Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Дипломна робота
магістра

з теми: «ДОСЛІДЖЕННЯ ІТЕРАЦІЙНИХ МЕТОДІВ РОЗВ’ЯЗУВАННЯ СЛАР У ВИПАДКАХ РОЗРІДЖЕНИХ ДАНИХ»

Виконав: студент групи KN1-M22
спеціальності 122 Комп’ютерні науки
Гончар Фаррух Шуфратович

Керівник:
Моцик Р.В., кандидат педагогічних наук,
доцент кафедри комп’ютерних наук

Рецензент:
Сморжевський Ю.Л., кандидат
педагогічних наук, доцент,
завідувач кафедри математики

Кам’янець-Подільський – 2023

ЗМІСТ

ВСТУП

Якщо взяти будь-яку задачу Коші, наприклад, для звичайного диференціального рівняння першого порядку

$$y'(x) = f(x)$$

з початковою умовою

$$y(x_0) = y_0,$$

то навіть у цьому простому випадку методом Ейлера із деяким кроком h ми прийдемо до системи лінійних алгебраїчних рівнянь

$$y_{i+1} - y_i = hf(x_i), i = \overline{0, n-1}.$$

В прикладних задачах, математичні моделі яких описуються більш складними функціональними залежностями у вигляді систем диференціальних рівнянь чи нерівностей, методом сіток ми також одержимо систему лінійних алгебраїчних рівнянь, але, цілком очевидно, що в таких випадках головна матриця буде мати набагато складнішу структуру.

Якщо проаналізувати її структуру, то тут важливо відмітити, що в обох випадках головна матриця буде дуже розрідженою. Практично завжди при побудові математичних моделей подібного характеру в кожному рядку із тисячами невідомих не більше десяти коефіцієнтів матимуть ненульові значення.

Зокрема, в наведеному вище прикладі головна матриця матиме діагональну структуру: по головній діагоналі стоятимуть -1, а над нею буде діагональ, де стоятимуть лише 1. Тобто лише два коефіцієнти в кожному рядку головної матриці будуть не нульовими хоча n (кількість рівнянь і невідомих) може бути як завгодно великим.

В наведеному прикладі матриця є структурованою – ненульові елементи не розміщені хаотично, а у більш складних задачах це не спостерігається.

Постає питання, якими алгоритмами найбільш ефективно розв’язувати систему лінійних алгебраїчних рівнянь з такою головною матрицею.

Виходячи з цього, можна зробити висновок, що тема кваліфікаційної роботи є досить актуальною.

Відповідно до цього була сформульована і **мета кваліфікаційної роботи** – дослідити поведінку в розумінні швидкодії ітераційних методів розв’язування системи лінійних алгебраїчних рівнянь, якщо її головна матриця має структуру розріджених даних хаотичного характеру. Поставлена мета найшла своє відображення у темі кваліфікаційної роботи: “Дослідження ітераційних методів розв’язування СЛАР у випадках розріджених даних”.

Об’єктом дослідження є – системи лінійних алгебраїчних рівнянь великої розмірності.

Предметом дослідження є – алгоритми розв’язування систем лінійних алгебраїчних рівнянь, методи та підходи пошуку оптимальних за швидкодією алгоритмів.

Досягнення поставленої мети передбачає вирішення таких *завдань*:

1. Здійснити загальну характеристику точних та наближених чисельних методів розв’язання систем лінійних алгебраїчних рівнянь.
2. Проаналізувати проблеми та особливості використання ітераційних методів.
3. Розглянути формати збереження розріджених матриць розріджених даних та провести комп’ютерний експеримент щодо застосування ітераційних алгоритмів розв’язування системи лінійних рівнянь з такими матрицями.

Методи дослідження. Для розв’язання поставлених завдань використано: теорія чисельних методів, принципи вибору ефективних алгоритмів, порівняльний аналіз.

Наукова новизна роботи:

Наукова новизна полягає у тому, що поєднано ітераційні методи, які виявляються ефективнішими для систем рівнянь великої розмірності, з оптимальними методами збереження т обробки розріджених даних.

Структура та обсяг дипломної роботи. Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатку.

В першому розділі обґрунтовано необхідність пошуку ефективніших ніж метод Гаусса алгоритмів розв'язання системи лінійних алгебраїчних рівнянь великої розмірності. Показано, що ітераційні методи мають ряд переваг:

- Час обчислення пропорційний n^2 на ітерацію, тоді як для методу Гауса – n^3. Якщо для розв’язку потрібно менше ніж n ітерацій, то втрати машинного часу будуть менші.
- Як правило похибки округлення при ітераційному методі впливають на остаточні результати значно менше, ніж при методі Гауса, оскільки при його використанні похибки не нагромаджуються.
- Ітераційний метод стає особливо зручним при розв’язуванні розріджених систем, тобто коли переважна кількість коефіцієнтів системи дорівнює 0.

У другому розділі показано, що ітераційні методи застосовуються головним чином для розв’язування складних задач великої вимірності.

Разом з тим, ефективному використанню ітераційних методів заважають ряд обставин, зокрема:

1) невідомо, який метод належить застосовувати і яким чином його можна реалізувати;

2) невідомо, як обирати ітераційні параметри, що необхідні для роботи конкретних методів, наприклад, коефіцієнт релаксації для методу послідовної верхньої релаксації.

3) неясний вибір моменту закінчення ітераційного процесу.

4) значною проблемою постає задача перетворення системи лінійних алгебраїчних рівнянь у загальній формі у збіжну ітераційну форму.

У третьому розділі було проведено цілий ряд чисельних експериментів розв’язання систем лінійних алгебраїчних рівнянь. Зокрема порівнювалася швидкодія методу Зейделя без врахування розрідженості матриці та з врахуванням розрідженості. Встановлено, що запропонований метод при великій розмірності та точності обчислення порядку 10^{-11} швидкодія запропонованого методу була кращою до 15 відсотків в порівнянні із звичними ітераційними методами без врахування розрідженості.

Одержані результати доцільно використовувати при читанні курсу “Обчислювальні методи”. Також вони будуть корисними в різного роду вибіркових дисциплінах, де звертається увага на використання ефективних чисельних методів. Цікавими можуть бути продовження досліджень з використанням розгалужених обчислень.

Апробація результатів дослідження проводилася шляхом виступу на наукових конференціях:

1. Фаррух ГОНЧАР. Обробка розріджених даних. Збірник наукових праць студентів та магістрантів Кам'янець-Подільського національного університету імені Івана Огієнка. [Електронний ресурс]. Кам’янець-Подільський: Кам’янець-Подільський національний університет імені Івана Огієнка, 2023. Вип. 17, с. 251-253.
2. Фаррух ГОНЧАР Дослідження ітераційних методів розв’язання СЛАР у випадках розріджених даних. Збірник матеріалів наукової конференції здобувачів вищої освіти фізико-математичного факультету Кам’янець-Подільського національного університету імені Івана Огієнка. 1 листопада 2023 року [Електронний ресурс]. Кам’янець-Подільський :

Кам’янець-Подільський національний університет імені Івана Огієнка, 2023, с. 30-31.

Результати досліджень опубліковано у роботі:

Гончар Ф.Ш., Моцик Р.В. Порівняльна характеристика точних та наближених чисельних методів розв’язання систем лінійних алгебраїчних рівнянь. Вісник Кам’янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Випуск 15. Кам’янець-Подільський : Кам’янець-Подільський національний університет імені Івана Огієнка, 2022, с. 12-15.

Повний обсяг дипломної роботи становить 44 сторінки, у тому числі: 38 сторінок основного тексту, список використаних джерел із 12 найменувань та 1 додатку.

РОЗДІЛ 1.
ЗАГАЛЬНА ХАРАКТЕРИСТИКА ТОЧНИХ ТА НАБЛИЖЕНИХ ЧИСЕЛЬНИХ МЕТОДІВ РОЗВ'ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ

1.1. Поняття про точні та наближені чисельні методи розв'язання систем лінійних алгебраїчних рівнянь

Досить часто доводиться мати справу з прикладними задачами, математичні моделі яких зводяться до задачі розв'язування системи лінійних алгебраїчних рівнянь. В таких моделях об'єкт апроксимується відрізками прямої а функціональні залежності між параметрами подаються у вигляді лінійних рівнянь чи нерівностей. Останні, у свою чергу, перетворюються у рівняння шляхом введення додаткових невід'ємних величин.

Такий напрямок побудови математичної моделі в свій час одержав назву метод лінеаризації.

Зазвичай методом лінеаризації одержують системи лінійних алгебраїчних рівнянь дуже великої розмірності (говорять навіть про гіпер велику розмірність). Це обумовлює проведення досліджень з ефективних методів одержання числових значень (коренів) системи лінійних алгебраїчних рівнянь.

Потреба їх розв'язування може бути як самостійною задачею (наприклад, аналіз рівноваги сил в жорсткій системі балок або дослідження умов та параметрів рівноваги хімічної реакції тощо), так і частиною більш складних задач, коли на деякому етапі їх розв'язання встановлюються лінійні залежності між певними величинами і потрібно визначити, при яких значеннях цих величин дані залежності виконуються.

