$$a_{ij}^{(2)} = \begin{cases} m-i+1, & \text{при } i>j, \\ m-i, & \text{при } i=j, \\ m-j+1, & \text{при } i<j. \end{cases}$$

Матриця $A_1$ розглядалася з такими варіантами вектора $c_1$ правих частин:

1) $c_i^{(1)} = 1,\ i=\overline{1,m};$

2) $c_i^{(1)} = (-1)^{i-1},\ i=\overline{1,m};$

3) $c_i^{(1)} = (1+(-1)^{i-1})/2,\ i=\overline{1,m}.$

Для матриці $A_2$ вектор правих частин $c_2$ визначається співвідношенням:

$$c_i^{(2)} = \begin{cases} m-i, & \text{при } i\le 2, \\ m-i+1, & \text{при } i>2. \end{cases}$$

Розв'язування СЛАР з використанням бібліотеки Longnum. При розв'язанні СЛАР розмірності $m$=50 з матрицею Гільберта та одиничною правою частиною з використанням точних обчислень було отримано точний розв'язок (табл. 1).

СЛАР розв'язувалися методами Гауса та МШБМ, з матрицею Гільберта розмірності $m$=100, посилання на точний розв'язок опубліковано в [22].

Таблиця 1

Точний розв‘язок СЛАР методом базисних матриць з матрицею Гільберта, $m$=50

| Компоненти розв’язку в точному вигляді | Компоненти розв’язку з плаваючою крапкою |
|---|---|
| -50 | -50 |
| 124950 | 124950 |
| -77968800 | -7.79688e+007 |
| 21580031200 | 2.158e+010 |
| -3350299843800 | -3.3503e+012 |
| 331679684536200 | 3.3168e+014 |
| -22701631741588800 | -2.27016e+016 |
| 1135544885686411200 | 1.13554e+018 |
| -43221677211439026300 | -4.32217e+019 |
| 1290780705857666723700 | 1.29078e+021 |
| -30978736940584001368800 | -3.09787e+022 |
| 609077811418589580631200 | 6.09078e+023 |
| -99651897479319239719938800 | -9.96519e+024 |
| 13744885977768825312850 6200 | 1.37449e+026 |
| -16157253720805802816738 68800 | -1.61573e+027 |
| 16336778762148089514702451200 | 1.63368e+028 |
| -14320207633695434715231 3673800 | -1.43202e+029 |
| 10955702103148998669680 46826200 | 1.09557e+030 |
| -73579036347074756497607 09548800 | -7.3579e+030 |
| 43597107685981413891518 442451200 | 4.35971e+031 |
| -22888481535140242293047 1822868800 | -2.28885e+032 |
| 10686481514932825143171 00869131200 | 1.06865e+033 |
| -44512286640711932827753 62297868800 | -4.45123e+033 |
| 16584823623599852477032 588070131200 | 1.65848e+034 |
| -55397917798274507232310 242095368800 | -5.53979e+034 |
| 16619375339482352169693 0726286106400 | 1.66194e+035 |
| -44842811566887293428284 2669742393600 | -4.48428e+035 |
| 10893912110419395975513 22864353606400 | 1.08939e+036 |
| -23844328037601637109669 26065345393600 | -2.38443e+036 |
| 47036551979050078436315 46185978606400 | 4.70366e+036 |
| -83620536851644583886783 04330628633600 | -8.36205e+036 |
| 13391467868333092050131 020150715366400 | 1.33915e+037 |
| -19302545482089495962884 165764117071100 | -1.93025e+037 |
| 25010001538317978699384 350682432678900 | 2.501e+037 |
| -29077372030708791844266 926744973633600 | -2.90774e+037 |
| 30264203542166293552196 189061095006400 | 3.02642e+037 |
| -28115818722814982590157 570701819743600 | -2.81158e+037 |

