О.М. ХІМІЧ, О.В. ЧИСТЯКОВ, БРУСНІКІН В.М.

ГІБРИДНИЙ АЛГОРИТМ УЗАГАЛЬНЕНОГО МЕТОДУ СПРЯЖЕНИХ ГРАДІЄНТІВ ДЛЯ ПРОБЛЕМИ ВЛАСНИХ ЗНАЧЕНЬ З СИМЕТРИЧНИМИ РОЗРІДЖЕНИМИ МАТРИЦЯМИ

1. Вступ

Велика кількість наукових та практичних задач, зокрема при дослідженні стійкості конструкцій, розрахунку динаміки напружено-деформованого стану об'єктів різної природи та інші зводяться до розв'язання часткової алгебраїчної проблеми власних значень з симетричними додатно-визначеними матрицями розрідженої структури великої розмірності [1]. Такі задачі можуть бути ефективно розв'язані за допомогою однокрокових та двокрокових ітераційних методів. У статті [2] розглядаються явні та неявні однокрокові ітераційні методи для часткової проблеми власних значень, серед яких: метод найшвидшого спуску, метод обернених ітерацій, степеневий метод та поперемінно-трикутний метод. Також висвітлено ряд питань щодо вибору параметрів та передобумовлювачів для підвищення швидкодії методів, доведено збіжність представлених методів. У роботі [3] розглянуто практичні аспекти розробки високоефективних паралельних однокрокових алгоритмів для MIMD-комп'ютерів та комп'ютерів гібридної архітектури, приведено результати обчислювальних експериментів на інтелектуальній робочій станції гібридної архітектури Інпраком_G.

У роботах Савинова В.Г. [4], [5] розглядаються двокрокові схеми на прикладі узагальненого методу спряжених градієнтів. Для даного методу доводиться збіжність, подаються рекомендації щодо вибору ітераційних параметрів, а також пропонується використовувати регуляризатор на основі методу симетричної верхньої релаксації для розв'язання систем лінійних алгебраїчних рівнянь.

В прикладному програмному забезпеченні бібліотечного виду Lis (Library of Iterative Solvers for Linear Systems) [6] реалізовано паралельні алгоритми для розв'язання на MIMD-комп'ютерах задач на власні значення з розрідженими матрицями з використанням прямих методів, що базуються на методах Крилова, Ланцоша, Арнольді, а також ітераційних методів, що базуються на степеневому методі та методі простої ітерації. Бібліотека паралельних програм SLEPc (Scalable Library for Eigenvalue Problem) [7] для розв'язування задач на власні значення надає інтерфейси до таких програмних засобів як ARPACK та BLOPEX.

Велика розмірність розріджених матриць в задачах на власні значення, які виникають при розв'язуванні практичних задач, та їх обчислювальна складність потребують використання потужних комп'ютерів для їх розв'язування. Гібридні комп'ютери з багатоядерними процесорами (CPU) та графічними процесорами (GPU − **g**raphics **p**rocessing **u**nits), а також технології GPGPU (General-purpose graphics processing units − GPU загального призначення) дають можливість значно зменшити час розв'язування таких задач.

В даній роботі розглядається задача на власні значення:

$$Ax = \lambda x, \qquad (1)$$

де A – симетрична розріджена додатно-визначена матриця порядку n, λ та x − власне значення та відповідний власний вектор.

Для розв'язання алгебраїчної часткової проблеми власних значень для симетричних додатно-визначених розріджених матриць на комп'ютерах гібридної архітектури пропонується неявний гібридний алгоритм спряжених градієнтів. Досліджено ефективність розробленого алгоритму та подано результати його апробації на комп'ютері гібридної архітектури.

