Лекція 7_дод. Деякі базові паралельні алгоритми обчислень (ІІ ч.).

План лекції.

1. Розпаралелювання методу виключення Гаусса для розв'язання СЛАР.
2. Розпаралелювання методу Ейлера для розв'язання задачі Коші.
3. Паралельні алгоритми розв'язання одновимірної задачі цифрової фільтрації:
   - синхронна схема обчислень;
   - асинхронна схема обчислень;
   - використання методу гіперплощин для розпаралелювання циклів.

1. Нехай A– задана невироджена матриця (визначник відмінний від нуля) розмірності $n \times n$, а $\vec{x}, \vec{b}$ – відповідно невідомий та відомий вектори розмірності n. Розглянемо СЛАР

$$A\vec{x} = \vec{b}, \qquad (1)$$

де $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$.

Вважаємо, що матриця A є щільною, тобто повністю заповненою.

<u>Прямий хід</u> найпростішого методу виключення **Гаусса** полягає в наступному:

1) виключаємо змінну x_1; нехай $a_{i1} \neq 0,\ i = \overline{1, n}$; тоді кожен i-й рядок матриці A та кожну i-ту компоненту вектора $\vec{b}$ ділимо на a_{i1} і потім віднімаємо 1-ше рівняння системи від решти рівнянь, тобто виконуємо обчислення:

$$\vec{a}_i^1 = \frac{\vec{a}_i}{a_{i1}} - \frac{\vec{a}_1}{a_{11}};\ b_i^1 = \frac{b_i}{a_{i1}} - \frac{b_1}{a_{11}};\ i = \overline{2, n}, \qquad (2)$$

де $\vec{a}_i,\ \vec{a}_i^1$ – i-ті рядки відповідно вихідної матриці A та обчисленої матриці A^1; $b_i,\ b_i^1$ – i-ті компоненти відповідно векторів $\vec{b},\ \vec{b}^1$. При цьому

$$\vec{a}_1^1 = \frac{\vec{a}_1}{a_{11}};\ b_1^1 = \frac{b_1}{a_{11}}.$$

До того ж

$$A^1 = \begin{pmatrix} 1 & a_{12}^1 & \dots & a_{1n}^1 \\ 0 & a_{22}^1 & \dots & a_{2n}^1 \\ \dots & \dots & \dots & \dots \\ 0 & a_{n2}^1 & \dots & a_{nn}^1 \end{pmatrix};\ \vec{b}^1 = \begin{pmatrix} b_1^1 \\ b_2^1 \\ \dots \\ b_n^1 \end{pmatrix};$$

2) далі виключимо x_2, для чого в матриці A^1 фіксуємо 1-й рядок і 1-й стовпець та в векторі $\vec{b}^1$ фіксуємо 1-у компоненту, а для решти компонент матриці та вектора повторюємо попередній крок. Унаслідок цього одержуємо матрицю A^2 та вектор $\vec{b}^2$, де

$$A^2 = \begin{pmatrix} 1 & a_{12}^1 & a_{13}^1 & \dots & a_{1n}^1 \\ 0 & 1 & a_{23}^2 & \dots & a_{2n}^2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a_{n3}^2 & \dots & a_{nn}^2 \end{pmatrix}; \quad \vec{b}^2 = \begin{pmatrix} b_1^1 \\ b_2^2 \\ \dots \\ b_n^2 \end{pmatrix}.$$

- -

n) після виконання *n* таких кроків одержимо матрицю A^n і вектор $\vec{b}^n$ для системи, до того ж

$$A^n = \begin{pmatrix} 1 & a_{12}^1 & a_{13}^1 & \dots & a_{1n}^1 \\ 0 & 1 & a_{23}^2 & \dots & a_{2n}^2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}; \quad \vec{b}^n = \begin{pmatrix} b_1^1 \\ b_2^2 \\ \dots \\ b_n^n \end{pmatrix}.$$

Зворотній хід методу виключення **Гаусса**:

1) оскільки СЛАР унаслідок прямого ходу набула вигляду

$$\begin{pmatrix} 1 & a_{12}^1 & a_{13}^1 & \dots & a_{1n}^1 \\ 0 & 1 & a_{23}^2 & \dots & a_{2n}^2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^1 \\ b_2^2 \\ \dots \\ b_n^n \end{pmatrix}, \qquad (3)$$

то з неї легко одержати, що $x_n = b_n^n$.

