ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

ВЕСТНИК ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

BULLETIN OF THE SOUTH URAL STATE UNIVERSITY

СЕРИЯ

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

2020, том 9, № 2

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2020, volume 9, no. 2

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2020
Т. 9, № 2

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2020
Vol. 9, no. 2

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents

УДК 004.021, 519.6 **DOI: 10.14529/cmse200201**

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ДЛЯ ПОЛУЧЕНИЯ РАВНОМЕРНОГО ПРИБЛИЖЕНИЯ РЕШЕНИЙ МНОЖЕСТВА ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ С НЕЛИНЕЙНЫМИ ОГРАНИЧЕНИЯМИ*

© 2020 В.В. Соврасов, К.А. Баркалов

Нижегородский государственный университет им. Н.И. Лобачевского
(603022 Нижний Новгород, пр. Гагарина, д. 23, к. 2)
E-mail: sovrasov.vladislav@itmm.unn.ru, konstantin.barkalov@itmm.unn.ru

Поступила в редакцию: 27.02.2020

В данной работе рассматривается построение параллельной версии алгоритма глобальной оптимизации, решающего одновременно множество задач с нелинейными ограничениями и получающего при этом равномерные оценки решений на этом множестве. Последнее свойство позволяет наиболее оптимально распределять вычислительные ресурсы, т.к. в процессе работы алгоритма погрешности численного решения во всех задачах убывают примерно с одинаковой скоростью. Алгоритм присваивает приоритет каждой задаче и на каждой итерации производит вычисления целевых функций в нескольких задачах параллельно. При окончании работы метода в произвольный момент времени во всех задачах из решаемой серии будут получены решения сходного качества. Серии из нескольких задач возникают, если задача глобальной оптимизации имеет дискретный параметр или, например, при решении задачи многокритериальной оптимизации методом свертки критериев. Рассматриваемый алгоритм использует отображения типа кривой Пеано для редукции многомерных задач оптимизации к одномерным. Эффективность реализованного алгоритма протестирована на наборах искусственно сгенерированных задач глобальной оптимизации, а также при решении серии задач, полученной в результате скаляризации задачи многокритериальной оптимизации. Также экспериментально подтверждена равномерная сходимость метода.

Ключевые слова: глобальная оптимизация, параллельные вычисления, алгоритмы прямой оптимизации, равномерная сходимость.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Соврасов В.В., Баркалов К.А. Параллельный алгоритм для получения равномерного приближения решений множества задач глобальной оптимизации с нелинейными ограничениями // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 5–18. DOI: 10.14529/cmse200201.

Введение

Нелинейная глобальная оптимизация невыпуклых функций традиционно считается одной из самых трудных задач математического программирования. Отыскание глобального минимума функции от нескольких переменных зачастую оказывается сложнее, чем локальная оптимизация в тысячемерном пространстве. Для последней может оказаться достаточно применения простейшего метода градиентного спуска, в то время как чтобы *гарантированно* отыскать глобальный оптимум методам оптимизации приходится накапливать информацию о поведении целевой функции во всей области поиска [6, 9, 10, 14]. Решение серии таких задач при ограниченных вычислительных ресурсах является еще более сложной задачей: помимо поиска глобального экстремума необходимо распределять вычислительные ресурсы так, чтобы сразу во всех решаемых задачах положение глобального экстремума

*Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2020».

было оценено примерно с одинаковым качеством. Обычно серию из q задач решают либо последовательно, либо параллельно порциями по $p \ll q$ задач, где p — количество параллельных вычислительных устройств. Такой подход ведет к тому, что в каждый момент времени до окончания вычислений остаются задачи, в которых оценка глобального оптимума не получена вообще, в то время, как в задачах из начала списка оптимум может быть оценен даже с избыточной точностью.

В данной работе рассматривается обобщение ранее разработанного в ННГУ им. Н.И. Лобачевского параллельного метода глобальной оптимизации для одновременного решения множества задач [3] на случай задач с нелинейными ограничениями. Для учета ограничений используется индексная схема [14], позволяющая работать с частично вычислимыми целевым функциями и обладающая экономичностью, сравнимой с другими подходами [1]. Эффективность реализованного алгоритма показана на примере решения множеств задач, сгенерированных специализированным механизмом, порождающим наборы задач заданной размерности с заданным количеством нелинейных ограничений [8]. Кроме искусственно сгенерированных задач, рассматриваемый метод протестирован также на множестве задач, возникающем при решении задачи многокритериальной оптимизации с нелинейными ограничениями методом свертки критериев [5].