Для знаходження мінімального власного значення використовується узагальнений метод спряжених градієнтів [4]:

$$x^{k+1} = \begin{cases} p^k / \left\| p^k \right\|_2, \alpha_k = +\infty; \\ x^k + \alpha_k p^k, k \neq N-1, 2N-1, ...; \\ \dfrac{x^k - \alpha_k p^k}{\left\| x^k - \alpha_k p^k \right\|_2}, k = N-1, 2N-1, ... \end{cases} \quad (2)$$

Тут ми маємо:

$$p^k = B(x^k) f(x^k) - \beta_k p^{k-1};$$

$$\beta_k = \begin{cases} 0, \quad k = 0, 2N, ...; \quad \alpha_{k-1} = +\infty; \\ \dfrac{(B(x^k) f(x^k), [A - \mu(x^k) I] p^{k-1})}{(p^{k-1}, [A - \mu(x^k) I] p^{k-1})}, \quad k = 1,2,3,\ldots; \end{cases}$$

$$f(x) = [A - \mu(x) I] x / \|x\|_2, \qquad \mu(x) = \frac{(Ax, x)}{(x, x)};$$

$B(x^k)$ – деяка матриця, яка вибирається із умов прискорення збіжності ітераційного процесу, N – момент відновлення, параметр α_k може бути вибраний як точка локального мінімуму функціоналу $\mu(x^k - \alpha p^k)$.

Ітераційний процес закінчується за критерієм:

$$\frac{\left| \mu^{(k+1)} - \mu^{(k)} \right|}{\mu^{(k)}} \leq eps.$$

Для узагальненого методу спряжених градієнтів справедлива наступна теорема [4].

Теорема

Якщо $m\|y\|_2^2 \leq (B(x)y, y) \leq M\|y\|_2^2$, $x, y \in R_n$, $0 < m \leq M < +\infty$ та $N < +\infty$, то ітераційний процес збігається для будь-якого початкового наближення x^0, причому: $\lim\limits_{k\to\infty} \mu(x^k) = \lambda_i$, $\lim\limits_{k\to\infty} x^k = v_i$, де λ_i, v_i – одне із власних значень та відповідний йому власний вектор матриці A.

Зазначимо, що при практичній реалізації методу через помилки заокруглення процес буде збігатися до λ_1, v_1.

При створенні ефективного паралельного алгоритму методу спряжених градієнтів необхідно враховувати, що виконання ітерацій методу здійснюється послідовно, тому доцільно розглядати математичні операції, які можуть бути розпаралелені на кожній ітерації. Аналіз послідовного варіанту алгоритму (2) показує, що найбільш трудоємкими операціями (підзадачами) на кожній ітерації за затратами обчислювальних ресурсів і часом виконання є множення розрідженої матриці на вектор та розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) з трикутними матрицями. Саме ці підзадачі і підлягають розпаралеленню з урахуванням архітектурних особливостей обчислювального середовища гібридного комп'ютера.

Для ефективного виконання гібридного алгоритму необхідно забезпечувати:

- ефективні способи збереження розріджених матриць для виконання матрично-векторних операцій на CPU і GPU;
- розв'язування підзадач, які потребують найбільших затрат комп'ютерного часу на GPU;

- рівномірне завантаження процесів, що використовуються при розв'язуванні задачі (балансування навантаження);
- синхронізацію обмінів між процесами CPU;
- мінімізацію обмінів між процесами CPU та між CPU і GPU.

2. Приведенення матриці до блочно-діагонального вигляду з обрамленням

З метою підвищення ефективності розв'язання задачі з розрідженими матрицями великих розмірностей була використана ідея попереднього приведення вхідної розрідженої матриці A за допомогою методу паралельних перерізів до блочно-діагональної матриці з обрамленням [8]. Такий підхід дає змогу використовувати більш природній підхід до розпаралелення обчислень на гібридному комп'ютері. Представлення розрідженої матриці у вигляді блочно-діагональному з обрамленням дає можливість ефективно виконувати матрично-векторні операції, як на CPU, використовуючи, наприклад, бібліотеку обчислювальної математики Intel MKL [9], так і на GPU з використанням бібліотек програм матрично-векторних операцій cuBLAS [10] та cuSPARSE [11].

Схематично матриця, яка приведена до блочно-діагонального вигляду з обрамленням, має вигляд:

$$\widetilde{A} = P^T AP = \begin{pmatrix} D_1 & 0 & 0 & \dots & 0 & B_1 \\ 0 & D_2 & 0 & \dots & 0 & B_2 \\ 0 & 0 & D_3 & & 0 & B_3 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & & D_{p-1} & B_{p-1} \\ B_1 & B_2 & B_3 & \dots & B_{p-1} & D_p \end{pmatrix},$$

де P – матриця перестановок, а блоки D_i і B_i зберігають розрідженість.