2) на підставі $(n-1)$-го рівняння системи (3) одержуємо, що

$$x_{n-1} = b_{n-1}^{n-1} - a_{n-1\,n}^{n-1} \cdot x_n \qquad (4)$$

- -

n) на підставі 1-го рівняння системи (3) одержуємо значення для x_1.

В обчислювальній практиці цей **найпростіший метод** виключення Гаусса використовується рідко із-за його **нестійкості** до обчислювальних похибок. Зазвичай використовується метод виключення Гаусса з **вибором головного елемента**. Розглянута схема при цьому ускладнюється необхідністю відшукання на кожному кроці максимального за модулем елемента у відповідному стовпці матриці.

Будемо вважати, що кількість процесорів у MIMD-системі з розподіленою пам'яттю *N*=*n*, а дані розподілені по процесорах так, як показано на рис. 1.

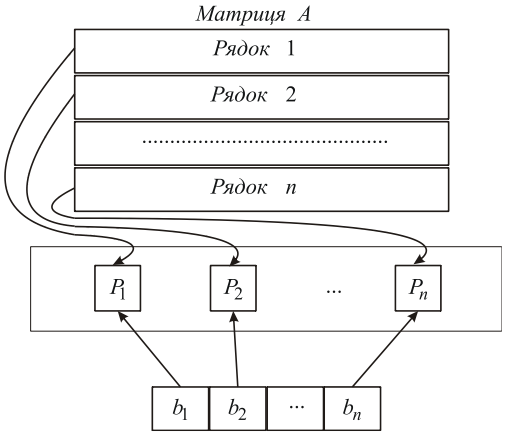

Рис. 1

Схема найпростішого **паралельного алгоритму** методу виключення Гаусса має наступний вигляд.

Прямий хід:

1) розсилаємо 1-й рядок матриці A і 1-у компоненту вектора $\vec{b}$ процесорам $P_2, P_3, \ldots, P_n$;
2) за формулами (2) паралельно на кожному із процесорів P_i обчислюємо компоненти рядка $\vec{a}_i^1$ матриці A^1 та компоненту b_i^1 вектора $\vec{b}^1$ для $i = \overline{2, n}$; а на процесорі P_1 обчислюємо $\vec{a}_1^1$ та b_1^1;
3) розсилаємо 2-й рядок матриці A^1 та 2-у компоненту вектора $\vec{b}^1$ процесорам $P_3, \ldots, P_n$;
4) паралельно на кожному із процесорів P_i обчислюємо компоненти рядка $\vec{a}_i^2$ матриці A^2 та компоненту b_i^2 вектора $\vec{b}^2$ для $i = \overline{3, n}$; а на процесорі P_2 обчислюємо $\vec{a}_2^2$ та b_2^2;
5) і т. д.

Зворотній хід:

1) на процесорі P_n визначаємо величину x_n;
2) пересилаємо цю величину процесорам $P_1, P_2, \ldots, P_{n-1}$;
3) за формулою (4) на процесорі P_{n-1} обчислюємо x_{n-1};
4) пересилаємо цю величину процесорам $P_1, P_2, \ldots, P_{n-2}$;
5) і т. д.

Слід відзначити, що після завершення роботи розглянутого алгоритму **всі компоненти вектора** $\vec{x}$ виявляться на процесорі P_1.