| | |
|---|---|
| 232278969872196824758715942028912564 00 | 2.32279e+037 |
| -16986606106997219317534905455853993600 | -1.69866e+037 |
| 109335222739975527362700016050500064 00 | 1.09335e+037 |
| -615010627912362341415187590284062860 0 | -6.15011e+036 |
| 29963932436658224724511519122108714 00 | 2.99639e+036 |
| -12501958204864202606145395733487536 00 | -1.2502e+036 |
| 44017170315665743085995959579367246400 | 4.40172e+035 |
| -12823183914274524328771549729500360 0 | -1.28232e+035 |
| 30079073379162464474896227760556400 | 3.00791e+034 |
| -54585842049141712468620753591936 00 | -5.45858e+033 |
| 71908012839747570522266361680640 0 | 7.1908e+032 |
| -61171747033813037423455759068600 | -6.11717e+031 |
| 25222836136391048333703124314 00 | 2.52228e+030 |

В результаті експериментів для СЛАР з матрицею Гільберта $A_1$ розмірності $m$=100 та одиничною правою частиною $c_1$ отримано точні розв'язки (табл. 2) як методом Гауса, так і методом штучних базисних матриць, які співпадають. У табл. 2–4 вказаний час виконання процедури обчислень точного розв'язку на ЕОМ з процесором AMD Athlon(tm) 64X2 Dual Core Processor 4200+, 2.21 ГГц, 3 ГБ ОЗП.

Для СЛАР з матрицею коефіцієнтів $A_2$ проведені розрахунки з використанням методу Гауса та МШБМ. Результати обчислювального експерименту наведені в табл. 3.

Таблиця 2

Розв'язування СЛАР з матрицею Гільберта $A_1$ в точних числах

| Вектор обмежень | Розмірність m | Метод Гауса з вибором максимального елементу | | Метод штучних базисних матриць | |
|---|---|---|---|---|---|
| | | Мінімальний ведучий елемент | Час виконання, сек. | Мінімальний ведучий елемент | Час виконання, сек |
| $(1,1,\ldots1)^T$ | 50 | 1.79177e-051 | 28.5 | 5.56135311E-59 | 31.6 |
| $(1,0,\ldots1,0)^T$ | 50 | 6.7767e-054 | 29.3 | 5.56135311E-59 | 32.1 |
| $(1,-1,...1,-1)^T$ | 50 | 6.7767e-054 | 29.2 | 0.556135311E-58 | 31.8 |
| $(1,1,\ldots1)^T$ | 60 | 1.66524e-065 | 63.3 | 0.1416498617E-70 | 68.3 |
| $(1,1,\ldots1)^T$ | 100 | 6.35121e-109 | 666.2 | 0.708461361E-119 | 616.2 |

Таблиця 3

Розв'язування СЛАР з матрицею $A_2$ в точних числах

| Розмірність m | Метод Гауса з вибором максимального елементу | | Метод штучних базисних матриць | |
|---|---|---|---|---|
| | Ведучий елемент | Час виконання, сек. | Ведучий елемент | Час виконання, сек |
| 50 | 9.03917e-059 | 4.79 | 0.2040816326E-1 | 3.99 |
| 60 | 1.06745e+076 | 9.74 | 0.090909090E-1 | 6.52 |
| 61 | 6.18569e+077 | 9.41 | 0.030303030E-1 | 7.13 |
| 100 | 2.47997e+151 | 63.9 | 0.1010101010E-1 | 31.99 |
| 101 | 2.42961e+153 | 66.2 | 0.1E-1 | 31.79 |

В результаті експериментів для СЛАР з матрицею Гільберта розмірності $m$=100 та одиничною правою частиною $c_1$ (з використанням методу Штрасена) отримано точні розв'язки як методом Гауса, так і методом штучних базисних матриць, які узгоджуються з [2].

