1 ПАРАЛЕЛЬНІ АЛГОРИТМИ РОЗВ'ЯЗУВАННЯ ЗАПОВНЕНИХ СИСТЕМ ЛІНІЙНИХ РІВНЯНЬ

1.1 Мета роботи – навчитися створювати і аналізувати паралельні алгоритми розв'язування заповнених систем лінійних рівнянь та оцінювати показники їх прискорення у порівнянні з послідовними алгоритмами.

1.2 *Методичні вказівки з організації самостійної роботи студентів*

Теоретичні відомості.

Система лінійних алгебраїчних рівнянь має вигляд

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 ; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 ; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m , \end{cases} \tag{1.1}$$

або в матричній формі:

$$AX = B , \tag{1.2}$$

де

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}; \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}; \qquad B = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{pmatrix}.$$

Розширена матриця системи має вигляд

$$\overline{A} = (A \mid B) = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & \ldots & a_{2n} & b_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mn} & b_m \end{array}\right). \tag{1.3}$$

Розв'язком системи називається матриця-стовпець X, яка обертає матричне рівняння $AX = B$ у тотожність.

Система називається *сумісною*, якщо має хоча б один розв'язок, у протилежному випадку, вона називається *несумісною*.

Система лінійних алгебраїчних рівнянь називається заповненою, якщо у її матриці коефіцієнтів A немає нульових значень.

Існують точні та наближені методи розв'язування систем лінійних алгебраїчних рівнянь.

До точних методів належать методи, що дають точний результат у припущенні ідеальної арифметики. Точні методи можна застосовувати й тоді, коли коефіцієнти й вільні члени рівняння задані в аналітичній, символьній формі.

1. *Метод послідовного виключення невідомих* є найпростішим, хоча важким для практичних застосувань, методом розв'язування системи лінійних алгебраїчних рівнянь.

Із першого рівняння змінна виражається через інші змінні, й підставляється в усі інші рівняння. Це можна зробити, якщо коефіцієнт відмінний від нуля. У випадку, якщо він нульовий, можна вибрати інше рівняння, оскільки перестановка рівнянь у системі дає еквівалентну систему. В результаті утворюється нова система рівнянь, в якій рівнянь на одне менше.

З цією системою рівнянь можна поступити так само, отримуючи ще меншу систему рівнянь. Продовжуючи так, отримують одне лінійне рівняння, з якого можна визначити одну із змінних, а інші, виключені, виразити через неї.

2. *Матричний метод* полягає у розв'язанні матричного рівняння $X = A^{-1}B$.

Реалізація методу полягає в знаходженні оберненої матриці і множенні її на стовпець вільних членів. Використовується для невироджених ($\det A \neq 0$) квадратних систем.

Оберненою до квадратної матриці A називається матриця A^{-1} така, що

$$A\,A^{-1} = A^{-1}A = E\,.$$

Для того щоб квадратна матриця A мала обернену, необхідно та достатньо, щоб матриця коефіцієнтів A була невиродженою.

Приєднаною до квадратної матриці A називається матриця

$$\widetilde{A} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix}, \qquad (1.4)$$

де A_{ij} – алгебраїчні доповнення елементів a_{ij} матриці A.

Згідно з цим методом обернена матриця знаходиться за формулою

$$A^{-1} = \frac{1}{\det A}\widetilde{A} = \frac{1}{\det A}\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}^{T} = \frac{1}{\det A}\begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix}. \qquad (1.5)$$

3. *Формули Крамера:* $x_i = \dfrac{\Delta_i}{\Delta}$, $i = \overline{1, n}$, де $\Delta = \det A \neq 0$ – визначник системи; визначник Δ_i одержується з визначника Δ шляхом заміни i–го стовпця стовпцем вільних членів. Також використовуються для невироджених квадратних систем.

4. *Метод Гауса* ґрунтується на тому, що елементарним перетворенням рядків розширеної матриці системи відповідає перетворення цієї системи в еквівалентну.

Прямий хід. За допомогою елементарних перетворень рядків розширеної матриці, а також зміни місцями стовпців (що відповідає перепозначенню змінних) розширена матриця $\overline{A}$ зводиться до східчастої (або трапецієвидної) форми з трикутною матрицею ненульових значень.

Зворотній хід. Обчислюється значення невідомої змінної для рядку з найбільшою кількістю нульових елементів у трикутній матриці коефіцієнтів. Її значення підставляється до сусіднього рядка та обчислюється значення другої змінної. Цей процес продовжується до обчислення значення останньої невідомої змінної.

Кожен з вказаних методів має декілька груп дій, які не залежать одна від одної і тому можуть бути виконані одночасно (паралельно). Це, зокрема, дії з обчислення визначників різного порядку (у тому числі алгебраїчних доповнень), множення матриць та елементарних перетворень.