В обох випадках практична цінність чисельного методу в значній мірі визначається швидкістю та ефективністю отримання розв'язку системи.

Важливе практичне значення таких систем зумовило розробку чималого арсеналу різних чисельних методів розв'язування систем лінійних рівнянь.

Виходячи з вищесказаного, констатуємо, що однією з найпоширеніших і важливих завдань обчислювальної математики є розв'язання системи лінійних алгебраїчних рівнянь (СЛАР), яка в загальному випадку записується так:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases}$$

або в матричній формі:

$$Ax=b.$$

Зазвичай вважають, що $m \leq n$, обумовлюючи цим самим, що система може мати безліч розв'язків. В таких випадках на практиці вибирають будь-який із них. Рідко коли на вільні невідомі накладають додаткові вимоги, наприклад, вони повинні бути невід'ємними.

Розглянемо найбільш вживані чисельні методи і ефективні алгоритми розв'язання систем лінійних алгебраїчних рівнянь. В своїй абсолютній більшості вони передбачають існування та єдиність розв'язку (в такому випадку систему рівнянь називають сумісною та визначеною) і поділяються на точні та наближені методи.

Перевірка умови існування та єдиності розв'язку є окремою математичною задачею. Тому, зазвичай, при описі чисельного методу ігнорують дослідженням на сумісність та визначеність системи рівнянь. Тобто припускають, що ця умова виконується. Тоді матриця A є квадратною ($m=n$).

Для розв'язання СЛАР традиційно використовують дві групи чисельних методів: точні та наближені. Таких методів побудовано досить багато. Наприклад, у відомій монографічній праці [5] їх нараховується 10, а в [2] – 13.

До точних належать метод Гауса та його модифікації, правило Крамара, метод квадратних коренів тощо. До наближених – метод ітерації, метод Зейделя та ряд інших.

Точні методи можна назвати точними лише умовно, бо округлення в процесі виконання проміжних обчислень неминуче приводяться до втрати точності кінцевого результату. Тому прийнято до точних методів відносити такі методи, які дозволяють отримати точний розв'язок системи за відповідну кількість операцій перетворення без урахування похибок округлення.

Не слід ігнорувати і той факт, що коефіцієнти головної матриці A та вектора вільних членів b, тобто початкові дані, можуть отримуватися в результаті деяких вимірювань або наближеного розв'язання деякої задачі. Тобто вони можуть бути наближеними числами. Також при виконанні проміжних обчислень доводиться проводити округлення. Це, у свою чергу, нівелює відмінність між цими двома групами (точні і наближені методи).

Отже, не варто зациклюватись лише на точних методах розв'язування СЛАР і ігнорувати наближені методи лише тому, що вони не точні.

Ще раз акцентуємо увагу на тому, що при виборі того чи іншого чисельного методу варто зосереджуватися саме на швидкості і ефективності отримання розв'язку.

На практиці майже завжди користуються методом Гауса (з різними його модифікаціями). Таку популярність цього методу можна пояснити, наприклад, його універсальністю: не потрібно перевіряти чи система має розв'язки; алгоритм сам визначає чи він існує чи ні; знаходить його, якщо розв'язок єдиний і будує множину розв'язків (фундаментальну систему розв'язків), якщо їх безліч.

1.2 Особливості методів Гауса та переваги ітераційних методів

Як уже зауважено вище, найбільш відомим з точних методів розв'язання системи лінійних алгебраїчних рівнянь є методи Гауса, суть яких полягає в тому, що система рівнянь, яка розв'язується, зводиться до еквівалентної системи з верхньою трикутною матрицею. Невідомі знаходяться послідовними підстановками, починаючи з останнього рівняння перетвореної системи. Алгоритми Гауса складаються із виконання однотипних операцій, які легко формалізуються і програмуються. Однак, точність результату й витрачений на його отримання час у більшості випадків залежить від алгоритму формування трикутної матриці системи.

Разом з тим, при великому числі невідомих у системі лінійних рівнянь, схема методу Гауса, яка дає точний розв'язок, стає досить складною а процес навіть комп'ютерної реалізації алгоритму – довготривалим. Наприклад, у [6] показано, що для розв'язання системи рівнянь розмірності $n \times n$ потрібно виконати n^3 арифметичних операцій. При розв'язуванні системи рівнянь з декількома тисячами невідомих за допомогою комп'ютера на цю операцію може піти декілька годин. У цих умовах, для розв'язку системи рівнянь, доцільніше використовувати наближені чисельні методи. Одним з таких є метод простої ітерації (його також називають методом послідовних наближень).

В літературних джерелах фахівці підкреслюють, що "точні" методи використовують при розв'язуванні на персональних комп'ютерах систем невисокого порядку ($n<10^3$, де n – число лінійних алгебраїчних рівнянь системи). Наближені методи використовують для систем високого порядку $n=10^3 \ldots 10^6$.

Переваги наближеного методу (наприклад, простих ітерацій) над точним методом (Гауса) наступні:

- Час обчислення пропорційний n^2 на ітерацію, тоді як для методу Гауса – n^3. Якщо для розв’язку потрібно менше ніж n ітерацій, то втрати машинного часу будуть менші.
- Як правило похибки округлення при ітераційному методі впливають на остаточні результати значно менше, ніж при методі Гауса, оскільки при його використанні похибки не нагромаджуються.
- Ітераційний метод стає особливо зручним при розв’язуванні систем, переважна кількість коефіцієнтів яких дорівнює 0. Рівняння, які отримують, наприклад, при розв’язуванні крайових задач, відносяться саме до цього класу.

Ітераційні методи дозволяють одержати приблизний розв’язок системи за скінченне число наближень (ітерацій) із заданою похибкою результату. Недоліком ітераційного методу є те, що він не завжди збігається.

Ітераційні методи розв'язування систем лінійних алгебраїчних рівнянь ефективно використовуються для розв'язування системи лінійних алгебраїчних рівнянь великої розмірності з розрідженими матрицями, а також для уточнення розв'язку системи лінійних алгебраїчних рівнянь, отриманого за допомогою прямого методу.

Великі масиви, зазвичай, подають у вигляді матриць: одновимірні (вектори), двовимірні (таблиці), тривимірні і, взагалі кажучи, n-вимірні в багатопараметричних задачах.

Традиційні методи обробки таких матриць в багатьох випадках є неефективні, а іноді навіть неможливі. Зокрема, ми встановили, що на домашньому комп’ютері не можна зберігати матриці розмірності 16.000x16.000. Не вистачає оперативної пам’яті.

Проводячи різного роду комп’ютерні дослідження з різними чисельними методами розв’язування СЛАР відповідно до тематики магістерської роботи було показано, зокрема, що розв’язок системи

рівнянь розмірності 12.000x12.000 методом Гауса вдалося одержати лише за 45 хвилин.

Може виникнути запитання, в яких прикладних задачах зустрічаються масиви таких розмірностей. В більшості динамічних задач математичної фізики доводиться дискретизувати значення параметричних функцій з малим кроком, що призводить до стрімкого зростання розміру масивів.

Незважаючи на інтенсивний розвиток грід-мереж та хмарних обчислень на сьогодні залишається актуальною проблема пошуку потужних комп'ютерних ресурсів для математичних розрахунків різного роду таких наукових задач, адже сьогодні розвиток фізичного пізнання охоплює все нові і нові сфери дійсності. Зрозуміло, цей процес не позбавлений численних проблемних аспектів, пов'язаних, зокрема, з можливістю використання сучасних комп'ютерних технологій у дослідженні фізичних процесів. Чисельні перешкоди на шляху експериментальних досліджень фізичних процесів зумовили заговорити про їх складність. У зв'язку з цим все частіше обговорюється спеціальна тема – фізика складних систем.

Постає потреба у високопродуктивних обчисленнях за зниження затрат, адже побудова сучасного кластера чи грід-мережі потребує значних витрат в тому числі і фінансових.

1.2. Дещо про схему Холецького

В багатьох випадках вдається суттєво прискорити процес розв'язання системи лінійних рівнянь, використовуючи схему Холецького (*L-P* факторизацію).

Відразу варто зауважити, що розклад Холецького ефективніший за метод Гауса у випадку симетричної матриці.

Основна ідея методу полягає у замінні матриці A на добуток двох трикутних матриць L і P. Тоді алгоритм складається з трьох етапів. На першому етапі визначають коефіцієнти матриць L і P.

Виходячи із співвідношення $Ax=b$, відповідно до заміни матриці A на добуток двох матриць одержимо $(LP)x=b$. Далі скористаємось асоціативністю множення матриць: $(LP)x = L(Px) = b$.

Якщо позначити $Px = y$, то одержимо систему лінійних рівнянь $Ly = b$, яку розв'язують на другому етапі. Оскільки матриця, зазвичай, L – нижня трикутна, то процес розв'язування значно спрощується.