Таблиця 4

Значення мінімального ведучого елементу та часу виконання експериментів для МШБМ із СЛАР різної розмірності

| Вектор обмежень | Розмірність $m$ | Мінімальний ведучий елемент | Час виконання, секунд | |
|---|---|---|---|---|
| | | | Попередня версія Longnum | Нова версія Longnum |
| $(1,1,\ldots1)^T$ | 50 | 5.56135311E-59 | 494.36 | 31.6 |
| $(1,1,\ldots1)^T$ | 60 | 0.1416498617E-70 | 538.92 | 68.3 |
| $(1,1,\ldots1)^T$ | 100 | 0.708461361E-119 | 1498.48 | 616.2 |
| $(1,1,\ldots1)^T$ | 120 | 0.8034028680E-143 | 3090.72 | 1397.4 |

Таким чином, таблиця 4 демонструє, що для МШБМ із СЛАР із матрицею Гільберта із зростанням розмірності зменшується мінімальний ведучий елемент, що пов'язано з властивістю поганої обумовленості матриці Гільберта. Крім того, збільшується час виконання обчислень, що обумовлюється як зростанням кількості арифметичних операцій над раціональними числами, так і зростанням мантис чисельників та знаменників раціональних чисел. Слід зазначити, що за даними порівняння таблиці 4, час виконання обчислювальних експериментів суттєво менший для нової версії бібліотеки Longnum, яка є удосконаленням класу longint3.

**6. Обговорення результатів обчислювального експерименту**

Застосування нового класу longint3 дозволило (табл. 4):

1) прискорити точні обчислення в 2.5 рази: СЛАР з матрицею Гільберта розмірності 100 була розв'язана МБМ в точних числах за 1397.4 секунд, тоді як зі старим класом longint2 на тій самій ЕОМ час виконання становив 3731.94 секунди [22];

2) розв'язувати погано обумовлені СЛАР великої розмірності.

В порівнянні із старою версією бібліотеки точних обчислень Longnum в новій версії було зроблено наступне:

– оптимізовані реалізації арифметичних операцій з довгими цілими числами з врахуванням нового стандарту мови програмування С++ 17;

– застарілі функції (itoa, strcpy, strcat, sprintf) стандартних бібліотек С++ замінені новими, більш ефективними і безпечними аналогами.

Таким чином, перевагою використання нової версії Longnum в порівнянні з попередньою версією [22] є більша ефективність обчислень.

Недоліком використання точних обчислень, зокрема, бібліотеки Longnum, при розв'язанні СЛАР є те, що обсяг обчислень суттєво залежить не тільки від розмірності СЛАР, але й властивостей (обумовленості) самої матриці СЛАР (табл. 2–4).

МШБШ в поєднанні з технологією точних обчислень бібліотеки Longnum може бути особливо корисним як в наукових дослідженнях властивостей СЛАР та СЛАН, математичному моделюванні, так і прикладних дослідженнях науки і техніки, які пов'язані з необхідністю розв'язання погано обумовлених СЛАР.

Наведене дослідження є розвитком підходів до організації, виконання основних операцій (представлених в роботах [8, 14]. Зокрема, технологію високоточних обчислень з раціональними числами, яку поєднано з контролем обумовленості системи за схемою алгоритму МБМ, та удосконалено програмну реалізацію (організацію типів даних).

**7. Висновки**

Результати обчислювального експерименту по реалізації алгоритмів проведення високоточних обчислень (технології довгої арифметики) з використанням методу штучних базисних матриць дають підстави стверджувати що:

1. Розроблено та реалізовано алгоритм СЛАР (лінійної системи), в якому ефективно застосовано значення ведучого елемента методу та елементів оберненої матриці для проведення контролю обумовленості системи (виняткова властивість елементів МБМ).

2. Реалізовано: типи довгих цілих чисел longint3 та відповідних раціональних чисел longrat3 із швидкими операціями множення, ділення, побудованих на сучасних алгоритмах, що перешкоджає накопиченню похибок в ході ітерацій (додаткове оснащення алгоритму МБМ).

Розроблено представлення (для проведення обчислень) раціональних чисел як пари: чисельник та знаменник типу longint3 і оформлено у вигляді класу longrat3 (удосконалено у порівнянні з попередньою версією алгоритму); досягнуто прискорення операції множення та ділення довгих чисел (для нової пропонованої версії бібліотеки Longnum в порівнянні з попередньою версією).

Слід зазначити, що додаткові можливості проведення контролю обчислень та заходи по підвищенню точності виконання операцій технології довгої арифметики обумовили, вцілому, деяке уповільнення роботи алгоритму та наклали обмеження на розмірності розв'язуваних СЛАР.

