УДК 004.932.4

Л.П. Фельдман, д-р техн. наук, професор,
І.А. Назарова, канд. техн. наук, доц.,
Я.О. Гризадубова, студент
Донецький національний технічний університет, г. Покровськ, Украина
ianazard@gmail.com

Ефективність паралельної реалізації екстраполяційних методів чисельного розв'язання СЗДР на графічних процесорах

Запропоновано методи підвищення ефективності паралельного чисельного розв'язання багатовимірної задачі Коші на базі технології локальної екстраполяції. Отримано послідовний алгоритм локальної екстраполяції, що мінімізує обчислювальні витрати. Розроблено екстраполяційні паралельні алгоритми на базі явних, повністю неявних та блокових опорних методів. Отримані екстраполяційні методи реалізовано на графічних процесорах із використанням технології CUDA.

Ключові слова: задача Коші, графічні процесори, паралельні екстраполяційні методи, технологія CUDA.

Вступ

Невід'ємну частину математичного моделювання поведінки динамічних об'єктів різної природи складають системи звичайних диференційних рівнянь (СЗДР). З огляду на розмір і складність сучасних моделей, для застосування апарату чисельних обчислень при інтегруванні СЗДР ресурсів послідовного комп'ютера часто виявляється недостатньо або їх використання не є ефективним. Проблема продуктивності чисельного інтегрування вирішується методами паралельного програмування. Зокрема, одним із сучасних напрямків є використання графічних процесорів загального призначення. Однак, зважаючи на особливості чисельних методів і архітектури багатопроцесорних обчислювальних систем, паралельна реалізація чисельного інтегрування також являє собою нетривіальну задачу і вимагає окремого дослідження. Виходячи з цього, в даний час все більш широко використовується новий напрямок в прискоренні обчислень – це застосування графічних процесорних пристроїв. Отже, реалізація високопродуктивних засобів моделювання на графічних процесорах загального призначення є актуальним завданням.

Метою дослідження являється підвищення ефективності паралельної реалізації екстраполяційних методів розв'язання багатовимірної задачі Коші для систем звичайних диференційних рівнянь за рахунок використання графічних процесорів.

Завдання дослідження: розробка послідовної реалізації технології локальної екстраполяції Річардсона (ЛЕР), мінімізуючої накладні витрати; конструювання паралельних методів ЛЕР з явними, повністю неявними та блоковими опорними методами; реалізація методів на графічних процесорах: GPU NVIDIA з використанням технології CUDA.

Особливості застосування технології локальної екстраполяції Річардсона (ЛЕР)

Методи інтегрування динамічних задач із зосередженими параметрами, які описуються системами звичайних диференційних рівнянь:

$$\begin{cases} \dfrac{d\bar{y}(x)}{dx} = \bar{f}(x,\bar{y}(x)), \\ \bar{y}(x_0) = \bar{y}_0, \end{cases} \quad F = \bar{f} : R \times R^m \to R^m, \tag{1}$$

незважаючи на наявність всебічних знань у цій області, продовжують залишатися джерелом активних досліджень, особливо в такому новому науковому напрямку, як паралельні обчислення. Це особливо важливо для науково-технічних задач великих розмірностей, складних правих частин та для швидкого моделювання динамічних процесів у реальному часі. Одним із головних питань, що виникають під час чисельного вирішення СЗДР, є проблема оцінки похибки наближеного розв'язку. Апостеріорна оцінка локальної похибки, яка отримується на кожному кроці обчислень, дозволяє автоматично вибирати крок інтегрування, що забезпечує задану точність чисельного розв'язку. Загальноприйнятими методами оцінки локальної апостеріорної похибки розв'язання СЗДР є: дублювання кроку за правилом Рунге, технологія локальної екстраполяції та вкладені методи [1-5].

Метод локальної екстраполяції Річардсона є узагальненням технології подвоєння кроку за правилом Рунге. Розв'язання задачі Коші розглядається при переході з точки x_n у точку $x_{n+1} = x_n + H$, де $H-$ базова довжина кроку. Вибирається ряд натуральних чисел

$P_i = \{n_1, n_2, ..., n_k, ..\}$ такий, що: $n_1 < n_2 < ... < n_k < ...$ та, відповідно, послідовність кроків: $h_1 > h_2 > ... > h_{k-1} > h_k > ...$, де $h_i = H / n_i$. Задається опорний чисельний метод порядку r_0 та після виконання n_i кроків інтегрування з довжиною h_i обчислюється наближений розв'язок в точці x_{n+1}: $T_{i,1} = \bar{y}_{h_i}(x_n + H) = \bar{y}^{(i)}_{n+1}$, $i = \overline{1,k}$. За рекурентним співвідношенням визначають $T_{i,j+1}$ – екстрапольовані значення для довільних *i,j*:

$$T_{i,j+1} := T_{i,j} + \frac{T_{i,j} - T_{i-1,j}}{(n_i / n_{i-j})^b - 1}. \qquad (2)$$

Величина $b=1$ в загальному випадку, для симетричних опорних методів $b = 2$. Переваги цього методу полягають у тому, що він дає таблицю результатів обчислень, які утворюють послідовність вкладених методів, дозволяють оцінити локальну похибку та вибрати стратегію для методів змінного порядку.