Одержавши на другому етапі координити вектора y, на завершальному третьому етапі розв'язують систему лінійних рівнянь $Px = y$. Знову ж таки, оскільки матриця P трикутна, то це спрощує процес відшукання коренів.

У випадку симетричної матриці A використання схеми Холецького розв'язання системи лінійних алгебраїчних рівнянь модифікується у метод квадратного кореня.

У випадку трьохдіагональної матриці A використання схеми Холецького розв'язання системи лінійних алгебраїчних рівнянь модифікується у метод прогонки.

Можна ще говорити про про ефективність схеми Холецького і для п'ятидіагональних матриць. При більшій розмірності доведеться чимало зусиль приділити математичному виведенню ітераційних формул, хоча це не дивлячись на рутинну роботу не так і складно, але, з іншої сторони, при чисельному розв'язуванні диференціальних рівнянь математичної фізики, де найчастіше зустрічаються матриці такого виду (багатодіагональні), обмежуються рівняннями другого порядку і більш ніж п'ятидіагональні матриці не розглядаються.

В більшості, навіть у задачах математичної фізики, накладаються додаткові умови і втрачається системність розміщення даних. Хаотичний

характер їх розташування зумовлює розробляти алгоритми обробки розріджених даних.

В більшості випадків вони є досить простими і легко програмуються. Найбільше труднощів викликає програмування суми векторів (а отже і матриць), але при високих ступенях розрідженості, як показали чисельні експерименти, ці алгоритми набагато ефективніші за традиційні методи обробки масивів.

Таким чином, для ефективної роботи у великих масивах доцільно використовувати структуру розміщення ненулевих даних: симетричність чи діагональність матриць, а при великих ступенях розрідженості – модифіковані алгоритми обробки закодованих даних.

Висновок до першого розділу. Досить часто при розв’язуванні прикладних математичних задач з обчислення у великих масивах доводиться мати справу з системами лінійних алгебраїчних рівнянь великої розмірності. Як відомо, метод Гаусса з головною матрицею порядку n потребує виконання n^3 алгебраїчних операцій. Для порівняння навіть сучасний потужний персональний комп’ютер, маючи потужність в 50 гігафлопсів, тобто виконуючи 50×10^9 алгебраїчних операцій в секунду, систему в 10 000 рівнянь та невідомих може розв’язати за 20 секунд.

Тому актуальним постає питання пошуку ефективніших ніж метод Гаусса алгоритмів розв'язання системи лінійних алгебраїчних рівнянь великої розмірності.

Це зумовило нас у кваліфікаційній роботі ігнорувати класичні точні методи розв’язування СЛАР (методи Гауса та метод-схема Холецького) і досліджувати ітераційні методи, які, як зауважено вище, мають ряд переваг:

- Час обчислення пропорційний n^2 на ітерацію, тоді як для методу Гауса – n^3. Якщо для розв’язку потрібно менше ніж n ітерацій, то втрати машинного часу будуть менші.

- Як правило похибки округлення при ітераційному методі впливають на остаточні результати значно менше, ніж при методі Гауса, оскільки при його використанні похибки не нагромаджуються.
- Ітераційний метод стає особливо зручним при розв’язуванні розріджених систем, тобто коли переважна кількість коефіцієнтів системи дорівнює 0.

РОЗДІЛ 2.
ОСОБЛИВОСТІ ВИКОРИСТАННЯ ІТЕРАЦІЙНИХ МЕТОДІВ

2.1. Проблеми застосування ітераційних методів

Ітераційні методи розв'язання СЛАУ - це методи наближеного розв'язування, що базуються на послідовному наближенні до розв'язку шляхом багатократного застосування деякої обчислювальної процедури, при цьому вихідними даними для кожної наступної процедури є результати застосування попередніх процедур. Наслідком такого ітераційного процесу є послідовність, яка при виконанні деяких умов збігається до розв'язку задачі.

Для систем середньої вимірності часто, як зауважено вище, більш привабливими є прямі методи або точні методи. Ітераційні методи застосовуються головним чином для розв'язування складних задач великої вимірності, для яких внаслідок обмежень, що накладаються на об'єм робочої пам'яті і число арифметичних операцій, використання прямих методів виявляється достатньо важким або значно менше ефективним. Наприклад, ітераційні методи, як правило, застосовуються для розв'язання задач з трьома просторовими змінними, задач, що включають системи нелінійних рівнянь, задач, що виникають при дискретизації систем рівнянь в частинних похідних, а також для розв'язування нестаціонарних задач з більш ніж однією просторовою змінною.

Формулювання і застосування ітераційних методів потребують спеціальних знань і деякого досвіду.

Ефективному використанню ітераційних методів заважають три обставини:

1) невідомо, який метод належить застосовувати і яким чином його можна реалізувати;

2) невідомо, як обирати ітераційні параметри, що необхідні для роботи конкретних методів, наприклад, коефіцієнт релаксації для методу послідовної верхньої релаксації.

3) неясний вибір моменту закінчення ітераційного процесу.

Мабуть найголовнішою проблемою виступає задача перетворення системи лінійних алгебраїчних рівнянь у загальній формі у збіжну ітераційну форму.

Внаслідок великого різноманіття задач та ітераційних процедур повністю позбавитися цих невизначеностей неможливо.

Вибір ефективного ітераційного методу розв'язування конкретної задачі залежить від її характерних властивостей та від архітектури обчислювальної машини, на якій буде розв'язуватися задача. З огляду на це, жодних загальних правил вибору найкращого методу розв'язування не існує. Проте, знання порівняльних характеристик ряду ітераційних процедур загального вигляду може суттєво спростити проблему.

Отже, підхід до вибору методу має ґрунтуватися на аналізі теоретичних та обчислювальних принципів загальних ітераційних методів. Ці принципи потім можна реально використовувати при виборі ефективного ітераційного методу.

2.2. Класи ітераційних методів

Розглянемо клас ітераційних методів, призначених для розв'язування лінійних систем

$$Ax = b\,, \tag{2.1}$$

де A – задана дійсна невироджена матриця розміру $n \times n$, а b – заданий вектор-стовпець, що складається з n дійсних компонент.

Всі методи, що будуть нами розглянуті, є лінійними стаціонарними методами першого порядку, які можуть бути записані в такому вигляді:

$$x_{k+1} = Gx_k + F,\ k = 0, 1, 2,... \tag{2.2}$$

де G – дійсна матриця переходу даного методу розміру $n \times n$, а F – відповідний відомий вектор.

Такий метод є методом першого порядку, оскільки наближення x_{k+1} залежить явно лише від x_k, але не залежить явно від x_{k-1}, x_{k-2},..., x_0.

Метод є лінійним, оскільки ані матриця G, ані вектор F не залежать від x_k.

Цей метод є стаціонарним, оскільки ані G, ані F не залежать від номера ітерації n .

В подальшому до основних ітераційних методів ми будемо відносити будь-який ітераційний метод виду (2.2). До найбільш відомих основних ітераційних методів належать: метод Річардсона (RF), метод Якобі (J), метод Гаусса-Зейделя (GZ), метод послідовної верхньої релаксації (SOR) і симетричної послідовної верхньої релаксації (SSOR).

Всюди будемо припускати, що

$$G = I - Q^{-1}A,\ F = Q^{-1}b \tag{2.3}$$

для деякої невиродженої матриці Q. Така матриця Q називається матрицею розщеплення. Із припущення (2.3) і того факту, що матриця A є невиродженою, витікає, що x є розв'язком суміжної системи

$$(I - G)\,x = F \tag{2.4}$$

тоді і тільки тоді, коли x є також розв'язком системи (1), тобто

$$x^* = A^{-1}\,b. \tag{2.5}$$

Ітераційний метод (2.2), суміжна система (2.4) для якого має єдиний розв'язок x, що співпадає із розв'язком (2.1), називається цілком узгодженим. Якщо $\{x_k\}$ є послідовністю ітераційних наближень, що визначаються за допомогою (2.2), то із властивості цілковитої (повної) узгодженості витікає, що

1) якщо $x_k = x^*$ для деякого k , то $x_{k+1} = x_{k+1} = \ldots = x^*$ і
2) якщо послідовність $\{x_k\}$ збігається до деякого вектора x^{e}, то $x^* = x^{\text{e}}$.

Завжди будемо припускати, що основний ітераційний метод (2.2) є цілком узгодженим, оскільки наявність цієї властивості є істотною для

будь-якого розумного методу. Іншою властивістю основних ітераційних методів, яку ми не завжди будемо заздалегідь припускати, є їхня збіжність. Кажуть, що метод (2.2) збігається, якщо для будь-якого початкового наближення x_0 послідовність x_1, x_2, ... , що визначається за допомогою (2.2), збігається до x^*.