Встановлено, що розроблені алгоритми можуть слугувати як тестувальні програми для перевірки точності проведення обчислень за іншими алгоритмами для задач середньої розмірності.

**Література**

1. Каханер Д., Моулер К., Нэш С. Численные методы и программное обеспечение. М.: Мир, 2001. 575 с.

2. Деммель Дж. Вычислительная линейная алгебра. Теория и приложение. М.: Мир, 2001. 430 с.

3. Han D., Zhang J. A comparison of two algorithms for predicting the condition number // Sixth International Conference on Machine Learning and Applications (ICMLA 2007). 2007. doi: https://doi.org/10.1109/icmla.2007.8

4. Ebrahimian R., Baldick R. State Estimator Condition Number Analysis // IEEE Power Engineering Review. 2001. Vol. 21, Issue 5. P. 64–64. doi: https://doi.org/10.1109/mper.2001.4311389

5. Nishi T., Rump S., Oishi S. A consideration on the condition number of extremely ill-conditioned matrices // 2013 European Conference on Circuit Theory and Design (ECCTD). 2013. doi: https://doi.org/10.1109/ecctd.2013.6662260

6. BLAS (Basic Linear Algebra Subprograms). URL: http://www.netlib.org/blas/sblat1

7. Li H., Yang H., Shao H. A note on the perturbation analysis for the generalized Cholesky factorization // Applied Mathematics and Computation. 2010. Vol. 215, Issue 11. P. 4022–4027. doi: https://doi.org/10.1016/j.amc.2009.12.009

8. The GNU Multiple Precision Arithmetic Library. URL: https://gmplib.org/

9. Multiple Precision Integer Library (MPI). URL: https://github.com/servo/nss/tree/master/lib/freebl/mpi

10. OpenSSL Cryptographic Toolkit. URL: http://openssl.org

11. Large Integer Package. URL: https://github.com/luckyaibin/BigInt/tree/master/freelip

12. Denis T., Rose G. BigNum Math. Implementing Cryptographic Multiple Precision Arithmetic. Syngress, 2006. 291 p. doi: https://doi.org/10.1016/b978-1-59749-112-9.x5000-x

13. Галовиц Я. C++17 STL. Стандартная библиотека шаблонов. СПб.: Питер, 2018. 432 с.

14. Анализ свойств линейной системы методом искусственных базисних матриц / Кудин В. И., Ляшко С. И., Хритоненко Н. М., Яценко Ю. П. // Кибернетика и системный анализ. 2007. № 4. С. 119–127.

15. Богаенко В. А., Кудин В. И., Скопецкий В. В. Анализ компьютерных схем метода базисных матриц // Компьютерная математика. 2009. № 2. С. 3–13.

16. Богаенко В. А., Кудин В. И., Скопецкий В. В. Анализ вычислительных схем моделирования процессов геогидродинамики // Пробл. упр. и информатики. 2009. № 4. С. 62–72.

17. Богаенко В. А., Кудин В. И., Скопецкий В. В. Об особенностях организации вычислений на основе метода базисных матриц // Кибернетика и системный анализ. 2012. Т. 48, № 4. С. 146–155.

18. Bogainenko V., Kudin V. Building preconditioners using basis matrix method // International journal Information Content and Processing. 2014. Vol. 1, Issue 2. P. 182–187.

19. Кнут Д. Искусство программирования. Т. 2. 3-е изд. М.: Издательский дом «Вильямс», 2000. 788 с.

20. Крэндалл Р., Померанс К. Простые числа: Криптографические и вычислительные аспекты. М.: УРСС, 2011. 664 с.

21. Страуструп Б. Язык программирования C++. Специальное издание. СПб.-М.: «Невский диалект» - «БИНОМ», 2006. 1104 с.

22. Кудін В. І., Оноцький В. В. Розвинення технології довгої арифметики при побудові алгоритмів дослідження задачі лінійного програмування // Журнал обчислювальної та прикладної математики. 2011. № 1. С. 77–84.