Екстраполяційна технологія включає опорний чисельний метод розв'язання задачі Коші, послідовність сіток інтегрування, рекурентне правило обчислення значень наближеного розв`язку. Ефективність її застосування безпосередньо залежить від правильного вибору та поєднання всіх трьох складових. Питання конструювання ефективного з точки зору мінімізації обчислювальних витрат послідовного методу інтегрування СЗДР на основі технології Річардсона детально викладено в [2-5].
У якості загальних висновків дослідження: комбінація h^2-екстраполяції на основі симетричних опорних методів та парної послідовності для генерації сіток інтегрування є найбільш ефективними з точки зору обчислювальних витрат. Щодо вибору порядку опорного методу, то показано, що його зростання зменшує довжину екстраполяційної таблиці, але загальні витрати на обчислення апроксимацій вирішення зростають.

Підсумовуючи всі отримані результати, можна зробити висновок, що для зменшення накладних витрат при застосуванні технології локальної екстраполяції, слід вибирати опорний метод малого порядку з достатніми властивостями стійкості. Зауважимо, що екстраполяційна таблиця обчислюється для кожної розмірності вхідної СЗДР. Теоретично немає ніяких обмежень на довжину екстраполяційної таблиці, але кінцівка пам'яті машини, обмежує довжину таблиці зверху і, як правило, використовується: $k_{max} \le 10$. Тому для отримання високоточних додатків $(10^{-15} - 10^{-20})$ немає іншої можливості, як використовувати технологію локальної екстраполяції Річардсона на базі опорного методу високого порядку.

Паралельні методи реалізації технології локальної екстраполяції Річардсона

Потенційно обчислення за технологією локальної екстраполяції містять три джерела внутрішнього паралелізму:

- системний паралелізм (обмежений розмірністю СЗДР);
- паралелізм екстраполяції (обмежений розміром таблиці екстраполяції);
- паралелізм опорного методу (мала ступінь паралелізму).

Після того, як визначені всі складові технології екстраполяції, процес розв'язання можна розбити на дві послідовно виконувані задачі: обчислення k апроксимацій в точці x_{n+1} із різними кроками інтегрування: $\bar{y}_{h_i}(x_n + H)$ та побудова екстраполяційної таблиці. Організація паралельних обчислень може здійснюватися двома способами, звідси і дві принципово різні схеми алгоритму. Перший варіант передбачає використання тільки системного паралелізму, другий - комбінацію паралелізму екстраполяції та системного. Розглянемо першу задачу алгоритму. В якості основної макрооперації для обох варіантів вводиться одноразове обчислення апроксимації точного розв'язку в точці сітки з заданим кроком інтегрування на базі явного опорного методу. Граф впливу першого варіанту наведено на рисунку 1, усі апроксимації розв'язку обчислюються послідовно одночасно для кожної розмірності системи. За другим варіантом паралельно обчислюються k незалежних апроксимацій розв'язку в точці $x_n + H$. Другий варіант алгоритму має більший ступінь паралелізму, $Dop = m \cdot k$, на відміну від першого: $Dop = m$. Проте, цей недолік першої схеми може бути компенсований кращим збалансуванням завантаження багатопроцесорної обчислювальної системи, а також низькою інтенсивністю обмінів.

Обчислення значень екстраполяційної таблиці в залежності від способу розв'язання першої підзадачі, також може бути распаралелено різними способами. Основноа макрооперація для цієї підзадачі - обчислення одного значення $T_{il,j}, j = \overline{1,m}, i = \overline{2,k}; l = \overline{i,k}$ за формулою (2). При побудові екстраполяційної таблиці розпаралелювання може бути здійснено за рахунок незалежного виконання обчислень за формулою поліноміальної екстраполяції по кожній розмірності системи ($Dop = m$), а також за рахунок розподіленого обчислення кожного рядка таблиці ($Dop = k - 1$). Як правило, $m > k$, тому з'являється можливість поєднати обидва види

паралелізму. Граф впливу для методу побудови екстраполяційної таблиці наведено на рисунку 2.