Блочно-діагональна матриця може бути представлена як $\widetilde{A} = I - \widetilde{L} - \widetilde{L}^T$, а матриця $\widetilde{L}$ є нижньою трикутною блочно-діагональною матрицею з обрамленням. Матриця $\widetilde{L}$ складається з трикутних матриць $\widetilde{D}_i$ на діагоналі та розріджених матриць довільної структури $\widetilde{C}_i$ в обрамленні, а саме:

$$\widetilde{L} = \begin{pmatrix} \widetilde{L}_1 \\ \widetilde{L}_2 \\ \widetilde{L}_3 \\ \vdots \\ \widetilde{L}_{p-1} \\ \widetilde{L}p \end{pmatrix} = \begin{pmatrix} \widetilde{D}_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & \widetilde{D}_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \widetilde{D}_3 & \dots & 0 & 0 \\ \vdots & \vdots & & \ddots & & \dots \\ 0 & 0 & 0 & & \widetilde{D}_{p-1} & 0 \\ \widetilde{C}_1 & \widetilde{C}_2 & \widetilde{C}_3 & \dots & \widetilde{C}_{p-1} & \widetilde{D}_p \end{pmatrix}. \qquad (3)$$

Таким чином, початкова задача зводиться до наступної еквівалентної:

$$\widetilde{A}y = \lambda y .$$

3. Неявний ітераційний гібридний алгоритм методу спряжених градієнтів

Для покращення швидкості збіжності ітераційного процесу використовується оператор (регуляризатор) *B(x)*, який можна вибирати на основі методу верхньої симетричної релаксації для розв'язання систем лінійних алгебраїчних рівнянь [4].

В цьому випадку:

$$\hat{A}(x) = \frac{1}{1-\mu(x)}(\widetilde{A}(x) - \mu(x)I) = I - \hat{L}(x) - \hat{L}^T(x);$$

$$B(x) = \omega(2-\omega)(I - \omega\hat{L}^T(x))^{-1}(I - \omega\hat{L}(x))^{-1};$$

$$\hat{L}(x) = \frac{1}{1-\mu(x)}\tilde{L}(x), \quad \text{де } \omega \text{ – параметр релаксації, } \omega \in (1;2).$$

З огляду на структуру $\hat{L}(x)$ реалізується такий розподіл блоків (підматриць) $\hat{L}(x)$ по процесах CPU: процеси з номерами $0 \le i < p$ зберігають блоки $\hat{D}_i(x)$ та $\hat{C}_i(x)$, а процес з номером p–1 зберігає блок $\hat{D}_i(x)$. Тут p – загальна кількість процесів.

Паралельна реалізація алгоритму в основному визначається блочно-трикутною структурою матриць $\hat{L}_i(x)$ (3) при розв'язанні системи:

$$(I - \omega\hat{L}^T(x))(I - \omega\hat{L}(x))w = r(x);$$
$$r(x) = \omega(2-\omega)f(x);$$
$$w^k = p^k - \beta_k p^{k-1}.$$

Алгоритм розв'язання нижньої трикутної системи $(I - \omega\hat{L}^T(x))y = r$ зводиться до одночасного та незалежного розв'язування трикутних систем на окремих процесах гібридного комп'ютера з використанням GPU:

$$(I - \omega\hat{D}_q(x))y_q = r_q, \ 1 \le q < p,$$

та наступного обчислення $\tilde{y}_q = \hat{C}_q(x)y_q, \ 0 \le q < p,$ де q – номер процесу.

Далі всі процеси надсилають $\tilde{y}_q$, в останній процес, в якому обчислюється y_p, розв'язуючи систему $(I - \omega\hat{D}_p(x))y_p = r_p - \sum_{q-1}^{p-1}\tilde{y}_q$.

Аналогічно розв'язується система: $(I - \omega\hat{L}(x))w = y$, де p-й процес розв'язує систему $(I - \omega\hat{L}(x))w_p = y_p$, а потім розсилає компоненти w_p іншим процесам, які незалежно розв'язують системи $(I - \omega\hat{L}(x))w_q = y_q - \hat{C}_q^{\mathrm{T}}(x)w_p$.