Статья имеет следующую структуру. В разделе 1 рассмотрена постановка решаемой задачи, далее в разделе 2 приведено описание параллельного метода оптимизациии. Результаты численных экспериментов, подтверждающие эффективность рассматриваемого метода приводятся в разделе 3. В заключении приводится краткая сводка результатов, полученных в ходе работы, и указаны направления дальнейших усилий по улучшению программной реализации рассмотренного метода.

1. Постановка задачи глобальной оптимизации

В рамках данной работы будем рассматривать следующую постановку задачи глобальной оптимизации: найти глобальный минимум N-мерной функции $\varphi(y)$ в гиперинтервале $D = \{y \in \mathbf{R}^N : a_i \leqslant x_i \leqslant b_i,\ 1 \leqslant i \leqslant N\}$. Для построения оценки глобального минимума по конечному количеству вычислений значения функции требуется, чтобы скорость изменения $\varphi(y)$ в D была ограничена. В качестве такого ограничения как правило принимается условие Липшица.

$$\varphi(y^*) = \min\{\varphi(y) : y \in D\}, \tag{1}$$

$$|\varphi(y_1) - \varphi(y_2)| \leqslant L\|y_1 - y_2\|,\ y_1, y_2 \in D,\ 0 < L < \infty$$

Существуют различные методы, решающие рассмотренную многомерную задачу напрямую [9, 12], а также эффективные методы решения одномерных задач [14, 17]. В данной работе рассматривается одномерный метод, который применяется совместно со схемой редукции размерности. Классической схемой редукции размерности исходной задачи для алгоритмов глобальной оптимизации является использование разверток — кривых, заполняющих пространство [13].

$$\{y \in \mathbf{R}^N : -2^{-1} \leqslant y_i \leqslant 2^{-1},\ 1 \leqslant i \leqslant N\} = \{y(x) : 0 \leqslant x \leqslant 1\} \tag{2}$$

Отображение вида (2) позволяет свести задачу в многомерном пространстве к решению одномерной ценой ухудшения ее свойств. В частности, одномерная функция $\varphi(y(x))$

является не Липшицевой, а Гёльдеровой:

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \leqslant H|x_1 - x_2|^{\frac{1}{N}},\ x_1, x_2 \in [0;1],$$

где константа Гельдера H связана с константой Липшица L соотношением

$$H = 4Ld\sqrt{N},\ d = \max\{b_i - a_i : 1 \leqslant i \leqslant N\}.$$

Область D также может быть задана с помощью функциональных ограничений, что значительно усложняет задачу. Постановка задачи глобальной оптимизации в этом случае будет иметь следующий вид:

$$\varphi(y^*) = \min\{\varphi(y) : g_j(y) \leqslant 0,\ 1 \leqslant j \leqslant m\}. \tag{3}$$

Обозначим $g_{m+1}(y) = \varphi(y)$. Далее будем предполагать, что все функции $g_k(y), 1 \leqslant k \leqslant m+1$ удовлетворяют условию Липшица в некотором гиперинтервале, включающем D.

Далее будем интересоваться решением серии из q задач вида (3):

$$\min\{\varphi_1(y), y \in D_1\},\ \min\{\varphi_2(y), y \in D_2\}, ...,\ \min\{\varphi_q(y), y \in D_q\}. \tag{4}$$

2. Описание метода глобальной оптимизации

Принимая во внимание схему редукции размерности (2), будем при описании метода считать, что требуется найти глобальный минимум функции $\varphi(x),\ x \in [0;1]$, удовлетворяющей условию Гёльдера, при ограничениях $g_j(x)$, также удовлетворяющих этому условию на интервале $[0;1]$.

Рассматриваемый индексный алгоритм глобального поиска (ИАГП) для решения одномерной задачи (3) предполагает построение последовательности точек x_k, в которых вычисляются значения минимизируемой функции или ограничений $z_k = g_s(x_k)$. Для учета последних используется индексная схема [14]. Пусть $Q_0 = [0;1]$. Ограничение, имеющее номер j, выполняется во всех точках области

$$Q_j = \{x \in [0;1] : g_j(x) \leqslant 0\},$$

которая называется допустимой для этого ограничения. При этом допустимая область D исходной задачи определяется равенством: $D = \cap_{j=0}^{m} Q_j$. Испытание в точке $x \in [0;1]$ состоит в последовательном вычислении значений величин $g_1(x), ..., g_\nu(x)$, где значение индекса ν определяется условиями: $x \in Q_j,\ 0 \leqslant j < \nu,\ x \notin Q_\nu$. Выявление первого нарушенного ограничения прерывает испытание в точке x. В случае, когда точка x допустима, т. е. $x \in D$ испытание включает в себя вычисление всех функций задачи. При этом значение индекса принимается равным величине $\nu = m + 1$. Пара $\nu = \nu(x),\ z = g_\nu(x)$, где индекс ν лежит в границах $1 \leqslant \nu \leqslant m + 1$, называется результатом испытания в точке x.