Необхідна і достатня умова збіжності має вигляд

$$S(G) < 1 \tag{2.6}$$

Для вимірювання швидкості збіжності лінійного стаціонарного ітераційного методу (2.2) визначимо вектор похибки ε_k :

$$\varepsilon_k = x_k - x^* . \tag{2.7}$$

Використовуючи (2.2) і той факт, що x також задовольняє суміжну систему (2.2), отримуємо, що

$$\varepsilon_k = G\,\varepsilon_{k-1} = G_k\,\varepsilon_0. \tag{2.8}$$

Доведення. $x_k = Gx_{k-1} + F = Gx_{k-1} + (I - G)\,x^* = G(x_{k-1} - x^*) + x^* \Rightarrow$

$\Rightarrow x_k - x^* = = G(x_{k-1} - x^*) \Rightarrow \varepsilon_k = G\,\varepsilon_{k-1} = \ldots = G_k\,\varepsilon_0.$

Таким чином, для будь-якої векторної норми α, $\alpha = 2,\infty$ і відповідної матричної норми α, $\alpha = 2,\infty$ внаслідок властивості

$$\|Av\|_\alpha \le \|A\|_\alpha \,\|v\|_\alpha$$

маємо

$$\|\varepsilon_k\|_\alpha = \le G\,\varepsilon_{k-1} = \|G_k\|_\alpha \,\|\varepsilon_0\|_\alpha. \tag{2.9}$$

Отже, величина $\|G_k\|_\alpha$ визначає, в скільки разів було зменшено норму похибки після k ітерацій.

Визначимо середню швидкість збіжності методу (2.2) в такий спосіб:

$$R_k(G) \equiv -\ln \|G_k\|_\alpha / k. \tag{2.10}$$

Можна показати, що якщо $S(G) < 1$, то

$$\lim_{k\to\infty} (\|G_k\|_\alpha)^{\frac{1}{k}} = S(G) . \tag{2.11}$$

Отже, ми прийшли до визначення асимптотичної швидкості збіжності

$$R_\infty(G) \equiv \lim_{k\to\infty} R_k\,(G) = -\ln S(G). \qquad (2.12)$$

Зауважимо, що в той час, як $R_k(G)$ залежить від конкретного виду норми α , величина $R_\infty(G)$ не залежить від α . Часто $R_\infty(G)$ називають просто швидкістю збіжності. Якщо $S(G) < 1$, то грубе наближення для кількості ітерацій, що потрібні для зменшення норми вектора початкової похибки в ζ^{-1} разів, визначається за формулою

$$k \approx -\frac{\ln\zeta}{R_\infty(G)} \qquad (2.13)$$

Для більшості методів прискорення наявність збіжності основного методу (2) не є необхідною. Як правило, виявляється достатньо, щоб метод був симетризованим.

Означення. Ітераційний метод (2.2) є симетризованим, якщо для деякої невиродженої матриці W матриця $W(I-G)W^{-1}$ виявляється симетричною і додатно визначеною.

Така матриця W називається матрицею симетризації.

Ітераційний метод, що не є симетризованим, називається несиметризованим.

Теорема. Якщо ітераційний метод (2.2) є симетризованим, то

1) власні значення матриці G є дійсними;
2) алгебраїчно найбільше власне число λ_{max} матриці G менше одиниці;
3) множина власних значень матриці G містить базис відповідного векторного простору.

Численні ітераційні методи є симетризованими. Наприклад, основний метод (2.2) є симетризованим, якщо матриця A і матриця розщеплення Q в (3) будуть симетричними і додатно визначеними. В цьому випадку матриці $A^{\frac{1}{2}}$ і $Q^{\frac{1}{2}}$ є матрицями симетризації.

Для симетричної додатно визначеної матриці A квадратним коренем $A^{\frac{1}{2}}$ називається матриця V: $V\times V = A$.

Оскільки симетрична матриця A завжди може бути записаною у вигляді

$$A = PDP^T,$$

де D =diag(λ_1, λ_2, ..., λ_n) – діагональна матриця, діагональні елементи якої дорівнюють власним числам A, а P – ортогональна матриця, стовпчики якої є власними векторами матриці A, то

$$A^{\frac{1}{2}} = PD^{\frac{1}{2}}P^T,$$

де D = diag($\sqrt{\lambda_1}$, $\sqrt{\lambda_2}$, ..., $\sqrt{\lambda_n}$) – діагональна матриця, діагональні елементи якої дорівнюють додатним квадратним кореням із власних чисел матриці A.

Крім того, матрицею симетризації є також будь-яка матриця W, така що

$$Q = W^T W.$$

Зауважимо, що із властивості симетризованості не обов’язково випливає властивість збіжності.

Якщо ітераційний метод (2.2) є симетризованим, то власні числа матриці G виявляються меншими за одиницю, але не обов’язково менше одиниці за модулем.

Отже, умова збіжності не завжди виконується. Проте завжди існує так званий екстрапольований метод, заснований на (2.2), який є збіжним, коли основний метод є симетризованим.

Екстрапольований метод визначається так:

$$x_{k+1} = \gamma\,(Gx_k + F) + (1 - \gamma)\,x_k = G_\gamma\,x_k + \gamma F, \qquad (2.14)$$

де

$$G_\gamma \equiv \gamma\,G + (1 - \gamma)I \qquad (2.15)$$

Тут γ – параметр, який часто називають "коефіцієнтом екстраполяції". Якщо ітераційний метод є симетризованим, то оптимальне значення γ^* для параметра γ в сенсі мінімізації $S(G_\gamma)$ визначається за формулою

$$\gamma^* = 2/(2-\lambda_{max}(G) - \lambda_{min}(G)), \quad (16)$$

де $\lambda_{max}(G)$ і $\lambda_{min}(G)$ найменшим і найбільшим власним числом матриці G, відповідно.

Легко показати, що

$$S(G_\gamma)=\frac{\lambda_{max}-\lambda_{min}}{2-\lambda_{max}-\lambda_{min}} = \frac{1}{\frac{2}{\lambda_{max}-\lambda_{min}}-\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}}.$$

Оскільки $\lambda_{max}+\lambda_{min} < 2$, внаслідок того факту, що обидва вони менші одиниці, маємо

$$\frac{2}{\lambda_{max}-\lambda_{min}} - \frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}} > 0.$$

Покажемо, що

$$\frac{2}{\lambda_{max}-\lambda_{min}} - \frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}} > 1.$$

Це еквівалентно тому, що

$$\frac{2}{\lambda_{max}-\lambda_{min}} > 1+\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}} = \frac{2\lambda_{min}}{\lambda_{max}-\lambda_{min}},$$

а це, в свою чергу, рівносильно умові $\lambda_{max} < 1$.

З цього випливає, що оптимальний екстрапольований метод, що визначається співвідношенням

$$x_{k+1} = G_{\gamma^*} x_k + \gamma^* F, \quad (2.17)$$

є збіжним.

2.3. Порівняння прямих та ітераційних методів

Як правило, великі розріджені системи пов'язані з деякою різницевою сіткою. Нехай $1/m$ – величина кроку сітки. Тоді у тривимірному випадку типовою є стрічкова матриця із шириною стрічки $\approx m^2 = n^{2/3} \Rightarrow n = m^3$. Оцінювана кількість операцій із числами з плаваючою комою (флопів) при застосуванні методу Гауса в типовому випадку дорівнює $O(nm^4) \approx n^{7/3}$.

Для двовимірної ситуації ширина стрічки дорівнює $\approx n^{1/2}$, так що число флопів для прямого методу змінюється як n^2.

1. Якщо потрібно розв'язувати велику кількість систем з різними правими частинами, треба лише один раз привести матрицю до трикутного вигляду, отже кількість операцій при розв'язанні кожної системи змінюється як $n^{5/3}$ для тривимірної задачі та $n^{3/2}$ для 2-вимірної задачі.

2. Для симетричної додатно визначеної системи зменшення похибки за одну ітерацію методу спряжених градієнтів приблизно дорівнює

$$\frac{\sqrt{k}-1}{\sqrt{k}+1},$$

де $k = \|A\|_2 \|A^{-1}\|_2$.

При дискретизації диференціальних рівнянь в частинних похідних другого порядку на сітці з кроком $1/m$, як правило, $k \approx m^2$. Отже, для 3-вимірної задачі ми отримуємо $k \approx n^{2/3}$, а для 2-вимірних задач: $k \approx n$.

Для зменшення похибки до рівня ε необхідно, щоб

$$\left(\frac{1-\frac{1}{\sqrt{k}}}{1+\frac{1}{\sqrt{k}}}\right)^j \approx \left(1-\frac{2}{\sqrt{k}}\right)^j \approx e^{\frac{2j}{\sqrt{k}}} < \varepsilon$$

де j – номер ітерації.

Для тривимірної задачі

$$j \approx -\frac{ln\varepsilon}{2}\sqrt{k} \approx -\frac{ln\varepsilon}{2}\sqrt[3]{n},$$

а для 2-вимірної –

$$j \approx -\frac{ln\varepsilon}{2}\sqrt{n}.$$

Якщо ми припустимо, що кількість флопів на одну ітерацію приблизно дорівнює fn (де f — кількість ненульових елементів в рядку матриці), то кількість флопів, яка потрібна для зменшення похибки до значення, меншого ε , дорівнює

1) для тривимірної задачі $f_\varepsilon \approx -fn^{4/3} ln\varepsilon$;

2) для двовимірної задачі $f_\varepsilon \approx -fn^{3/2}$.