4. Оцінки ефективності неявного гібридного алгоритму спряжених градієнтів

Розглядаючи розроблений гібридний алгоритм зазначимо, що основними, за складністю, арифметичними операціями є множення розрідженої матриці на щільний вектор та розв'язання двох трикутних систем.

Для однієї операції множення розрідженої матриці на щільний вектор кількість операцій дорівнює $M \approx \sum_{i=1}^{n}\eta_i^2$ операцій. Для розв'язання однієї трикутної системи справедлива така оцінка складності послідовного алгоритму: $N \approx \sum_{i=1}^{n}\frac{\eta_i^2 - \eta_i}{2}$ операцій.

Для визначення оцінок ефективності гібридного алгоритму, в якому всі арифметичні операції в основному виконуються на GPU, введемо такі позначення: p – загальна кількість графічних процесів, n – розмірність матриці, η_i − кількість ненульових елементів у i-му рядку нижньої трикутної матриці, k_0 – кількість арифметичних операцій, що виконуються на одному GPU одночасно, t – час виконання однієї арифметичної операції на GPU, t_0 – час обміну одним блоком між GPU, t_1 – час обміну одним блоком між CPU та GPU. Загальний час розсилки даних можна оцінити як $t_0 \log_2 p$.

Позначимо T_1 – час виконання алгоритму на одному GPU, а T_p – алгоритму з використанням p GPU. Тоді маємо наступні наближені співвідношення:

$$T_1 = q\left(\frac{2t}{k_0}(M+N)+t_1\right)b, \quad T_p = q\left(2\frac{M}{pk_0}t+\max\left(\frac{2N}{pk_0}t, t_0\log_2 p\right)+t_1\right)b,$$

де q – емпіричний коефіцієнт, що залежить від рівномірності навантаження на обчислювальний пристрій, від формату даних, що оброблюються та архітектурних особливостей обчислювальної системи, $q \in (0.8;1.2)$, b – кількість ітерацій.

Провівши грубе заокруглення, отримаємо оцінки прискорення та ефективності алгоритму:

$$S_p = \frac{T_1}{T_p} \approx \begin{cases} p, & \frac{N}{pk_0}t \geq t_0\log_2 p; \\ \frac{2t(M+N)p + t_1pk_0}{2Mtp + t_0pk_0\log_2 p + t_1pk_0}, & \frac{N}{pk_0}t < t_0\log_2 p \end{cases},$$

$$E_p = \frac{S_1}{p} \approx \begin{cases} 1, & \frac{N}{pk_0}t \geq t_0 l\log_2 p; \\ \frac{2t(M+N) + t_1k_0}{2Mt + t_0k_0 l\log_2 p + t_1k_0}, & \frac{N}{pk_0}t < t_0\log_2 p \end{cases}.$$

Отримані оцінки дають змогу стверджувати, що розроблений гібридний алгоритм забезпечує високе прискорення, оскільки наявність у дільниках комунікаційних складових величини n^2 сприяє зменшенню частки часу, необхідного на обміни та синхронізацію процесів при великих порядках системи.

5. Особливості програмної реалізації гібридного алгоритму спряжених градієнтів для розв'язування часткової проблеми власних значень з розрідженими матрицями

5.1. Формати збереження розріджених матриць. При створенні алгоритмів розв'язування задач з розрідженими матрицями велике значення має вибір способів задання та збереження ненульових елементів. Ці способи визначаються насамперед структурою (розміщенням ненульових елементів) розрідженої матриці, вимогами алгоритму, який використовується, а також архітектурою гібридного комп'ютера.

Складність ефективної обробки розріджених матриць привела до створення великої кількості форматів їх збереження в комп'ютері, наприклад, CSR (compressed sparse row), CSC (compressed sparse column), COO (coordinate) та інші. Проте не можна рекомендувати єдиний, близький до оптимального, формат зберігання даних. Це у великій мірі залежить від виду розрідженої матриці. В працях багатьох авторів, наприклад [8], проведено аналіз різних форматів збереження розріджених матриць, який показав, що формат CSR, який відноситься до найбільш універсальних і дозволяє легко реалізувати оптимальний алгоритм множення розрідженої матриці на вектор на CPU, не завжди забезпечує таку продуктивність для цієї операції на GPU. У ряді випадків виявляється, що переваги мають модифіковані формати, які враховують розрідженість матриці, архітектурні особливості гібридного комп'ютера та організації пам'яті на GPU.