Такой подход к проведению испытаний позволяет свести исходную задачу с функциональными ограничениями к безусловной задаче минимизации разрывной функции:

$$\psi(x^*) = \min_{x\in[0;1]} \psi(x),$$
$$\psi(x) = \begin{cases} g_\nu(x)/H_\nu & \nu < M \\ (g_M(x) - g_M^*)/H_M & \nu = M \end{cases}.$$

Здесь $M = \max\{\nu(x) : x \in [0;1]\}$, а $g_M^* = \min\left\{g_M(x) : x \in \cap_{i=0}^{M-1} Q_i\right\}$. В силу определения числа M, задача отыскания g_M^* всегда имеет решение, а если $M = m+1$, то $g_M^* = \varphi(x^*)$. Дуги функции $\psi(x)$ гельдеровы на множествах $\cap_{i=0}^{j} Q_i,\ 0 \leqslant j \leqslant M-1$ с константой 1, а сама $\psi(x)$ может иметь разрывы первого рода на границах этих множеств. Несмотря на то, что значения констант Гёльдера H_k и величина g_M^* заранее неизвестны, они могут быть оценены в процессе решения задачи.

Множество троек $\{(x_k, \nu_k, z_k)\},\ 1 \leqslant k \leqslant n$ составляет поисковую информацию, накопленную методом после проведения n шагов.

На первой итерации метода испытание проводится в произвольной внутренней точке x_1 интервала $[0;1]$. Индексы точек 0 и 1 считаются нулевыми, значения z в них не определены. Пусть выполнено $k \geqslant 1$ итераций метода, в процессе которых были проведены испытания в k точках $x_i,\ 1 \leqslant i \leqslant k$. Тогда точка x^{k+1} поисковых испытаний следующей $(k+1)$-ой итерации определяются в соответствии с правилами:

Шаг 1. Перенумеровать точки множества $X_k = \{x^1, \ldots, x^k\} \cup \{0\} \cup \{1\}$, которое включает в себя граничные точки интервала $[0;1]$, а также точки предшествующих испытаний, нижними индексами в порядке увеличения значений координаты, т.е.

$$0 = x_0 < x_1 < \ldots < x_{k+1} = 1$$

и сопоставить им значения $z_i = g_\nu(x_i),\ \nu = \nu(x_i),\ i = \overline{1,k}$.

Шаг 2. Для каждого целого числа $\nu,\ 1 \leqslant \nu \leqslant m+1$ определить соответствующее ему множество I_ν нижних индексов точек, в которых вычислялись значения функций $g_\nu(x)$:

$$I_\nu = \{i : \nu(x_i) = \nu,\ 1 \leqslant i \leqslant k\},\ 1 \leqslant \nu \leqslant m+1,$$

определить максимальное значение индекса $M = \max\{\nu(x_i),\ 1 \leqslant i \leqslant k\}$.

Шаг 3. Вычислить текущие оценки для неизвестных констант Гёльдера:

$$\mu_\nu = \max\left\{\frac{|g_\nu(x_i) - g_\nu(x_j)|}{(x_i - x_j)^{\frac{1}{N}}} : i, j \in I_\nu,\ i > j\right\}. \tag{5}$$

Если множество I_ν содержит менее двух элементов или если значение μ_ν оказывается равным нулю, то принять $\mu_\nu = 1$.

Шаг 4. Для всех непустых множеств $I_\nu,\ \nu = \overline{1,M}$ вычислить оценки

$$z_\nu^* = \begin{cases} \min\{g_\nu(x_i) : x_i \in I_\nu\} & \nu = M \\ -\varepsilon_\nu & \nu < M \end{cases},$$

где вектор с неотрицательными координатами $\varepsilon_R = (\varepsilon_1, .., \varepsilon_m)$ называется вектором резервов.

Шаг 5. Для каждого интервала $(x_{i-1}; x_i),\ 1 \leqslant i \leqslant k$ вычислить характеристику

$$R(i) = \begin{cases} \Delta_i + \frac{(z_i - z_{i-1})^2}{(r_\nu \mu_\nu)^2 \Delta_i} - 2\frac{z_i + z_{i-1} - 2z_\nu^*}{r_\nu \mu_\nu} & \nu = \nu(x_i) = \nu(x_{i-1}) \\ 2\Delta_i - 4\frac{z_{i-1} - z_\nu^*}{r_\nu \mu_\nu} & \nu = \nu(x_{i-1}) > \nu(x_i), \\ 2\Delta_i - 4\frac{z_i - z_\nu^*}{r_\nu \mu_\nu} & \nu = \nu(x_i) > \nu(x_{i-1}) \end{cases} \tag{6}$$

где $\Delta_i = (x_i - x_{i-1})^{\frac{1}{N}}$. Величины $r_\nu > 1,\ \nu = \overline{1,m}$ являются параметрами алгоритма. От них зависят произведения $r_\nu \mu_\nu$, используемые при вычислении характеристик в качестве оценок неизвестных констант Гёльдера.