Отже, маємо висновок: якщо ми повинні розв'язувати одну систему, то при великому n , або маленькому f , або помірному ε ітераційні методи

можуть бути більш привабливими; якщо ми маємо розв'язувати подібні системи із різними правими частинами і якщо ми припускаємо, що їх кількість настільки велика, що кількість операцій для декомпозиції A відносно мала, то для 2-вимірного випадку прямі методи можуть бути більш ефективними.

В той же час, для 3-вимірного випадку це є сумнівним, оскільки кількість флопів для прямого розв'язання СЛАУ є величиною порядку $\approx n^{5/3}$, а для ітераційного розв'язання – $\approx n^{4/3}$

Приклад. Horst Simon оцінив час, який потрібен для розв'язання системи з 5×10^9 невідомими найбільш ефективним і економічним прямим методів, відомим на даний час, як 520040 років при обчисленнях із швидкістю 1 TFLOP/сек. З іншого боку, при використанні передобумовленого методу спряжених градієнтів з тією ж швидкістю обчислень оцінюваний час дорівнює 575 сек.

Крім того, вимоги до розмірів пам'яті при використанні ітераційних методів, як правило, менші. Часто це є аргументом для використання ітераційних методів в 2-вимірних задачах, коли кількість флопів для обох класів методів є приблизно однаковими.

Зауваження. При відповідному передобумовленні ми можемо отримати $\sqrt{k} \approx n^{1/6}$, і тоді кількість флопів може стати рівною $- fn^{7/6}ln\varepsilon$. Для спектральних задач найшвидшими можуть бути спеціальні методи, наприклад, багатосіткові. Вимоги до пам'яті роблять ітераційні методи більш привабливими. Для матриць, які не є симетричними додатно визначеними, ситуація може бути більш проблематичною: часто важко знайти відповідний і придатний ітераційний метод або передобумовлення. Проте, проекційні методи, такі як GMRES, Bi-CG, CGS і Bi-CGSTAB часто мають кількість флопів, яка близька до CG. Ітераційні методи можуть бути привабливими навіть для щільних матриць. Якщо матриця є симетричною, додатно визначеною і число обумовленості дорівнює $n^{2-2\varepsilon}$, то кількість флопів на ітерацію дорівнює $\approx n^2$, а кількість ітераційних

кроків $\approx n^{1-\varepsilon}$. Отже, загальна кількість флопів грубо оцінюється як $n^{3-\varepsilon}$ і це є асимптотично менше, ніж в методі Холецького, де $f_\varepsilon \approx n^3$.

Висновок до другого розділу. Ітераційні методи застосовуються головним чином для розв'язування складних задач великої вимірності. Формулювання і застосування ітераційних методів потребують спеціальних математичних знань і деякого досвіду.

Ефективному використанню ітераційних методів заважають ряд обставин, зокрема:

1) невідомо, який метод належить застосовувати і яким чином його можна реалізувати;

2) невідомо, як обирати ітераційні параметри, що необхідні для роботи конкретних методів, наприклад, коефіцієнт релаксації для методу послідовної верхньої релаксації.

3) неясний вибір моменту закінчення ітераційного процесу.

На практиці значною проблемою постає задача перетворення системи лінійних алгебраїчних рівнянь у загальній формі у збіжну ітераційну форму. Тому вирішенню цієї задачі присвячено ряд досить глибоких теоретичних досліджень у математиці. Виділено цілі класи для головної матриці.

Ітераційні методи можуть бути привабливими навіть для щільних матриць. Для розріджених матриць варто користуватися лише ітераційними методами.

Якщо матриця має ряд додаткових властивостей, наприклад, є симетричною, то кількість ітераційних кроків може суттєво знижуватися.

РОЗДІЛ 3.
ЗАСТОСУВАННЯ ІТЕРАЦІЙНИХ АЛГОРИТМІВ РОЗВ'ЯЗУВАННЯ СИСТЕМИ ЛІНІЙНИХ РІВНЯНЬ

3.1. Алгоритм простої ітерації

Будемо вважати, що система рівнянь подана у вигляді

$$\begin{cases} x_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + b_1; \\ x_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + b_2; \\ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + b_n \end{cases}$$

або в матричній формі $x = Ax + b$.

Щоб переконатися, що ітераційний метод буде збіжним перевіримо спочатку, чи виконується хоча б одна із умов:

1) $\max\limits_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| < 1,$

2) $\max\limits_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| < 1,$

3) $\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}^2| < 1.$

За початкове (нульове) наближення приймемо стовпець вільних членів, тобто вектор $x_0 = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}$.

Далі, послідовно будуємо вектори-стовпці $x_1 = Ax_0 + b$ (перше наближення), $x_2 = Ax_1 + b$ (друге наближення), $x_3 = Ax_2 + b$ (третє наближення) і так далі.

Ітераційний процес методу простої ітерації будемо продовжувати до тих пір, поки не буде виконуватись умова $| x_{k+1} - x_k | < \varepsilon$, де ε – задана точність обчислювального процесу. Під $| x_{k+1} - x_k |$ будемо розуміти максимальне відхилення відповідних координат, тобто

$$| x_{k+1} - x_k | = \max_{1 \le i \le n} \left(x_i^{k+1} - x_i^{k+1} \right).$$

3.2. Алгоритм методу Зейделя та верхньої релаксації

Як і в попередньому випадку (в методі простої ітерації) будемо вважати, що система рівнянь подана у вигляді, який є зручним для ітераційних алгоритмів:

$$\begin{cases} x_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + b_1; \\ x_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + b_2; \\ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + b_n. \end{cases}$$

Якщо система подана в загальній формі, то спочатку доведеться перетворити її до такого вигляду одним із описаних раніше методів (див. розділ 2).

Знову ж таки, перш ніж користуватися ітераційним алгоритмом (в даному випадку алгоритмом Зейделя) спочатку потрібно переконатися, що ітераційний метод буде збіжним, тобто чи виконується хоча б одна із умов:

1) $\max\limits_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| < 1,$

2) $\max\limits_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| < 1,$

3) $\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}^2| < 1.$

Як і в попередньому випадку, традиційно, за початкове (нульове) наближення приймемо стовпець вільних членів, тобто вектор

$$x_0 = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}.$$

Будемо позначати через x_i^k i-ту координату шуканого вектора x, знайдену на k-тому кроці, а через x_i^{k+1} – на k+1-тому кроці.

Оскільки ітераційний процес є наближеним, то на старті (нульовому кроці) потрібно також задати умову його завершення. Традиційно вважають, що розв’язок в ітераційному процесі буде знайдено з необхідною точністю, якщо буде виконуватись умова $| x_{k+1} - x_k | < \varepsilon$, де ε – задана точність обчислювального процесу.

Можна по різному розуміти вираз $| x_{k+1} - x_k |$, наприклад, корінь квадратний і суми квадратів покоординатних різниць, але, з позицій складності обчислювального процесу, краще перевіряти чи кожна з покоординатних різниць менша ε, тобто покладатимемо

$$| x_{k+1} - x_k | = \max_{1 \le i \le n} \left(x_i^{k+1} - x_i^{k+1} \right).$$

Маючи стартове значення для вектора x, тобто задавши x_0, за методом Зейделя на k+1-му обчислення будемо проводити, користуючись наступними співвідношеннями:

$$\begin{cases} x_1^{k+1} = a_{11} x_1^k + a_{12} x_2^k + a_{13} x_3^k + \ldots + a_{1n} x_n^k + b_1; \\ x_2^{k+1} = a_{21} [\![x_1^{k+1}]\!] + a_{22} x_2^k + a_{23} x_3^k + \ldots + a_{2n} x_n^k + b_2; \\ x_3^{k+1} = a_{31} [\![x_1^{k+1}]\!] + a_{32} [\![x_2^{k+1}]\!] + a_{33} x_3^k + \ldots + a_{3n} x_n^k + b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{k+1} = a_{n1} [\![x_1^{k+1}]\!] + a_{n2} [\![x_2^{k+1}]\!] + a_{n3} [\![x_3^k]\!] + \ldots + a_{nn} x_n + b_n. \end{cases}$$

Вираз $[\![x_1^{k+1}]\!]$ (аналогічно $[\![x_2^{k+1}]\!]$, $[\![x_3^{k+1}]\!]$,…) взято у подвійні квадратні дужки для того, щоб підкреслити, що це значення обчислюється у попередньому співвідношенні (рівнянні).

Порівнюючи ці два алгоритми, стає очевидним, що вони ідентичні по своїй структурі. Їх спільну блок-схему можна представити наступним чином (див. рис 3.1).