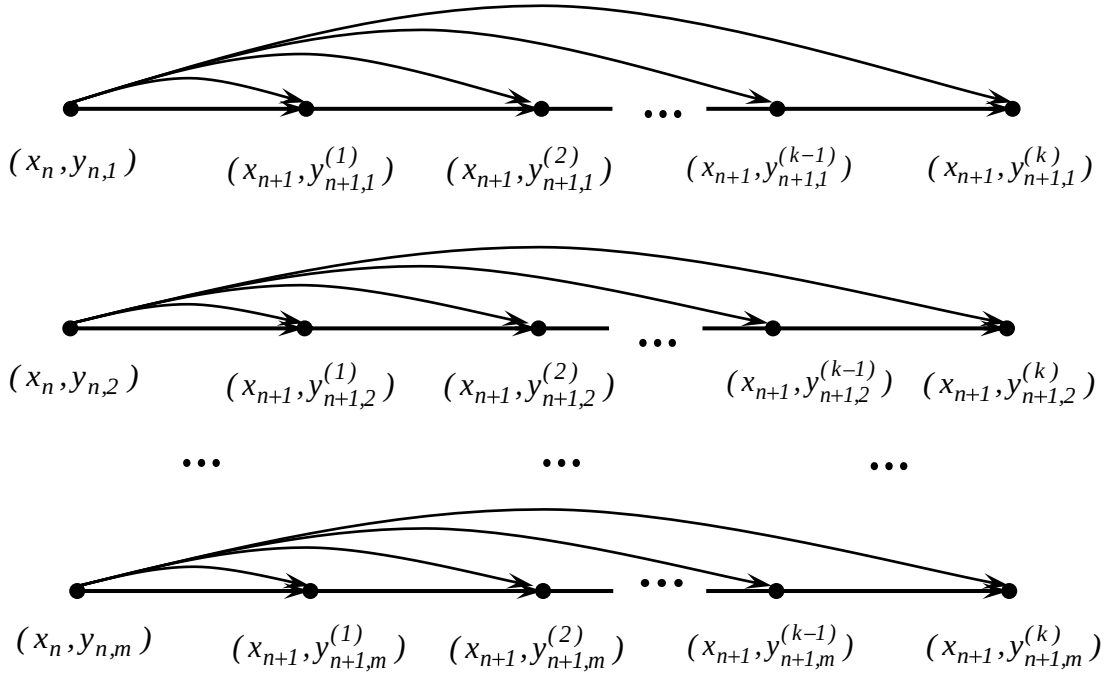

Рисунок 1 – Граф впливу для ЛЕР з системним паралелізмом

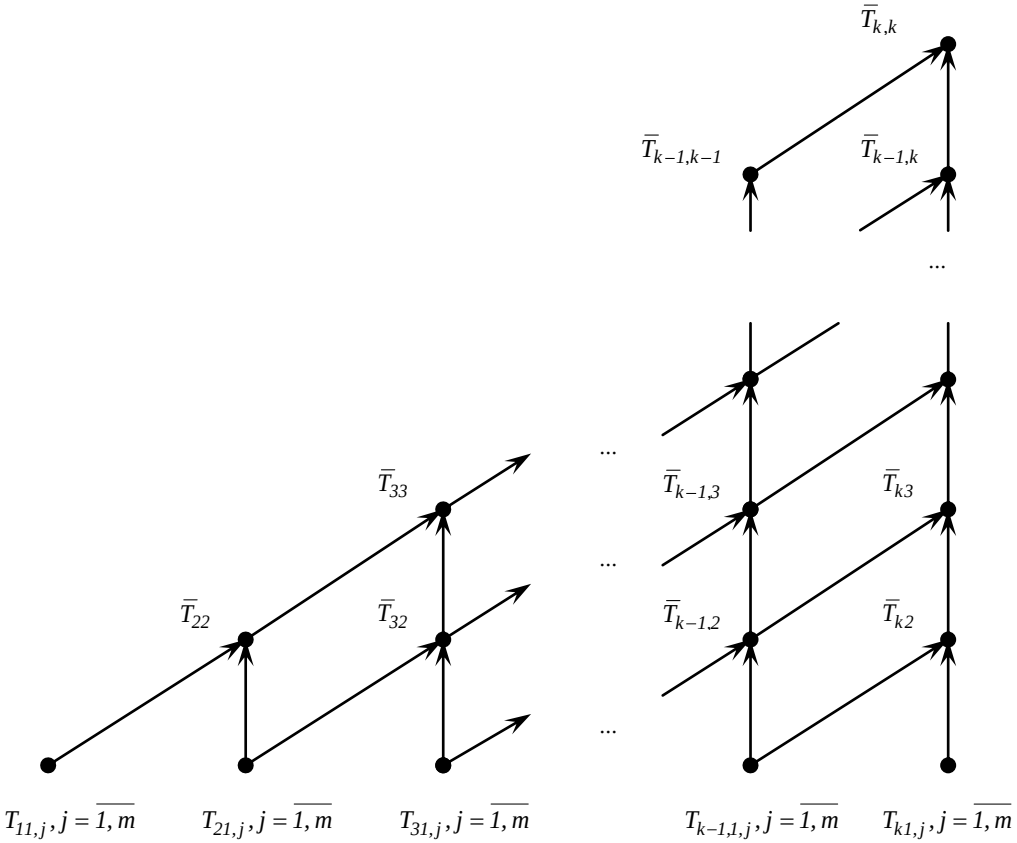

Рисунок 2 – Граф впливу для побудови екстраполяційної таблиці

Методи балансування навантаження у паралельній реалізації екстраполяційної технології