Припустимо, що матриця A є лівою трикутною матрицею з одиничною діагоналлю. Тоді маємо

$$x_1 = b_1, \quad x_i = b_i - \sum_{j=1}^{i-1} a_{ij} x_j, \; (2 \le i \le n). \tag{1.6}$$

Цей запис також не визначає алгоритм однозначно, бо не вказано порядок обчислення сум. Розглянемо наступне уточнення процесу (1.6):

$$\begin{gathered} x_i^{(0)} = b_i, \\ x_i^{(j)} = x_i^{(j-1)} - a_{ij} x_j^{(j-1)}, \quad i = 1,2,\ldots,n, \; j = 1,2,\ldots,i-1, \\ x_i = x_i^{(i-1)}. \end{gathered} \tag{1.7}$$

Основна операція алгоритму має вигляд $a - b\,c$. Вона виконується для усіх допустимих значень індексів i та j. Для побудови графа алгоритму в декартовій

системі координат з осями i та j побудуємо прямокутну сітку і розмістимо у вузлах при $2 \le i \le n,\ 1 \le j \le i-1$ вершини графа, які відповідають операціям $a - b\,c$. Також зобразимо на графі вершини, які відповідають вводу вхідних даних a_{ij} та b_j. Цей граф для випадку $n = 5$ зображено на рис. 1.1. Верхня кутова вершина відповідає знаходженню у точці (1, 0).

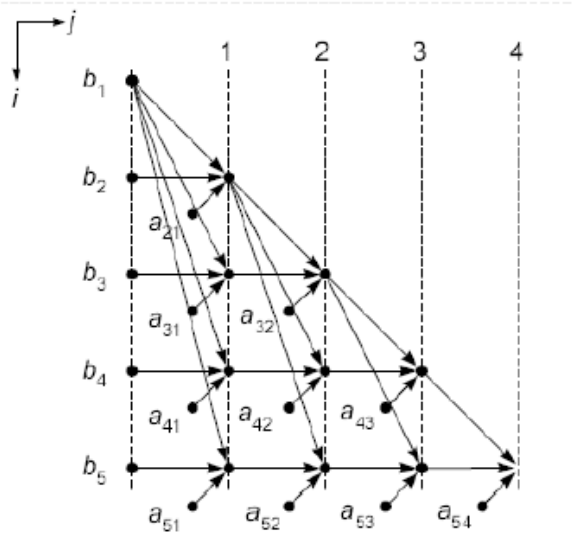

Рис. 1.1 - Граф для алгоритму зворотної підстановки (1.7) для трикутної системи

На цьому рисунку зображена одна із максимальних паралельних форм. Її яруси помічені пунктиром. Ця паралельна форма стане канонічною, якщо вершини, відповідні за ввід елементів a_{ij}, розташувати у першому ярусі. Загальна кількість ярусів (без урахування вводу) дорівнює n - 1.

Вибір зростаючого за j напрямку додавання у (1.6), який призвів до алгоритму (1.7), був зроблений, взагалі кажучи, випадково.

Аналогічно можна побудувати алгоритм зворотної підстановки з використанням додавання за спаданням індексу j:

$$x_i^{(i)} = b_i,$$

$$x_i^{(j)} = x_i^{(j+1)} - a_{ij}x_j^{(j)},\ \ i = 1,2,\ldots,n,\ j = 1,2,\ldots,i-1, \qquad (1.8)$$

$$x_i = x_i^{(i-1)}.$$

Відповідний граф для випадку $n = 5$ наведено на рис. 1.2. Тепер верхня кутова вершина розташована у точці (1, 1).