Легко бачити, що розглянута схема породжує проблему сильної незбалансованості завантаження процесорів обчислювальної системи. Дійсно, після завершення двох кроків прямого ходу процесор P_1 простоює до останнього кроку зворотнього ходу. Аналогічно, після завершення чотирьох кроків прямого ходу про-

цесор P_2 простоює до передостаннього кроку зворотнього ходу, а потім знову простоює до завершення обчислень і т.д.

Розглянемо більш реалістичну ситуацію, коли $N < n$, тобто кількість рівнянь системи (1) є більшою за кількість процесорів. Для простоти запису покладемо $p = n / N$ (n є кратним до N).

Пошарова схема зберігання матриці. У цьому разі перші p рядків матриці A можна розташувати на процесорі P_1, другі p рядків – на процесорі P_2 і т. д. У даному випадку перший процесор буде простоювати після $2p$ кроків, другий процесор – після $4p$ кроків і т.д. Однак тут можна організувати паралельні обчислення та пересилання даних так, що завантаження процесорів стає більш збалансованим.

Циклічна пошарова схема зберігання матриці. За такої схеми рядки матриці A розміщуються на процесорах так:

- процесор P_1 – рядки $1, N+1, 2N+1, \ldots, (p-1)N+1$;
- процесор P_2 – рядки $2, N+2, 2N+2, \ldots, (p-1)N+2$;
- ……..
- процесор P_N – рядки $N, 2N, 3N, \ldots, pN$.

Це дозволяє здійснювати розсилку рядків (кроки 1), 3) і т. д. прямого ходу) усім процесорним елементам, а також залучати всі процесори для обчислення елементів матриць $A^1, A^2, \ldots, A^n$.

Така схема зберігання практично знімає проблему незбалансованості завантаження процесорів (**<u>за бажання можна перевірити це самостійно</u>**).

2. Розглянемо **задачу Коші** для системи звичайних диференціальних рівнянь:

$$\begin{cases} x_1' = f_1(t, x_1, x_2, \ldots, x_n), \;\; x_1(t_0) = x_1^0, \\ x_2' = f_2(t, x_1, x_2, \ldots, x_n), \;\; x_2(t_0) = x_2^0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n' = f_n(t, x_1, x_2, \ldots, x_n), \;\; x_n(t_0) = x_n^0, \end{cases}$$

де $t \in [t_0, t_T]$, або у векторній формі:

$$\vec{x}' = \vec{f}(t, \vec{x}), \;\; \vec{x}(t_0) = \vec{x}^0.$$

Розглянемо ситуацію, коли багатопроцесорна обчислювальна система містить значну кількість процесорів N і порядок n системи рівнянь є кратним до кількості процесорів, до того ж $p = n / N$.

Покриваємо інтервал $[t_0, t_T]$ рівномірною сіткою з кроком h. Для того, щоб не ускладнювати схему паралельного алгоритму деталями, розглянемо **явний метод інтегрування Ейлера** (має перший порядок точності):

$$\vec{x}^i = \vec{x}^{i-1} + h\vec{f}(t_{i-1}, \vec{x}^{i-1}), \;\; i = 1, 2, \ldots, M.$$

Схема **паралельного методу** Ейлера:

1) розподіляємо вихідні дані задачі по процесорах так, як показано на рис. 2;