Для СЗДР, що складається з m рівнянь, на одному часовому кроці m_1, $m_1 = \lceil m/p \rceil$ компонент кожної апроксимації розв'язку може бути обчислено паралельно. Відповідно, при побудові екстраполяційної таблиці усі значення $T_{il}, i = \overline{2,k}; l = \overline{i,k}$ визначаються послідовно, але на кожному процесорі обчислюється одна при $p = m$ або $m_1 = \lceil m/p \rceil$ при $p < m$ компонент СЗДР. Міжпроцесорний обмін при першому варіанті розпаралелювання визначається виконанням $N(k)$ разів операції множинного пересилки даних, необхідної при обчисленні коефіцієнтів $\bar{k}_l, l = \overline{1,s}$. Перевагою розглянутої схеми є відсутність необхідності на кожному кроці перегруповувати дані для наступного кроку ітерації, тобто всі обчислення за m_1 компонентами локалізовано на одному процесорі, включаючи й обчислення для таблиці екстраполяції. Паралельний алгоритм розв'язання СЗДР для другої макрооперацийної схеми: якщо кількість процесорів велика, є можливість виконувати макрооперації першої підзадачі паралельно. Для цього необхідно розбити процесори на групи таким чином, щоб забезпечити найбільш раціональне збалансування завантаження багатопроцесорної системи. Розглянемо наступні засоби розбиття процесорів, а відповідно й розподілу даних за процесорами: рівномірний та пропорційний. При найпростішому, рівномірному засобі, кожен з процесорів буде відповідати за обчислення однакової кількості апроксимацій. Якщо розмірність процесорного поля дорівнює p, кількість рядків у екстраполяційній таблиці - k, то при такому засобі буде k груп із $p_i = \lceil p/k \rceil$, $i = \overline{1,k}$ процесорів у кожній. Кожен процесор в i-тій групі буде містити $m_i = \lfloor m/p_i \rfloor$ компонент відповідної апроксимації розв'язку та відповідного екстрапольованого значення. Кожна група буде відповідати за обчислення певної апроксимації розв'язку: $T_{j1}, j = \overline{1,k}$, час вирішення визначається як максимум із $T_{i1}, i = \overline{1,k}$, усі групи, крім останньої, будуть простоювати деякий час.

Розглянемо різні варіанти організації паралельних обчислень екстраполяційної таблиці при рівномірному розподілі для обчислювальної схеми 2. Після паралельного обчислення k апроксимацій розв'язку на k групах процесорів, існує наступний розподіл вхідних даних для обчислення екстрапольованих значень. Перша група процесорів містить вектор $T_{11,j}; j = \overline{1,m}$, друга – $T_{21,j}; j = \overline{1,m}$ і, відповідно, k-та – $T_{k1,j}; j = \overline{1,m}$. Кожен процесор у групах містить однакову m_i кількість компонент. Для обчислень елементів екстраполяційної таблиці може бути реалізовано два варіанти. Перший полягає у використанні системного паралелізму, кожен процесор групи передає відповідному процесору сусідньої групи всі компоненти власного вектора апроксимації розв'язку. Так, перший процесор першої групи передасть першому процесору другої групи подвектор $T_{11,j}; j = \overline{1,m_1}$: для обчислення $T_{22,j}; j = \overline{1,m_2}$; другий процесор першої групи передає $T_{11,j} j = \overline{m_1 + 1, 2m_1}$ другому процесору другої групи для обчислення $T_{22,j}; j = \overline{m_2 + 1, 2m_2}$ і так далі, перший процесор $(k-1)$ групи передає першому процесору k групи значення $T_{k-1,1,j} j = \overline{1,m_{k-1}}$ для обчислення $T_{k2,j}; j = \overline{1,m_k}$. Висота такого паралельного алгоритму дорівнює $k-1$, тобто на першому кроці обчислюються підвектори другого рядка таблиці екстраполяції, на останньому $k-1$ кроці обчислюються відповідні підвектори k-того рядка таблиці екстраполяції.