У методі верхньої релаксації наступне наближення кореня обчислюється за формулою

$$x_i^{(k+1)} = x_i^{(k)} + \omega \left(\overline{x}_i^{(k+1)} - x_i^{(k)} \right) = (1 - \omega) x_i^{(k)} + \omega \overline{x}_i^{(k+1)},$$

де

$\overline{x}_i^{(k+1)}$ – уточнене значення змінної за методом Зейделя,

ω – параметр релаксації, значення якого визначається з інтервалу

$$1 \le \omega \le 2.$$

При ω=1 метод тотожний методу Зейделя. Швидкість збіжності ітераційного методу верхньої релаксації залежить від значення ω.

Звідси можна зробити висновок, що блок-схеми усіх трьох ітераційних методів є ідентичними. Різниця полягає лише у суті блоку “Процедура обчислення x_{k+1}”.

Рекомендації з вибору оптимального значення ω для методу верхньої релаксації у наявній науковій літературі не зустрічаються. Не виключено, що в кожному конкретному прикладі ці значення різні.

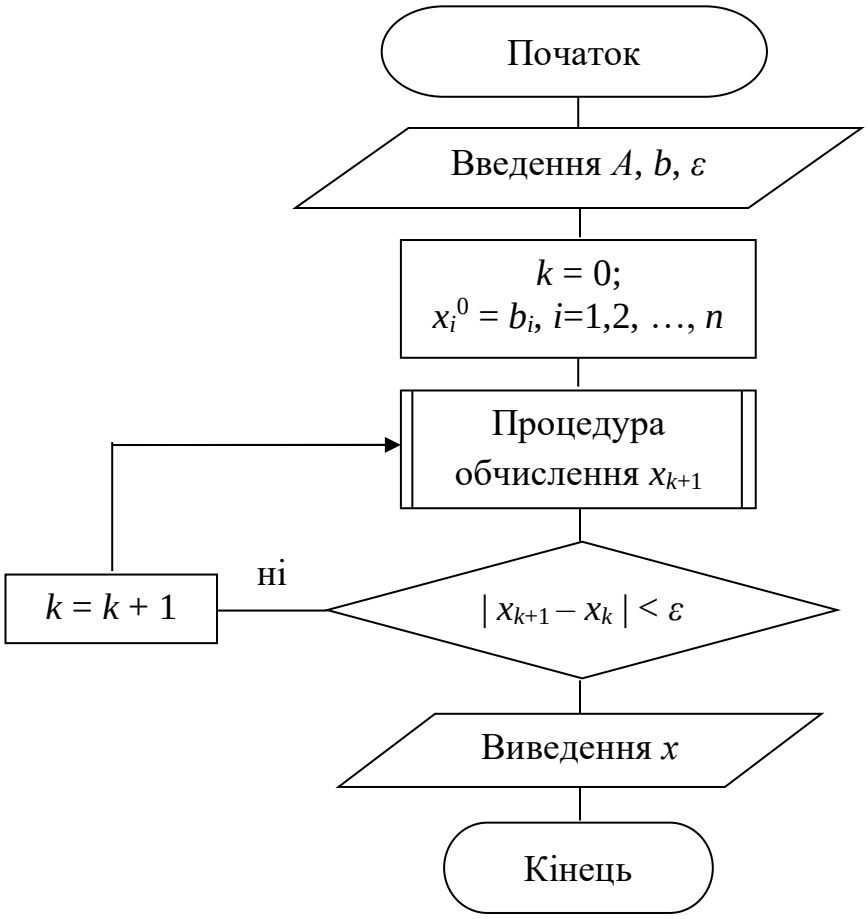

Рисунок 3.1. Схема алгоритму методу ітерацій розв’язування СЛАР

Виходячи з вище сказаного, немає потреби досліджувати усі три методи для порівняння швидкості збіжності з точними методами та з випадком розрідженості даних. Зрозуміло, що метод Зейделя покаже кращу ефективність в порівнянні з методом простої ітерації, оскільки використовує уточнені попередні координати. Щодо методу верхньої релаксації, то тут є додатковий параметр ω, який, як зазначено вище,

впливає на швидкість збіжності і важко наперед судити наскільки сильно впливає.

Враховуючи складності процедури обчислення x_{k+1} для кожного із запропонованих методів, заключаємо, що метод простої ітерації та метод Зейделя мають рівноцінну складність, а метод верхньої релаксації містить більше обчислювальних маніпуляцій, що в кінцевому випадку при багаточисельних розрахунках може призвести до додаткових витрат часу і це негативно вплине на його швидкодію.

Тому для проведення чисельних експериментів ми вибрали метод Зейделя.

Програмний код цього методу, написаний на Visual Basic, розміщено у додатку.

У другому розділі говорилося про те, як перейти від загальної до ітераційної. Це є дійсно складною задачею при довільному формуванні головної матриці. Від ітераційної форми до загальної перехід досить простий. Потрібно вирази з невідомими з правої частини рівнянь перенести у ліві. Отже, для методів Гауса систему рівнянь можна подати у наступному вигляді:

$$\begin{cases} (a_{11}-1)x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = -b_1; \\ a_{21}x_1 + (a_{22}-1)x_2 + \ldots + a_{2n}x_n = -b_2; \\ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + (a_{nn}-1)x_n = -b_n. \end{cases}$$

3.3. Поняття розріджених даних

Коли говорять про масив розріджених даних, то в основному розуміють розріджені дані двовимірного масиву, тобто матриці. Хоча немає єдиного визначення розрідженої матриці, але прийнято вважати, що матриця має розріджену структуру (будову), якщо значна кількість її елементів є нульовими.

Виходячи із рекомендацій зі збереження розріджених масивів, доцільно вважати масив розрідженим, якщо кількість ненульових

елементів у масиві менша $n/2$ для n-вимірних векторів (лінійних масивів), менша $n^2/3$ для квадратним матриць розмірності $n \times n$, менша $n^3/4$ для тривимірних масивів розмірності $n \times n \times n$ і т.д. Річ у тім, що у багатьох варіантах збереження розріджених даних пропонують проводити у вигляді декількох лінійних масивів.

Припустимо, що є k ненульових елементів у розрідженому масиві. Один із лінійних масивів відводять під збереження значень цих ненульових елементів розрідженого масиву. Отже, нам потрібно k змінних для їх збереження і ще k змінних для збереження номерів координат, в яких розміщені ці елементи. Всього потрібно $2k$ позицій для кодування розрідженого лінійного масиву. Якщо $n \leq 2k$, то немає сенсу кодувати такий масив.

Отже, щоб дані вважати розрідженими потрібно, щоб виконувалося співвідношення $n > 2k$, або $k < n/2$, тобто ненульових елементів повинно бути менше половини.

Аналогічно, для матриць, якщо є k ненульових елементів, то для їх кодування потрібно k змінних для збереження значень цих елементів, k змінних для збереження номерів рядків, де вони розміщені, і ще k змінних для збереження номерів стовпців, в яких розміщені ці елементи. Всього $3k$. Аналогічно, матрицю можна вважати розрідженою, якщо $3k < n^2$, або $k < n^2/3$.

Отже, кількість ненульових елементів у квадратній матриці розмірності $n \times n$ повинна бути меншою $n^2/3$.

Аналогічні вирази можна одержати і для масивів більших розмірностей.

3.4. Формати збереження розріджених матриць.

При створенні алгоритмів розв’язування задач з розрідженими матрицями велике значення має вибір способів задання та збереження ненульових елементів. Складність ефективної обробки розріджених

матриць привела до створення великої кількості форматів для компактного зберігання розріджених даних, наприклад, CSR (com- pressed sparse row), CSC (compressed sparse column), COO (coordinate) та інші.

Ці формати (способи кодування) визначаються насамперед структурою (розміщенням ненульових елементів) розрідженої матриці та вимогами алгоритму, який використовується. Оптимальність того чи іншого формату залежить не лише від рівня економного використання машинної пам’яті, але й від зручності оперування цими даними у закодованому вигляді.

Не можна рекомендувати єдиний, близький до оптимального, формат зберігання даних. Практичні дослідження з проведенням аналізу різних форматів збереження розріджених матриць показав, що формат CSR належить до найбільш універсальних і дозволяє легко реалізувати оптимальний алгоритм множення розрідженої матриці на вектор.

Розглянемо коротенько його суть. Передбачається, що розріджена матриця зберігається з використанням трьох масивів.

• aelem містить всі ненульові елементи матриці A, перераховані по рядках;

• jptr містить стільки ж елементів, скільки й aelem, і для кожного з них указує, в якому стовбці перебуває даний елемент;

• iptr містить число елементів, що дорівнює розмірності системи, збільшеній на одиницю. Його i-й елемент указує, з якої позиції в масивах aelem і jptr починається i-й рядок матриці. Відповідно, jptr[i+1]- iptr[i] дорівнює кількості ненульових елементів в i-му рядку. Останній елемент jptr[n+1] дорівнює кількості елементів в масиві aelem, збільшеній на одиницю.