Оскільки розріджені матриці великої розмірності потребують великих об'ємів обчислювальних ресурсів для їх збереження в комп'ютері та витрат часу на їх обробку, виникає потреба у виконанні великої кількості додаткових операцій при їх переформатуванні або переіндексації для забезпечення масштабованості обчислювального процесу. Тому був розроблений комбінований формат збереження матриць, що базується на CSR форматі та враховує особливості представлення блочно-діагональної розрідженої матриці з обрамленням. Схематично комбінований формат представлено на рис. 1.

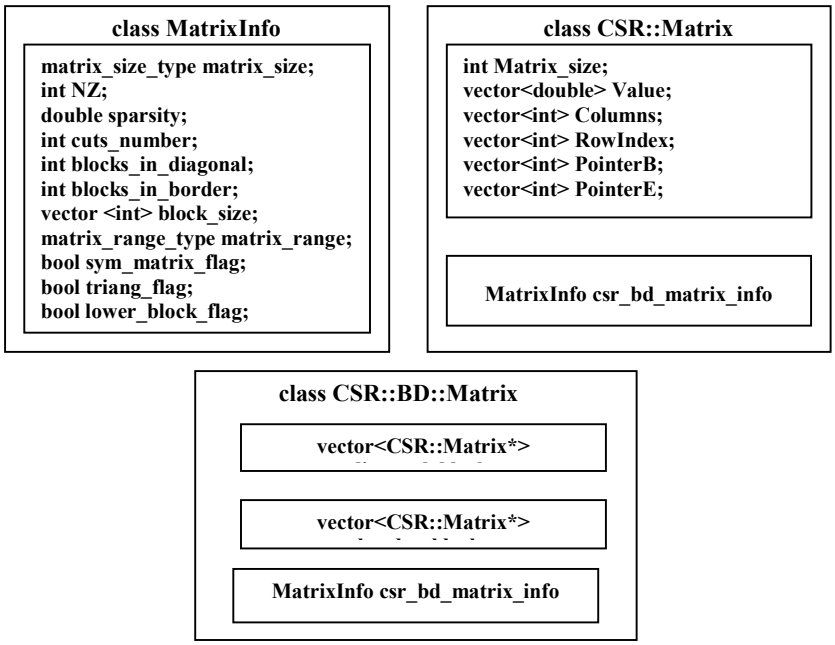

Рис. 1. Комбінований формат збереження блочно-діагональної розрідженої матриці з обрамленням

Зі схеми на рис. 1 видно, що блочно-діагональна матриця з обрамленням зберігається у class CSR::BD::Matrix в масивах вказівників на діагональні блоки та блоки з обрамленням. Тобто кожен блок є окремою підматрицею, збереженою у модифікованому форматі CSR (class CSR::Matrix).

Такий підхід дозволяє маніпулювати кожним блоком незалежно та забезпечувати можливість розпаралелення обчислень, як на MIMD-комп'ютерах, так і на комп'ютерах гібридної архітектури.

Class CSR::BD::Matrix зберігає інформацію про вхідну матрицю class MatrixInfo, а саме:

- розмірність матриці, кількість ненульових елементів та її розрідженість;
- кількість заданих перерізів при перетворенні матриці;
- кількість отриманих блоків у діагоналі та в обрамленні;
- характеристики матриці такі як симетричність, представлення матриці у трикутному вигляді, та місце розташування трикутника (над чи під головною діагоналлю);
- базис матриці (нульовий чи одиничний).

В свою чергу class CSR::Matrix зберігає підматриці у модифікованому форматі CSR, тобто поєднує у собі 3-х та 4-х векторний формат збереження, а саме зберігаються такі масиви:

- vector<double> Value – ненульові елементи матриці;
- vector<int> Columns – позиція ненульового елементу у рядку;
- vector<int> RowIndex – кількість ненульових елементів у кожному рядку;
- vector<int> PointerB – кількість ненульових елементів у кожному попередньому рядку;
- vector<int> PointerE – кількість ненульових елементів у кожному наступньому рядку;
- MatrixInfo csr_bd_matrix_info – містить інформацію про кожну окрему підматрицю та її розташування у блочно-діагональній матриці з обрамленням.