Другий алгоритм обчислення екстраполяційної таблиці полягає в наступному: у кожній групі процесорів рівно один процесор буде відповідати за передачу даних між групами. Для цього після підрахунку елементів першого стовпця таблиці в кожній групі процесорів проводиться обмін значеннями за типом "усі-усім", у результаті кожен процесор у групі буде містити весь вектор апроксимації розв'язку, а не його частину: процесори першої групи $\bar{T}_{11}$, другої – $\bar{T}_{21}$ та k-тої – $\bar{T}_{k1}$. Потім i-тий процесор кожної групи, крім останньої, передає всім процесорам сусідньої групи вектор розв`язків, щоб підрахувати екстрапольовані значення. Далі кожен процесор кожної групи крім першої (ширина алгоритму зменшується з кожним

кроком) обчислює свій підвектор (довжиною m_i) власного вектора екстрапольованих значень: процесори другої групи – $\bar{T}_{22}$, третьої – $\bar{T}_{32}$ та k-тої – $\bar{T}_{k2}$. На останньому $k-1$ кроці будуть працювати тільки процесори останньої групи, і кожен буде обчислювати m_k значень вектора $\bar{T}_{kk}$. Щоб збільшити збалансування завантаження, розіб'ємо процесорне поле на таку ж кількість груп, але не рівномірно, а пропорційно до чисел:

$$n_i, i=\overline{1,k} : p_1 : p_2 : \ldots : p_k = n_1 : n_2 : \ldots : n_k,$$

$$\sum_{i=1}^{k} p_i = p, \quad p_i = pn_i / N(k).$$

Алгоритм розбиття на групи наступний: якщо кількість процесорів p не можна поділити на $N(k)$ без остачі, то береться $p_i = \hat{E}pn_i / N(k)\acute{U}$, а потім процесори, що залишилися по одному додаються до кожної з груп. Таким чином, кожна група буде містити тим більшу кількість процесорів, чим більше кроків інтегрування на базовому кроці їй належить виконати для отримання апроксимації розв'язку.

Скоротити час простою можна при використанні комбінаційного способу. Ідея комбінаційного методу полягає у використанні $\lceil k/2 \rceil$ груп процесорів, причому i-та група обчислює i-ту та $(k-i+1)$-ту апроксимації розвязку. Для отримання i-тої апроксимації необхідно n_i раз виконати звернення до опорного методу вирішення, для $(k-i+1)$-тої – n_{k-i+1} раз. Тобто перша група повинна n_1+n_k раз звернутися до обчислення рішення щодо опорного методу, друга група – n_2+n_{k-1} та, нарешті, $k1=\lceil k/2 \rceil$ група – $n_{k1}+n_{k1+1}$.

Очевидно, що існує практично рівномірне забезпечення завантаження процесорних груп, окрім першої, яка відрізняється від інших незалежно від довжини таблиці на одиницю. У випадку непарної довжини таблиці розбиття стає менш рівномірним за рахунок непарної останньої групи. Дослідження динамічних характеристик паралелізму отриманих обчислювальних схем дозволяє затверджувати, що перша обчислювальна схема незалежно від складності правої частини володіє кращими характеристиками прискорення. Найгіршим потенційним прискоренням володіють варіанти схеми 2 незалежно від складності правої частини.

Перша схема, що використовує системний паралелізм, має переваги ще й тому, що може бути реалізована на паралельному комп'ютері будь-якого типу та добрі показники завантаження процесорів. Суттєвим параметром для цих схем є складність правої частини СЗДР, що забезпечує домінування обчислень над обмінами. Для нескладних правих частин обчислювальна схема, яка об'єднує паралелізм системи та екстраполяції, має явну перевагу, і цей ефект із зростанням числа процесорів збільшується.

Особливості застосування графічних процесорів при паралельній реалізації ЛЕР з використанням технології CUDA

Архітектура GPGPU принципово відрізняється від архітектури класичних паралельних обчислювальних систем. Кожен пристрій NVIDIA являє собою співпроцесор, що працює в зв'язці з центральним процесором. З апаратної точки зору пристрій має велику кількість потокових процесорів. З програмної точки зору, пристрій здатний виконувати безліч потоків, об'єднаних в один або кілька блоків. На відміну від класичних паралельних ВС, в яких створення і підтримка потоку вимагає значних накладних витрат, апаратний планувальник графічного процесора може ефективно працювати з сотнями і тисячами потоків. Тому стандартним підходом до вирішення завдань на GPU є виділення окремого потоку на кожен елемент даних. Блоки в свою чергу об'єднуються в сітку, яка може бути одно/двох/тривимірною. Апаратний планувальник потоків розподіляє блоки між пристроями, що об'єднують кілька потокових процесорів. Потоки всередині кожного блоку поділяються на варпи по 32 штуки і виконуються в кожен момент часу на одному мультипроцесорі за принципом SIMD [6-7].