Нехай, наприклад, маємо розріджену матрицю розмірності 7×7:

$$\begin{pmatrix} 9 & 0 & 0 & 3 & 1 & 0 & 1 \\ 0 & 3 & 2 & 1 & 0 & 0 & 2 \\ 0 & 1 & 4 & 2 & 0 & 0 & 0 \\ 2 & 1 & 2 & 9 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 7 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 \\ 2 & 2 & 0 & 0 & 3 & 0 & 8 \end{pmatrix}$$

Її код в форматі CSR матиме вигляд:

- aelem = [9,3,1,1,3,2,1,2,1,4,2,2,1,2,9,1,1,1,7,1,8,2,2,3,8],
- jptr = [1,4,5,7,2,3,4,7,2,3,4,1,2,3,4,5,1,4,5,7,6,1,2,5,7],
- iptr = [1, 5, 9, 12, 17, 21, 22, 26].

Варто зауважити, що конкретні значення ненульових елементів матриці у загальному випадку неможливо передбачити. Тому масив типу aelem присутній у всіх типових форматах.

Існують формати, в яких використовують не три а два масиви. Наприклад, значення у другому масиві формують за формулою $i \bullet (n - 1) + j$, де i – номер рядка, j – номер стовпця, у яких розміщено відповідний елемент масиву aelem. Для наведеного вище прикладу цей масив виглядатиме наступним чином:

[1, 4, 5, 7, 9, 10, 11, 12, 14, 16, 17, 18, 22, 23, 24, 25, 26, 29, 32, 33, 35, 41, 43, 44, 47, 49].

Такий формат не є зручним для проведення операцій над елементами розрідженого масиву, оскільки для визначення їх розміщення потрібно проводити додаткові маніпуляції.

Тому ми вибираємо формат, який складатиметься з трьох масивів:

- aelem містить всі ненульові елементи матриці A (елементи можуть мати хаотичне розміщення);
- nomer_r містить стільки ж елементів, скільки й aelem і для кожного з них указує, в якому рядку перебуває даний елемент;
- nomer_s містить стільки ж елементів, скільки й aelem, і для кожного з них указує, в якому стовбці перебуває даний елемент.

Такий формат на відміну від попередніх не потребує додаткових розрахунків для визначення місцерозташування конкретного ненульового елементу і, що не маловажно, не вимагає впорядкованості масиву aelem за зростанням номерів рядків і стовпців, а при маніпуляціях з елементами матриці такий порядок може порушитися.

Для наведеного вище прикладі відповідні масиви матимуть вигляд:

nomer_r = [1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 6, 7, 7, 7, 7],

nomer_s = [1, 4, 5, 7, 2, 3, 4, 7, 2, 3, 4, 1, 2, 3, 4, 5, 1, 4, 5, 7, 6, 1, 2, 5, 7].

Скалярний добуток векторів

Обчислення нових значень шуканого розв'язку в ітераційних методах зводиться фактично операції до множення матриці на вектор, яка, у свою чергу, розбивається на ряд скалярних добутків векторів. Тому опишемо алгоритми, як ми пропонуємо проводити ці операції у випадку розріджених даних.

Нехай ненульові значення вектора *a* задаються масивом mas1_*a* , а їх розміщення – масивом mas2_*a*. Аналогічні початкові дані стосуються і вектора *b*. Припустимо, що нам невідомі розмірності векторів, а лише кількість ненульових елементів *sa* і *sb* відповідно. Алгоритм обчислення скалярного добутку $c = a \cdot b$ може мати такий вигляд:

1. Скалярному добутку присвоюємо нульове значення (c=0).
2. Беремо по черзі ненульове значення першого вектора.
3. Шукаємо серед ненульових значень другого вектора той, що має таке ж значення в другому масиві, і добутки значень перших масивів додаємо до значення добутку.

Процес продовжується доти, поки не дійдемо до кінця масивів.

3.5. Множення матриці на вектор

Будемо вважати, що матриця A задана трьома масивами, а вектор *b* лише двома (першим та другим). По черзі вибираємо ненульові рядки

матриці. Після цього обчислюємо скалярний добуток цих вектор-рядків на заданий вектор. Процедуру обчислення скалярного добутку описано вище. Оскільки при цьому можуть виникнути нульові значення, то зменшуємо одержаний масив, шляхом вилучення нульових записів.

3.6. Результати експерименту

Ми вибрали точність розв'язування задачі $\varepsilon=0.00000000001=10^{-11}$. Це досить висока, але не максимальна точність. Максимальна, як відомо, становить 10^{-16}.

Було проведено ряд чисельних експериментів розв'язання систем лінійних алгебраїчних рівнянь методом Зейделя без врахування розрідженості матриці.

1. При n=6 алгоритм завершив роботу на 19-му кроці і час виконання завдання становив 28 секунд. В середньому на один крок витрачалось 1,47 секунди.
2. При n=60 алгоритм завершив роботу на 14-му кроці і час виконання завдання становив 30 секунд. В середньому на один крок витрачалось 2,14 секунди. Час збільшився у 1,46 рази в порівнянні з попереднім випадком, тобто із збільшенням розмірності задачі в 10 раз.
3. При n=600 та алгоритм завершив роботу на 13-му кроці і час виконання завдання становив 1 хвилина і 3 секунди. В середньому на один крок витрачалось 4,85 секунди. Уже більше як у два рази в порівнянні з попереднім.
4. При n=6000 та алгоритм завершив роботу на 12-му кроці і час виконання завдання становив 6 хвилина і 33 секунди. В середньому на один крок витрачалось 32,75 секунди, що майже у вісім разів. Більше за попередній випадок.
5. Експерименти показують, що пам'ять комп'ютера дозволяє обробляти матрицю з максимальним розміром задачі

16318×16318. При максимальній розмірності алгоритм завершив роботу на 11-му кроці і час розв’язування становив 16 хвилин і 56 секунд. В середньому на один крок витрачалось 92,36 секунди.

З наведених прикладів стає очевидним, що із збільшенням розмірів задачі час виконання одного кроку алгоритму, а отже, і загального часу на розв’язання задачі, стрімко зростає.

Висновок: проблема мінімізації затрат часу для розв’язання задачі великої (гіпервеликої) розмірності є досить актуальнішою.

Проаналізуємо тепер поведінку алгоритму, якщо враховувати розрідженість матриці.

Продовження розробки програми проводилось так, щоб початкові дані зберігалися і окремо виводилися дані стосовно часу розв’язання задачі обома методами: із звичайним поданням матриці і з закодованим способом для розріджених матриць.

1. При n=6 час виконання завдання в секундах співпадав при обох методах.
2. При n=60 алгоритм завершив роботу за 31 секунду при звичайному поданні матриці і на 4 секунди швидше при врахуванні розрідженості.
3. При n=600 алгоритм завершив роботу за 62 секунди при звичайному поданні матриці і на 8 секунд швидше при врахуванні розрідженості.

ВИСНОВКИ

За проведеним аналізом можна зробити висновки, що про великих розмірах системи ітераційні методи розв’язування систем лінійних алгебраїчних рівнянь ефективніші за точні методи. Як правило похибки округлення при ітераційному методі впливають на остаточні результати значно менше, ніж при методі Гауса, оскільки при його використанні похибки не нагромаджуються. Ітераційні методи стають особливо зручним при розв’язуванні розріджених систем, тобто коли переважна кількість коефіцієнтів системи дорівнює 0.

Разом з тим, ефективному використанню ітераційних методів заважають ряд обставин, зокрема:

1) невідомо, як обирати ітераційні параметри, що необхідні для роботи конкретних методів, наприклад, коефіцієнт релаксації для методу послідовної верхньої релаксації.

2) значною проблемою постає задача перетворення системи лінійних алгебраїчних рівнянь у загальній формі у збіжну ітераційну форму.

Встановлено, що запропонований метод врахування розрідженості для ітераційних методів при великій розмірності та точності обчислення порядку 10^{-11} показує кращу швидкодію до 15 відсотків в порівнянні із звичними ітераційними методами без врахування розрідженості.

Одержані результати можна використати при читанні курсу “Обчислювальні методи”. Також вони будуть корисними в різного роду вибіркових дисциплінах, де звертається увага на використання ефективних чисельних методів.