Таким чином, отримано уніфікований 5-ти рядковий формат збереження розрідженої матриці, що дозволяє використовувати відомі високопродуктивні бібліотеки обчислювальної математики. Крім того, якщо в процесі обчислень додаткові масиви не

потрібні, то вони видаляться автоматично, тим самим досягається більш оптимальне використання оперативної пам'яті.

Такий формат є ефективним як для використання на CPU, так і на GPU. Він дає змогу, наприклад, застосовувати до блоків матриць в одній і тій же програмі, без проведення переіндексації їх елементів, як функції розпаралелення системи MPI, так і програми матрично-векторних операцій з бібліотеки Intel MKL. Тим самим забезпечується більш ефективне використання обчислювальних ресурсів та висока масштабованість гібридних алгоритмів. Серед переваг обраного формату можна також зазначити прозорий підхід до проведення обчислень, передачі повідомлень між обчислювальними процесами, зменшення витрат часу на розробку, відлагодження та тестування гібридних програм.

5.2. Налаштування гібридних алгоритмів та програм на обчислювальні ресурси гібридного комп'ютера. В сучасних гібридних комп'ютерах відмічається постійне оновлення типів обчислювальних вузлів та відеокарт, розробляються нові версії систем паралельного програмування MPI та CUDA, а також бібліотек програм обчислювальної математики, наприклад Intel MKL, NVidia CuBLAS, CuSparse тощо. Використання найновіших розробок програмно-технічного устаткування сприяє підвищенню ефективності гібридних алгоритмів та програм. Тому необхідно передбачити такий спосіб їх розробки, щоб можна було легко вносити відповідні зміни зі змінами в обчислювальних ресурсах гібридного комп'ютера.

З цією метою при створенні гібридного алгоритмічно-програмного засобу для розв'язування часткової проблеми власних значень для розріджених матриць ітераційними алгоритмами було використано концепцію «функцій-обгорток» [12]. «Обгортка» (англ. Wrapper) – це функція, яка є проміжною ланкою між прикладними функціями або програмами та іншою бібліотекою чи програмним інтерфейсом, а також може модифікувати або узагальнювати інтерфейси прикладних програм. При цьому реалізується модульний принцип програмування, який дає можливість при оновленні програмного забезпечення вносити поправки лише в окремі програмні модулі. Комбінований формат збереження блочно-діагональної розрідженої матриці з обрамленням дозволяє ефективно використовувати концепцію «функцій-обгорток».

До «обгорток» винесені службові функції для роботи з графічним прискорювачем, наприклад, `cusparseCreate(…)`, `cudaMemcpy(…)` та інші.

Блок-схему функціонального складу розробленого алгоритмічно-програмного засобу представлено на рис. 2, до якого входять:

- `Data structures` – модуль, що відповідає за структури збереження даних та їх обробку;
- `Wrappers function` – модуль, в якому реалізовуються «функції-обгортки»;
- `Hight performance computing libraries` – зовнішні високопродуктивні бібліотеки та інтерфейси паралельного програмування;
- `Statistics functions` – модуль, що відповідає за збір, збереження та обробку статистичних даних, підвищення ефективності обчислювального процесу, а також за обробку виключних та помилкових ситуацій;
- `Computing methods functions` – модуль, в якому реалізовано обчислювальні алгоритми (для своєї роботи цей модуль використовує вище приведені модулі);
- `External user interface` – модуль, що реалізує інтерфейс користувача.