Розглянемо реалізацію екстраполяційного методу інтегрування СЗДР на базі деякого опорного методу з використанням поліноміальної ЛЕР на графічних процесорах загального призначення. З опису архітектури графічних процесорів [6-8] слідкує, що вони мають перевагу перед центральним процесором в тому випадку, якщо завдання можна розбити на велику кількість паралельних підзадач. Таким чином, задачі невисокої розмірності (до сотень рівнянь) вирішувати на GPU, як правило, неефективно. Для систем високої розмірності можна виділити дві сфери застосування графічних процесорів: розпаралелювання обчислень правої частини системи, а також, для неявних методів, розпаралелювання рішення одержуваної системи алгебраїчних рівнянь. Інші джерела паралелізму при вирішенні систем ЗДР, як правило, не грають великої ролі при використанні GPU, тому що не здатні забезпечити ефективне розпаралелювання на сотні і тисячі потоків. Відповідно, для явного опорного методу основне прискорення для паралельної реалізації ЛЕР при інтегруванні великих СЗДР виходить від паралельної реалізації обчислення правої частини системи, відповідно

така програмна реалізація буде відрізнятися для кожної розв'язуваної системи. Проте можна виділити кілька загальних прийомів придатних для використання при вирішенні широкого класу задач.

Перше, що слід вказати - стратегію розпаралелювання, тобто розподіл даних по потоках. Графічний процесор є системою із загальною пам'яттю, але ієрархія пам'яті влаштована таким чином, що загальна пам'ять є досить повільним ресурсом, а більш швидка пам'ять доступна тільки на рівні блоку або одного потоку. Отже, для системи диференційних рівнянь розумним є горизонтальне розбиття даних, якщо кожен потік обчислює одне або більше рівнянь, але кожне рівняння обчислюється не більше ніж одним потоком.

В даній роботі пропонується загальна схема паралельних методів ЛЕР на основі явних чисельних схем:

1. Ініціалізація всіх початкових даних, включаючи необхідні для вирішення коефіцієнти методу, виконується на центральному процесорі.

2. Виклик ядра та завантаження даних в відеопам'ять.

3. Ядро виконує реалізацію опорного методу, включаючи знаходження проміжних коефіцієнтів $\bar{k}_{i,j}, i=\overline{1,s}, j=\overline{1,m}$ для одноточкових багатостадійних методів та покрокове обчислення значень $\bar{y}$.

Ієрархія потоків, що використана при виклику ядра, являє собою двовимірний масив блоків розмірністю $S^*\left(N/m\right)$, кожен з яких включає в себе T потоків. S – порядок методу, m– кількість рівнянь в системі, а T – кількість потоків, об'єднаних в один блок [7]. Величина T вибирається з міркувань максимальної завантаженості робочих процесорів і має бути кратним 32.

Обчислення матриці коефіцієнтів $\bar{k}_{i,j}, i=\overline{1,s}, j=\overline{1,m}$ відбувається паралельно на всіх потоках. Блоки *s*-групи розраховують значення одного стовпця матриці коефіцієнтів *k*. Усі значення початкових даних і коефіцієнтів зберігаються в глобальній пам'яті. Процесори по черзі звертаються до даних, копіюють її до пам'яті, що розподіляється та виконують відповідні обчислення, після чого поточне значення матриці коефіцієнтів *k* оновлюється і слідкує наступна ітерація. Цей процес повторюється *S* разів.

Алгоритм з використанням розподіляємої пам'яті реалізується наступними етапами:

1) завантажується в розподілену пам'ять m елементів вектора $\bar{y}_n$ (кожний потік блоку завантажує m/T елементів);

2) __syncthreads() (очікується інформація від інших тому, що потік використовує не тільки ті елементи, які він завантажив);

3) обчислюються коефіцієнти $\bar{k}_{i,j}^{(0)}, i=\overline{1,s}, j=\overline{1,m}$ (кожний потік знаходить m/T елементів);

4) завантажується в розподілену пам'ять рядок матриці відповідний номеру блоку $A=(a_{ij}), i,j=\overline{1,s}$ та T елементів матриці $\bar{k}_{i,j}^{(0)}, i=\overline{1,s}, j=\overline{1,m}$ (кожний потік блоку завантажує m/T елементів);

5) __syncthreads() (синхронізація);

6) обчислюється матриця на s-му кроці $\bar{k}_{i,j}^{(s)}, i=\overline{1,s}, j=\overline{1,m}$ (кожний потік блоку завантажує m/T елементів);

7) __syncthreads() ;

8) якщо обчислено не усі стадійні коефіцієнти $\bar{k}_{i,j}^{(s)}, i=\overline{1,s}, j=\overline{1,m}$ - повертаються на крок 4.

Після цього всі блоки першої групи обчислюють значення $\bar{y}_{n+1}$.

Отримані аналітичні оцінки та проведений експеримент свідчать про те, що динамічні характеристики розпаралелювання ЛЕР на базі CUDA при розв'язанні систем звичайних диференціальних рівнянь великої розмірності є високими. Використання великого числа потоків, що виконуються паралельно, дає високі показники навіть на невеликих розмірах правих частин і низьких порядках методу. На рисунках 3-4 наведено графіки залежності коефіцієнтів прискорення паралельних додатків реалізації ЛЕР для явних опорних методів різних порядків та різних розмірностей СЗДР.