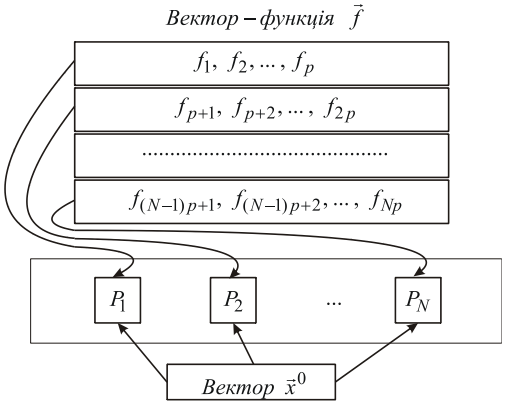

Рис. 2

2) паралельно на всіх процесорах обчислюємо значення відповідних функцій f_i та компонент вектора $\vec{x}^1$:
   - на процесорі P_1 – значення $f_1(t_0, \vec{x}^0), f_2(t_0, \vec{x}^0), \ldots, f_p(t_0, \vec{x}^0)$ і компоненти $x_1^1, x_2^1, \ldots, x_p^1$;
   - на процесорі P_2 – значення $f_{p+1}(t_0, \vec{x}^0), f_{p+2}(t_0, \vec{x}^0), \ldots, f_{2p}(t_0, \vec{x}^0)$ і компоненти $x_{p+1}^1, x_{p+2}^1, \ldots, x_{2p}^1$;
   - …….
   - на процесорі P_N – значення $f_{(N-1)p+1}(t_0, \vec{x}^0), f_{(N-1)p+2}(t_0, \vec{x}^0), \ldots, f_{Np}(t_0, \vec{x}^0)$ і компоненти $x_{(N-1)p+1}^1, x_{(N-1)p+2}^1, \ldots, x_{Np}^1$;
3) кожен із процесорів системи посилає кожному із решти процесорів обчислені компоненти вектора $\vec{x}^1$;
4) аналогічно до кроків 2), 3) проводиться обчислення компонент вектора $\vec{x}^2$;
5) і т.д.

Якщо **обчислювальні складності** компонент вектор-функції $\vec{f}(t, \vec{x})$ є **різними**, то розглянута схема паралельного алгоритму може призвести до виникнення проблеми незбалансованого завантаження процесорів. Для подолання незбалансованості необхідно компоненти вектор-функції об'єднати в **блоки** так, щоб сумарна обчислювальна складність кожного із блоків була приблизно одинаковою.

Розглянута схема розпаралелювання призводить до значного завантаження комунікаційної мережі обчислювальної системи. Дійсно, на кожному кроці інтегрування кожен процесор повинен передати кожному із решти процесорів всі обчислені ним компоненти вектора $\vec{x}$. **Комунікаційне завантаження** процесорів системи можна зменшити, якщо під час об'єднання компонент вектор-функції $\vec{f}(t, \vec{x})$ в блоки, крім **обчислювальної складності** цих компонент, врахувати їх інформаційні залежності. Логічно об'єднати в один блок компоненти вектор-функції, які залежать від значень, обчислюваних на цьому ж процесорі, що і блок.

У зв’язку з невисокою точністю метод Ейлера використовується рідко. Однак, розглянута схема розпаралелювання з невеликими змінами переноситься як на **явні методи** інтегрування (більш високого порядку) ЗДР, наприклад, на методи Рунге-Кутта, так і на **неявні методи**.

3. Розглянемо одновимірну задачу цифрової фільтрації (ЗЦФ), яка полягає у виконанні k переобчислень масиву значень n змінних за формулою:

$$x_i = \sum_{s=-m}^{m} x_{i+s} \cdot f_s \ , \ i = \overline{1,n}. \qquad (5)$$

Тут переобчислення згладжування здійснюються через рухоме вікно розміром $2m+1$. При цьому значення $x_{1-m}, x_{2-m}, \ldots, x_0$; $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ та вагові коефіцієнти $f_{-m}, f_{-m+1}, \ldots, f_m$ – задані константи. Вагові коефіцієнти вибираються такими, щоб їх сума дорівнювала 1.

У більшості випадків ЗЦФ доводиться розв'язувати в режимі реального часу (попереднє оброблення сигналів, зображень, великих масивів експериментальних даних), тому для високошвидкісної фільтрації необхідно використовувати паралельні алгоритми, орієнтовані на реалізацію на високопродуктивних обчислювальних засобах.

Стандартний спосіб розв’язання сформульованої ЗЦФ реалізується таким ***<u>послідовним алгоритмом</u>***:

$FOR\ \ j = 1, k\ \ DO$
{ $FOR\ i = 1, n\ \ DO$
{ $p = 0$
$FOR\ s = -m, m\ \ DO$
{ $p = p + x_{i+s} * f_s$ }
$x_i = p$ } }.