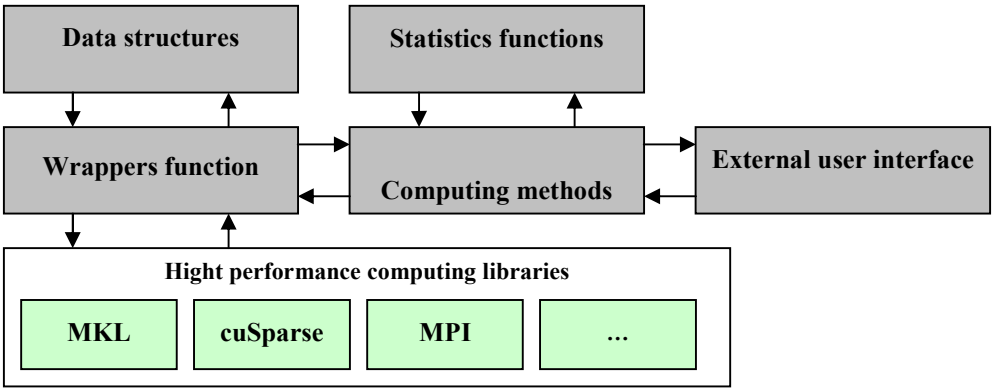

Рис. 2 Блок-схема гібридного алгоритмічно-програмного засобу

Такий підхід до розробки гібридного алгоритмічно-програмного засобу дозволяє отримати наступні переваги:

- розширюваність – за рахунок малої залежності між модулями можна легко додавати нові функціональні можливості, наприклад, підтримку нових високопродуктивних бібліотек або структур збереження даних;
- зручне налагодження алгоритмічно-програмного засобу на гібридні комп'ютери з різними математичними та технічними характеристиками;
- зменшення часу на відлагодження та супровід програмного засобу – відлагодження модулів відбувається лише один раз, і в разі повторного використання повторне тестування не потрібно проводити.

6. Апробація гібридного алгоритму спряжених градієнтів

Для експериментального дослідження гібридного алгоритму, що розглядається, було використано матриці з колекції Флоридського університету [13] (табл. 1).

Таблиця 1. Набір тестових матриць з Флоридської колекції розріджених матриць

Назва задачі	Проблемна область	Порядок матриці	Кількість ненульових елементів
Bmwcra_1	structural problem	148,770	10,641,602
Bone010	Model reduction problem	986,703	47,851,783
Emila_923	structural problem	923,136	40,373,538

Обчислювальні експерименти проводилися на інтелектуальній робочій станції гібридної архітектури Інпарком-G [14] з такими технічними характеристиками: чотири вузли з двома 4-х-ядерними Intel Xeon E5606 процесорами, оперативна пам'ять: 3 Гб на одне фізичне обчислювальне ядро. Програми, які реалізують гібридні алгоритми, написано на мові програмування C++ з використанням системи розпаралелювання MPI, бібліотеки Intel MKL для MIMD-комп'ютерата, а також технології розпаралелення CUDA та бібліотек програм cuBLAS, cuSPARSE на GPU, [15].

В табл. 2 приведено порівняльну характеристику знаходження найменшого власного значення на Інпарком-G однокроковим гібридним поперемінно-трикутним алгоритмом [3] та гібридним алгоритмом узагальненого методу спряжених градієнтів при використанні різної кількості ядер CPU та процесорів GPU.

Таблиця 2. Порівняльна часова характеристика (сек) гібридним алгоритмів

Гібридний алгоритм узагальненого методу спряжених градієнтів								
Задача	CPU				CPU + GPU			
	1 Core	8 Core	16 Core	32 Core	1GPU	2GPU	4 GPU	8 GPU
Bone	155,37	22,32	11,93	8,39	23,29	14,32	7,92	4,52
Bmwcra_1	306,55	43,18	24,48	13,24	43,18	24,27	13,73	8,06
Emilia_923	489,74	69,47	38,08	20,53	66,09	35,28	22,41	13,41
Гібридний алгоритм неявного методу скорішого спуску								
Bone	423,80	55,76	31,68	18,00	67,05	29,47	17,86	11,90
Bmwcra_1	820,74	115,27	62,64	34,04	117,41	56,02	30,78	18,76
Emilia_923	1185,52	173,06	90,13	46,94	161,07	77,99	44,31	26,06

З табл. 2 видно, що за неявним гібридним алгоритмом методу спряжених градієнтів на MIMD-комп'ютері найбільше прискорення обчислень при використанні 32 процесів у порівнянні з послідовною версією програми одержано від 18 до 23 разів. Використання одного GPU дає прискорення обчислень в 6,5 – 7,5 разів, а при масштабування гібридної системи до восьми GPU − в 34 – 38 разів у порівнянні з послідовною версією програми.