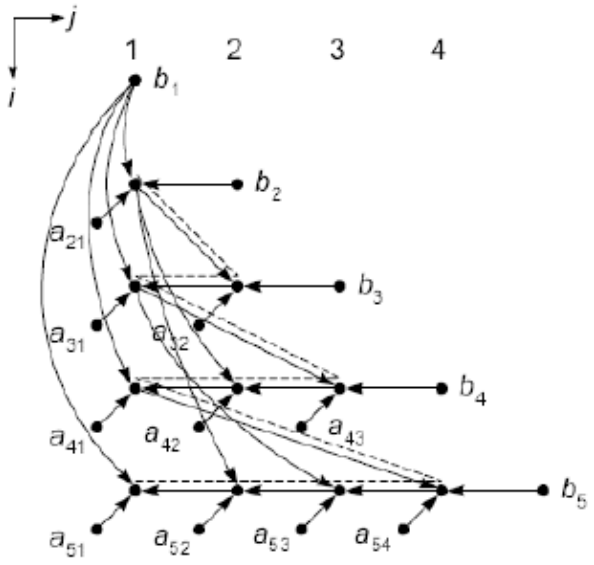

Рис. 1.2 - Граф для алгоритму зворотної підстановки (10) для трикутної системи

Пробуючи розташувати вершини, які відповідають операціям $a - b\ c$, за ярусами хоча б однієї паралельної форми, приходимо до висновку, що тепер у кожному ярусі завжди може знаходитися тільки одна вершина. Цей факт пояснюється тим, що усі вершини графа на рис. 6.3 лежать на одному шляху, який позначений на рисунку пунктиром. Тому загальна кількість ярусів алгоритму (10), які містять операції вигляду $a - b\ c$, завжди рівна $(n^2 - n + 2)/2$, що набагато більше за число $n - 1$ — кількість ярусів для відповідних операцій у алгоритмі (1.7).

Обидва алгоритми (1.7) та (1.8) призначені для розв'язування тієї самої задачі і розроблені на основі формул (1.6). Обидва алгоритми абсолютно однакові з точки зору їх реалізації на багатопроцесорній системі, оскільки потребують виконання однакової кількості операцій множення та віднімання і однакового об'єму пам'яті і є еквівалентними з точки зору помилок заокруглення.

Тим не менш, паралельні графи алгоритмів принципово різні. Якщо ці алгоритми реалізувати на паралельній системі з n універсальними процесорами, то алгоритм (1.7) можна реалізувати за час, пропорційний n, а алгоритм (1.8) — лише за час пропорційний n^2. У першому випадку завантаженість процесорів близька до 0,5, а у другому — до 0.

Таким чином, алгоритми, цілком однакові при послідовній реалізації, можуть виявитися принципово відмінними при реалізацій на паралельній обчислювальній системі.

Показники прискорення за рахунок паралельного виконання частини дій, передбачених у певному алгоритмі, можна обчислити згідно законів Амдала або Густавсона.

Прискорення S (як відношення до часу виконання послідовного алгоритму) за законом Амдала задається рівнянням:

$$S = 1/(p + (1 - p)/n),$$

де: p— частина яку можна виконувати послідовно; 1 - p частина, яка виконується паралельно; — кількість процесорів.

Прискорення за законом Густавсона обчислюється за формулою:

$$S_p = g + (1 - g)p = p + (1 - p)g,$$

де g — частка послідовних обчислень в програмі, p — кількість процесорів.

Дану оцінку прискорення називають **прискоренням масштабування** (*scaled speedup*), через те, що вона, наскільки ефективно можуть бути організовані паралельні обчислення при збільшенні складності обчислювальних задач.

1.3 *Варіанти індивідуальних завдань*

Система лінійних алгебраїчних рівнянь (СЛАР) задана матрицею коефіцієнтів *A* та вектором вільних членів *B*. Побудувати графи послідовного та паралельного алгоритмів розв’язання заданої СЛАР одним з наступних методів:

а) матричним методом з використанням приєднаної матриці,

б) матричним методом з використанням елементарних перетворень рядків,

в) методом Крамера,

г) методом Гауса,

д) методом LU-розкладання матриці коефіцієнтів *A*.

Реалізувати алгоритми програмно. Обчислити значення показників прискорення та ефективності розпаралелювання.

1.4 Контрольні запитання і завдання.

1. Поясніть, у чому полягає суть алгоритму викреслювання стовпців, як виконується оцінка коефіцієнта прискорення цього алгоритму у разі блокового розподілу даних. Спробуйте оцінити коефіцієнт прискорення у разі блочно-циклічного розподілу даних в припущенні відсутності часу на обмін даними між процесами.

2. В чому полягає ефект Гайдна, що визначає коефіцієнт Гайдна? Чому дорівнює коефіцієнт Гайдна при блоковому розподілі даних? Оцініть коефіцієнт Гайдна для випадку блочно-циклічного розподілу даних.

3. Опишіть модифікацію алгоритму викреслювання стовпців, яка дозволяє практично уникнути впливу ефекту Гайдна на прискорення паралельних обчислень.

4. Опишіть алгоритм логарифмічного підсумовування для розв’язання лінійної рекурентної задачі підсумовування n чисел. Як визначається коефіцієнт прискорення для цього алгоритму?

5. На чому оснований алгоритм рекурентного добутку? Що таке оцінка складності алгоритму і як через неї виражається коефіцієнт прискорення?

6. Сформулюйте та поясніть закон Амдала про залежність прискорення виконання програми від кількості процесорів при різних рівнях розпаралелювання програмного коду.

7. Сформулюйте та поясніть закон Густавсона-Барсіса про оцінку прискорення виконання програми в залежності від кількості процесорів при різних рівнях розпаралелювання програмного коду.