Цікавими можуть бути продовження досліджень з використанням розгалужених обчислень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Волонтир Л.О, Зелінська О.В., Потапова Н.А., Чіков І.А. Чисельні методи : навчальний посібник. Вінниця : ВНАУ, 2020. 322 с.
2. Березин И.С., Жидков Н.П. Методы вычислений: в 2 Т. М.: 1962. Т. 1. 464 с. 1962. Т.2. 620 с.
3. Гончар Ф.Ш., Моцик Р.В. Порівняльна характеристика точних та наближених чисельних методів розв'язання систем лінійних алгебраїчних рівнянь. Вісник Кам'янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Випуск 15. Кам'янець-Подільський : Кам'янець-Подільський національний університет імені Івана Огієнка, 2022, с. 12-15.
4. Демидович Б. Л., Марон И. А. Основы вычислительной математики. М.: 1970. 664 с.
5. Калиткин Н.Н. Численные методы. М. : Наука, 1978. 512 с.
6. Ляшенко М.Я., Головань М.С. Чисельні методи : підручник. Київ : Либідь, 1996. 288 с.
7. Мястковська М.О., Щирба В.С., Щирба О.В. Чисельні методи – Кам'янець-Подільський : видавець ПП Зволейко Д.Г., 2013. 84 с.
8. Мястковська М. О., Фуртель О. В., Щирба В. С. Лабораторний практикум з курсу обчислювальних методів: навчально-методичний посібник. 2-е вид, доп. і перероб. [Електронний ресурс]. Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2023. 167 с. Електронна версія посібника доступна за покликаннями:
   URL: http://elar.kpnu.edu.ua/xmlui/handle/123456789/7476
9. Стіренко С. Г. Зберігання розріджених матриць великих розмірностей в системі з локальною пам'яттю. Вісник НТУУ «КПІ»

Інформатика, управління та обчислювальна техніка: зб. наук. праць. Київ: Век+, 2020. № 52. С. 111–117.

10. Хейгеман Л., Янг Д. Прикладные итерационные методы. М.: Мир, 1986. 446 с.
11. Джерела інформації

    1. http://eprints.zu.edu.ua/18543/1/metody_obchyslen.pdf

    2. http://www.unicyb.kiev.ua/Library/OM/ZAD1/index.html

Додаток. Лістинг програми алгоритму методу Зейделя розв’язування СЛАР

```
Imports System.Reflection.Emit
Imports System.Timers
Imports System.Windows.Forms.VisualStyles.VisualStyleElement.TaskbarClock

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        End
    End Sub

    Private Sub ListBox1_SelectedIndexChanged(sender As System.Object, e As System.EventArgs) Handles
ListBox1.SelectedIndexChanged

        Dim i, j As Integer
        'DataGridView1.Visible = True

        'DataGridView1.Columns.Clear()
        Select Case ListBox1.SelectedIndex
            Case 0 ' 'Портрет матриці А
                'Підготомка таблиці
                For j = 0 To n + 1
                    DataGridView1.Columns.Add("", "")
                Next
                For i = 0 To n
                    DataGridView1.Rows.Add()
                Next
                ' """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
                For i = 1 To n  'заповнення індексів таблиці для рядків "i"
                    DataGridView1(0, i).Value = "i=" & i
                Next
                For j = 1 To n  'заповнення індексів таблиці для стовпців "J"
                    DataGridView1(j, 0).Value = "j=" & j
                Next
                DataGridView1(n + 1, 0).Value = "вектор b"
                ' """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
                For j = 1 To n
                    For i = 1 To n
                        DataGridView1(i, j).Value = a(j, i)
                    Next
                    DataGridView1(n + 1, j).Value = b(j)
                Next
            Case 1 ' Кодове подання матриці А
                ' DataGridView2.Visible = True
                ' визначення кількості ненульових елементів і формування кодового подання

                'Підготомка таблиці
                For j = 0 To s + 1

                    DataGridView2.Columns.Add("", "")
                Next
                For i = 0 To 4
                    DataGridView2.Rows.Add()
                Next
                ' """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

                For j = 1 To s

                    DataGridView2(j, 0).Value = "j=" & j
                    DataGridView2(j, 1).Value = a1(j)
                    DataGridView2(j, 2).Value = a2(j)
```

```
            DataGridView2(j, 3).Value = a3(j)
            DataGridView2(0, 1).Value = "a1"
            DataGridView2(0, 2).Value = "a2"
            DataGridView2(0, 3).Value = "a3"
            'DataGridView2(j + 1, 5).Value = ss(j)
            'DataGridView2(j + 1, 6).Value = s
            'DataGridView2(j + 1, 7).Value = sumar(j)
        Next
    Case 2 ' вектор вільних членів
        For i = 0 To n
            DataGridView1.Columns.Add("", i)
        Next
        For i = 0 To 1
            DataGridView1.Rows.Add()
        Next

        For i = 1 To n   'заповнення індексів таблиці для стовпців "i"
            DataGridView1(i, 0).Value = "i=" & i
        Next
        For j = 1 To n 'заповнення координатими вектора в
            DataGridView1(j, 1).Value = b(j)
        Next
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Case 3 ' Розріджений варіант

        Dim k, l As Integer
        For j = 1 To n
            x(j) = b(j)
        Next
        Label4.Text = Label4.Text + TimeOfDay
        'For j = 0 To 600 ' n
        '    DataGridView2.Columns.Add("", "j")
        'Next
        'For i = 0 To 44 '* n
        '    DataGridView2.Rows.Add()
        'Next
        'DataGridView2(0, 0).Value = "Крок"
        For j = 1 To 6 * n
            delta = 0
            l = 0
            For i = 1 To n
                sumar(i) = 0

                For k = 1 To ss(i)
                    sumar(i) = sumar(i) + a1(k + l) * x(a3(k + l))
                Next
                l = l + k - 1
                delta = delta + Math.Abs(x(i) - sumar(i) - b(i))
                x(i) = sumar(i) + b(i)
            Next
            If delta < 0.00000000000001 Then
                Exit For
            End If

        Next
        'For i = 1 To n
        '    l = Int((i - 1) / 500)
        '    DataGridView2(i - l * 500, l + 2).Value = x(i)
        'Next
        'DataGridView2(0, 1).Value = j
        Label5.Text = Label5.Text + TimeOfDay
        Label5.ForeColor = SystemColors.ActiveBorder
        Label5.ForeColor = SystemColors.HotTrack
    Case 4 ' виведення часу розв'язку задачі і кількості кроків з повним портретом

        Dim k, l As Integer
```

```
        For j = 1 To n
            x(j) = b(j)
        Next
        Label1.Text = Label1.Text + TimeOfDay
        'For j = 0 To 600 ' n
        '    DataGridView1.Columns.Add("", "j")
        'Next
        'For i = 0 To 44 '* n
        '    DataGridView1.Rows.Add()
        'Next
        'DataGridView1(0, 0).Value = "Крок"
        For j = 1 To 10 * n
            delta = 0
            For i = 1 To n
                sumar(i) = 0
                For k = 1 To n
                    sumar(i) = sumar(i) + a(i, k) * x(k)
                Next
                delta = delta + Math.Abs(x(i) - sumar(i) - b(i))
                x(i) = sumar(i) + b(i)
            Next
            If delta < 0.00000000000001 Then
                Exit For
            End If
        Next
        'For i = 1 To n
        '    l = Int((i - 1) / 500)
        '    DataGridView1(i - l * 500, l + 2).Value = x(i)
        'Next
        'DataGridView1(0, 1).Value = j
        Label3.Text = Label3.Text + TimeOfDay
        Label3.ForeColor = SystemColors.ActiveBorder
        Label3.ForeColor = SystemColors.HotTrack
    End Select

End Sub
Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button2.Click ' стартові дані задачі

    Dim i, j, K As Integer
    n = Val(TextBox1.Text)

    Randomize()
    For i = 1 To n
        a(i, i) = 2 * Rnd() / n
    Next
    For K = 1 To n
        i = Int(n * Rnd() + 1)
        j = Int(n * Rnd() + 1)
        a(i, j) = 2 * Rnd() / n
    Next
    For i = 1 To n
        b(i) = 1
        For j = 1 To n
            b(i) = b(i) - a(i, j)
        Next
        x(i) = b(i)
    Next
    s = 0
    For j = 1 To n
        ss(j) = 0
        sumar(j) = 0
    Next
    For j = 1 To n
        For i = 1 To n
            If a(j, i) <> 0 Then
                s = s + 1 ' s, кількість ненульвових елементів
```

```
                ss(j) = ss(j) + 1 '    ss(100), кількість ненульвових елементів у відповідному рядку
                sumar(j) = sumar(j) + a(j, i) '    рядкова сума ненульових елементів для перевірки умов збіжності
                a1(s) = a(j, i)  'значення s-того по порядку ненульового елементу
                a2(s) = j '     a2(100), рядкові індекси ненульових елементів
                a3(s) = i '    a3(100) стовпцеві індекси ненульових елементів

            End If

        Next
    Next
    'Label1.Text = Hour(TimeOfDay) 'години системного часу
    'Label1.Text = Label1.Text + "годин "
    "Label1.Text = Label1.Text + Minute(TimeOfDay) 'хвилини
    " Label1.Text = Hour(TimeOfDay) + Minute(TimeString)
    'Label1.Text = Minute(TimeString) '+ CStr("хвилин ") + CStr(Second(TimeOfDay)) + CStr("секунд ")
'секунди
    ' Label1.Text = Hour(TimeOfDay) + Minute(TimeOfDay) + Second(TimeOfDay) 'секунди
    'Label1.Text = Label1.Text + "хвилин "
    'End If
  End Sub


End Class
```