За гібридним алгоритмом поперемінно-трикутного методу на MIMD-комп'ютері найбільше прискорення обчислень при використанні 32 процесів у порівнянні з послідовною версією програми одержано від 22 до 25 разів. При використанні одного GPU прискорення обчислень складає 6,3 – 7,3 разів, а при використанні восьми GPU − в 35 – 45 разів у порівнянні з послідовною версією програми.

Отже, порівнюючи часи виконання задач при однакових вхідних даних та критеріях закінчення ітераційного процесу двокроковий неявний гібридний алгоритм методу спряжених градієнтів забезпечує прискорення у 2,0 – 2,8 разів у порівнянні з однокроковим гібридним алгоритмом поперемінно-трикутного методу.

7. Висновки

Обчислення власних значень та відповідних їм векторів для розріджених матриць великих розмірностей потребують значних обчислювальних ресурсів. В роботі запропоновано високопродуктивний двокроковий неявний гібридний алгоритм методу спряжених градієнтів для розв'язання алгебраїчної часткової проблеми власних значень для симетричних додатно-визначених розріджених матриць на гібридних комп'ютерах.

Створений алгоритм забезпечує суттєве скорочення часу обчислень, тобто відмічається пропорціональне зменшення часу розв'язування задачі зі збільшенням кількості процесорних ядер MIMD-комп'ютера та кількості графічних прискорювачів на гібридних обчислювальних системах.

Програма, що реалізує розроблений гібридний алгоритм, входить до алгоритмічно-програмного забезпечення інтелектуальної робочої станції Інпарком-G і може використовуватися для розв'язання науково-технічних задач.

СПИСОК ЛІТЕРАТУРИ

1. Приказчиков В.Г. Прототипы итерационных процессов в задаче на собственные значения. // Дифференциальные уравнения. – 1980. – том 16. – №. 9. – С. 1688 − 1697.
2. Приказчиков В.Г., Химич А.Н. Итерационные методы решения задач устойчивости и колебания пластин и оболочек. // Прикладная механика. – 1984. – том 20. – №. 1. – С. 88 − 94.
3. Хіміч О.М., Чистяков О.В. Паралельні однокрокові ітераційні методи розв'язання алгебраїчної проблеми власних значень для розріджених матриць. // Компьютерная математика». – 2014. № 2. – С. 81 − 88.

4. Савинов Г.В. Исследование сходимости одного обобщенного метода сопряженных градиентов для определения экстремальных собственных значений матрицы. // Зап. Науч. сем. ЛОМИ. – 1980. – том 111. – С. 145-150.

5. Савинов Г.В. Обобщенный метод сопряженных градиентов для решения линейных систем. // Зап. Науч. сем. ЛОМИ. –1978, – том 80. – С. 181–188.

6. Lis (Library of Iterative Solvers for Linear Systems) [Електронний ресурс] / Akira Nishida. Japan. – Режим доступу: http://www.ssisc.org/lis/index.en.html.

7. SLEPc (Scalable Library for Eigenvalue Problem Computations) [Електронний ресурс] / Universitat Politècnica de València (Spain). – Режим доступу: http://slepc.upv.es/

8. Писсанецки С. Технология разреженных матриц. // М.: Мир. – 1988. –411 с.

9. Intel Maths Kernel Library. [Електронний ресурс] / Intel. – Режим доступу: http://software.intel.com/en-us/intel-mkl.

10. cuSPARSE (The NVIDIA CUDA Sparse Matrix library). [Електронний ресурс] / NVIDIA. – Режим доступу: http://docs.nvidia.com/cuda/cusparse.

11. cuBLAS (The NVIDIA CUDA Basic Linear Algebra Subroutines). [Електронний ресурс] / NVIDIA. – Режим доступу: http://developer.download.nvidia.com/CUBLAS Library.pdf.

12. Wrapper function. [Електронний ресурс] /Olaf Davis. – Режим доступу: http://en.wikipedia.org/wiki/Wrapper_function

13. The University of Florida Sparse Matrix Collection. [Електронний ресурс] / University of Florida. – Режим доступу: http://www.cise.ufl.edu/research/sparse/matrices.

14. Химич А.Н., Молчанов И.Н., Мова В.И. и др. Численное программное обеспечение MIMD–компьютера Инпарком. // – Киев: Наукова думка, – 2007. – 222 с.

15. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. // – М.: Пресс, – 2010. – с. 232.