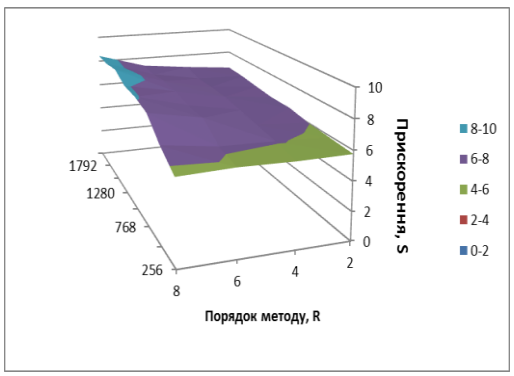

Рисунок 3 – Графік залежності коефіцієнтів прискорення від порядку методу і складності правих частин

Однак, варто відзначити, що в силу обмеженості розподіляємої пам'яті і кількості створюваних потоків на блоках, коефіцієнти

прискорення ростуть досить повільно, а при подальшому збільшенні кількості рівнянь у системі буде навіть знижуватися. Якщо порівнювати результати, отримані з використанням технологій MPI та CUDA, то очевидно, що застосування технології CUDA є більш ефективним.

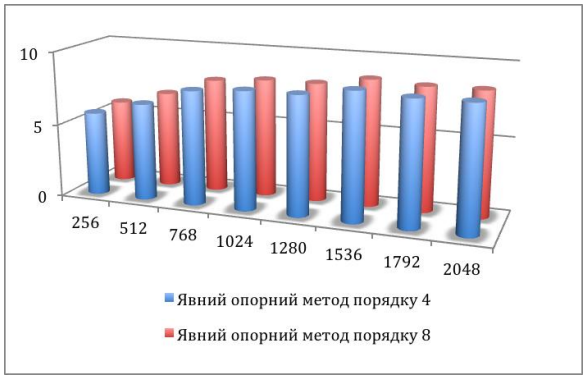

Рисунок 4 – Графік залежності коефіцієнтів прискорення від розміру СЗДР на базі явних опорних методів

Ефективність паралельних додатків технології локальної екстраполяції при використанні неявних опорних методів докладно викладено у [4-5]. Пріоритетними областями застосування таких чисельних схем є жорсткі задачі Коші. Із досліджених методів неявні багатоточкові або блокові методи дають найкращі результати в паралельній реалізації ЛЕР.

Таким чином, для ефективної алгоритмічної та програмної реалізації методів ЛЕР на графічних процесорах загального призначення існує ряд умов, які не залежать від конкретної СЗДР, але дозволяють досягти оптимізації обчислень. По-перше, це скорочення накладних витрат на звернення до GPU за рахунок зменшення кількості викликів ядер. По-друге, це застосування низькорівневих оптимізацій, як-то використання розподіляємої пам'яті та зменшення числа розгалужень.

Висновки

В статті розглянуто застосування екстраполяційних методів для розв'язання систем звичайних диференційних рівнянь з використанням технології CUDA. Розроблено паралельні алгоритми та їх програмні реалізації для ЛЕР-методів з використанням глобальної та розподіляємої пам'яті. Проведені експерименти свідчать про те, що використання технології CUDA є досить ефективним для вирішення СЗДР великої розмірності. Коефіцієнт прискорення, одержаний при реалізації паралельного алгоритму, збільшується з ростом порядку методу і розмірності системи. Таким чином, технологія CUDA з використанням розділяємої пам'яті є ефективним рішенням для інтегрування задач Коші для систем звичайних диференційних рівнянь великого розміру.

Наукова новизна отриманих результатів полягає в розробці методів підвищення ефективності розв'язання багатовимірних задач Коші для СЗДР за рахунок ефективного відображення на архітектуру графічних процесорів загального призначення на базі технології CUDA.

Практична значимість одержаних результатів заключається в розробці паралельних алгоритмів та їх програмних додатків, що ефективно відображаються на архітектуру графічних процесорів загального призначення.

Напрямом подальших досліджень являється аналіз існуючих та розробка нових методів скорочення накладних витрат за рахунок застосування, наприклад, ефективних алгоритмів редукції при використанні ЛЕР в сукупності з GPGPU.

Список літератури

1. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи / Э. Хайрер, С.Нёрсет, Г Ваннер. – М.: Мир, 1990. – 512с.
2. Паралельні однокрокові методи чисельного розв'язання задачі Коші: монографія / Л.П. Фельдман, І.А. Назарова. – Донецьк: ДВНЗ «ДонНТУ», 2011. – 185 с.
3. Фельдман Л.П. Параллельные алгоритмы экстраполяционных методов решения задачи Коши для компьютеров с распределенной памятью / Л.П. Фельдман, І.А. Назарова // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка, випуск 11 (164): – Донецьк, ДонНТУ, 2010. – С. 7-13.
4. Назарова И.А. Экстраполяционные блочные одношаговые численные методы решения жестких задач Коши / И.А. Назарова // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект», №3 , 2010. – Донецк: ИПИИ, 2010. – С.116-126.
5. Фельдман Л.П., Назарова И.А. Современные параллельные методы численного решения задачи Коши. – Донецк: ГВУЗ «ДонНТУ», 2013. – 206с.
6. Технология CUDA в примерах: введение в программирование графических процессоров / Дж. Сандерс, С. Кэндрот: Пер. с англ. Слинкина А.А. – М.: ДМК Пресс, 2011 – 232с.
7. Параллельные вычисления на GPU. Архитектура и программная модель CUDA / Боресков А.В., Харламов А.А., Марковский Н.Д. – М.: МГУ, 2012. – 336с.