З цього алгоритму випливає, що для послідовного переобчислення значень змінної x_i на j-му кроці беруться значення $x_{i-m}, x_{i-m+1}, \ldots, x_{i-1}$, які є вже переобчисленими на цьому ж кроці.

Для $m = 2, n = 7, k = 3$ інформаційні зв’язки між ітераціями (i, j) наведеного алгоритму схематично зображені на рис. 3. Для наглядності ці зв’язки подані для окремих ітерацій.

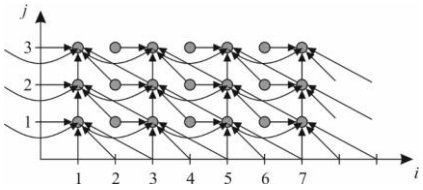

Рис. 3

Синхронна схема обчислень.

Паралельний режим обробки пов'язаний з одночасним переобчисленням значень всіх змінних x_i $(i = \overline{1, n})$, який можна здійснити з допомогою алгоритму:

$$
\begin{aligned}
&FOR\ \ j=1,\ k_1\ \ DO\\
&\{\ FOR\ i=1,\ n\ \ DO\ \ PAR\\
&\ \{\ p_i=0\\
&FOR\ s=-m,\ m\ \ DO\\
&\{\ p_i=p_i+x_{i+s}*f_s\ \}\\
&\ \ x_i=p_i\ \}\ \}.
\end{aligned}
\qquad (6)
$$

Даний паралельний алгоритм задає переобчислення значень всіх змінних, при якому для j-го переобчислення беруться значення змінних, одержані виключно під час (j-1)-го переобчислення. Для забезпечення цього необхідна синхронізація паралельних гілок після виконання чергового переобчислення. Наведений алгоритм є зорієнтований на реалізацію на SIMD-обчислювальних системах.

Для $m=2,\ n=7,\ k_1=4$ інформаційні зв'язки між ітераціями (i, j) алгоритму (6) зображені на рис. 4. Для кращої наглядності ці зв'язки подано лише для окремих ітерацій.

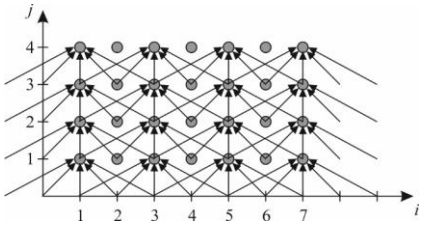

Рис. 4

Асинхронна схема обчислень.

Якщо в (6) не забезпечувати синхронізацію паралельних гілок після виконання чергового переобчислення, то згаданий алгоритм буде працювати за асинхронною схемою: переобчислення значення деякої змінної на заданому кроці відбувається незалежно від переобчислень значень інших змінних, при цьому як аргументи використовуються поточні значення змінних.

Метод гіперплощин.

Використовуючи відомий **метод гіперплощин** для розпаралелювання циклів стосовно послідовного алгоритму розв'язання сформульованої ЗЦФ, одержуємо ***<u>паралельний алгоритм</u>***, який зберігає інформаційний режим обробки послідовного алгоритму і має такий вигляд:

$$
\begin{aligned}
&FOR\ \ t=1,\ n+(k-1)(1+m)\ \ DO\\
&\{\ FOR\ \ ALL\ \ (j,i)\in\{(j,i):\ i=(m+1)(1-j)+t\wedge 1\le j\le k\wedge 1\le i\le n\}\ \ DO\ \ PAR\\
&\ \{\ p_i=0\\
&\ FOR\ \ s=-m,\ m\ \ DO\\
&\ \{\ p_i=p_i+x_{i+s}*f_s\ \}\\
&\ \ x_i=p_i\ \}\ \}.
\end{aligned}
$$

Наведена конструкція задає незалежне (одночасне, паралельне) виконання ітерацій, які знаходяться на одній гіперплощині (прямій).