8. CUDA C Programming Guide [Электронный ресурс]. – Режим доступа: http://docs.nvidia.com/cuda/cuda-c-programmingguide/#axzz3k3HvOyd7. – Загл. с экрана

Надійшла до редакції 15.10.2016

Л.П. ФЕЛЬДМАН, И.А. НАЗАРОВА, Я.О. ГРИЗАДУБОВА
Донецкий национальный технический университет (Украина)

ЭФФЕКТИВНОСТЬ ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ ЭКСТРАПОЛЯЦИОННЫХ МЕТОДОВ ЧИСЛЕННОГО РЕШЕНИЯ СОДУ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

Предложены методы повышения эффективности параллельного численного решения многомерной задачи Коши на базе технологии локальной экстраполяции. Получен последовательный алгоритм локальной экстраполяции, минимизирующий вычислительные затраты. Разработаны экстраполяционные параллельные алгоритмы на базе явных, полностью неявных и блочных опорных методов. Полученные экстраполяционные методы реализованы на графических процессорах с использованием технологии CUDA.

Ключевые слова: задача Коши, графические процессоры, параллельные экстраполяционные методы, технология CUDA.

L.P. FELDMAN, I.A. NAZAROVA, Y.O. GRIZADUBOVA
Donetsk National Technical University (Ukraine)

THE EFFICIENCY OF PARALLEL IMPLEMENTATION OF EXTRAPOLATION METHODS FOR THE NUMERICAL SOLUTION OF SODE ON GPUs

The methods of increasing the efficiency of the parallel numerical solution of multidimensional Cauchy's problem on the basis of extrapolation of local technology are proposed. Efficient serial algorithm of local extrapolation that minimizes the computational cost is obtained. Extrapolational parallel algorithms on the basis of explicit, implicit and block basic methods are developed. These extrapolation methods are implemented on GPUs with CUDA technology.

This article presents the results of development and research of efficiency of parallel algorithms for local extrapolation Richardson methods of numerical solution of large dimension SODE for Cauchy's problem. Schemes mapping of parallel methods on GPU's using CUDA technology within the concept of GPGPU (General-purpose graphics processing units) are demonstrated. Extrapolation methods for solving Cauchy's problem for ordinary differential equations possess a high degree of potential parallelism. The developed algorithms are implemented on parallel systems with global /shared memory of GPU. The estimations of the runtime and exchanges, total overhead of parallelism, speedup and efficiency of parallel solutions are defined.

Extrapolation technology of Richardson consists of basic numerical method for solving Cauchy's problem, sequence of grids integration, recursive rule for calculation values approximate solution. The effectiveness of its application depends on the correct choice and combination of all three components. Questions of designing effective to minimize the computational cost consistent method for integrating SODE based technology Richardson is described in detail in [1-4]. As general conclusions of the study: combination extrapolation based on symmetric basic methods and the pair sequences to generate grid integration are the most efficient in terms of computational cost. To reduce overhead in applying technology of local extrapolation should be chosen basic method of small order with sufficient stability properties. Potentially calculation on the extrapolation of the local technology contains three sources of internal parallelism: system parallelism (limited SODE dimension), parallelism extrapolation (limited by the size extrapolation table), concurrency of basic method (small degree of parallelism). The article examined the parallel algorithms for local extrapolation using various types of parallelism, describes the following load balancing methods for LER: uniform, proportional and combinational.

In the article the use of extrapolation methods for solving systems of ordinary differential equations using technology CUDA for GPGPU is considered. Parallel algorithms and their software implementation are developed for local Richardson extrapolation - methods using global and shared memory. The experiments indicate that using CUDA technology is a very effective solution for large-scale SODE. Coefficient of speed-up obtained in the implementation of parallel algorithm increases with the order of the method and system dimensions. There are a number of conditions that do not depend on the particular soda, but make it possible to achieve optimization calculations for efficient algorithmic and software implementation methods Lehr on GPU general purpose. Firstly, the reduction of overhead costs to appeal to the GPU by reducing the number of calls nuclei. Secondly, it is the use of low-level optimizations such as shared memory and reducing the number of branches. So, CUDA technology using shared storage is an effective solution for integrating the Cauchy problems for large systems of ordinary differential equations.

Key words: Cauchy's problem, GPUs, parallel extrapolation methods, technology CUDA.