Шаг 6. Выбрать наибольшую характеристику:

$$t = \arg\max_{1 \leqslant i \leqslant k+1} R(i). \tag{7}$$

Шаг 7. Провести очередное испытание в середине интервала $(x_{t-1}; x_t)$, если индексы его концевых точек не совпадают: $x^{k+1} = \frac{1}{2}(x_t + x_{t-1})$. В противном случае провести испытание в точке

$$x^{k+1} = \frac{1}{2}(x_t + x_{t-1}) - \operatorname{sgn}(z_t - z_{t-1}) \frac{|z_t - z_{t-1}|^n}{2r_\nu \mu_\nu^n},\ \nu = \nu(x_t) = \nu(x_{t-1}),$$

а затем увеличить k на 1.

Алгоритм прекращает работу, если выполняется условие $\Delta_t \leqslant \varepsilon$, где $\varepsilon > 0$ есть заданная точность. В качестве оценки глобально-оптимального решения выбираются значения

$$\varphi_k^* = \min_{1 \leqslant i \leqslant k} \varphi(x_i),\ x_k^* = \arg\min_{1 \leqslant i \leqslant k} \varphi(x_i). \tag{8}$$

Далее следуя подходу, описанному в [3], для решения серии задач (4) будем использовать q синхронно работающих копий ИАГП с тем лишь отличием, что на шаге 6 при выборе интервала с наилучшей характеристикой, выбор будет осуществляться из всех интервалов, которые породили на данный момент q копий ИАГП. Если наибольшая характеристика соответствует задаче i, то выполняется шаг 7 в копии метода с номером i, а остальные копии метода простаивают. Таким образом, на каждой итерации испытание проводится в задаче, наиболее перспективной с точки зрения характеристик (6), что позволяет динамически распределять ресурсы метода между задачами.

В [3] приведена теория сходимости такого подхода на случай решения задач без ограничений. При наличии ограничений характеристики интервалов, на концах которых нарушено разное количество ограничений, вычисляются в соответствии с нижними строчками из (6). Нетрудно заметить, что и в этом случае величины характеристик нормированы, а в случае точных оценок z_ν^* и μ_ν их масштаб не зависит от целевой функции и ограничений конкретной задачи. Таким образом, рассуждения из [3] можно провести и в случае использования индексной схемы.

Параллельная модификация метода не отличается от рассматриваемой в [3] и заключается в выборе p интервалов на шаге 6 и выполнения p испытаний параллельно на следующем шаге. При этом все ресурсы метода в рамках итерации могут быть направлены как на одну, так и на $l \leqslant p$ задач одновременно (в зависимости от того, какой из задач принадлежат выбранные методам интервалы).

3. Результаты численных экспериментов

Использование сгенерированных некоторыми случайными механизмами наборов тестовых задач с известными решениями является одним из общепринятых подходов к сравнению алгоритмов оптимизации [4]. В данной работе будем использовать два генератора тестовых задач, порождающих задачи различной природы [7, 16]. Эти генераторы порождают задачи без нелинейных ограничений, поэтому в дополнение к ним использована система GCGen[1] [8], позволяющая генерировать задачи с ограничениями на основе произвольных нелинейных функций.

[1]Исходный код системы доступен по ссылке https://github.com/UNN-ITMM-Software/GCGen

Вместе с системой GCGen распространяются примеры ее использования и построения наборов задач, каждая из которых состоит из целевой функции и двух ограничений, порожденных генератором F_{GR} [16] или GKLS [7].

Генератор GKLS [7] позволяет получать функции заданной размерности и с заданным количеством экстремумов. В сочетании с GCGen были порождены два множества по 100 задач размерности 2 и 3. Каждая из задач имеет два ограничения. Также с целью демонстрации того, что свойства метода сохраняются при существенно разных свойствах задач был сгенерирован смешанный класс, состоящий из 50 задач с двухмерными функциями GKLS и 50 задач с функциями F_{GR}. На рис. 1 и рис. 2 представлены примеры линий уровня рассматриваемых задач. Допустимая область закрашена.

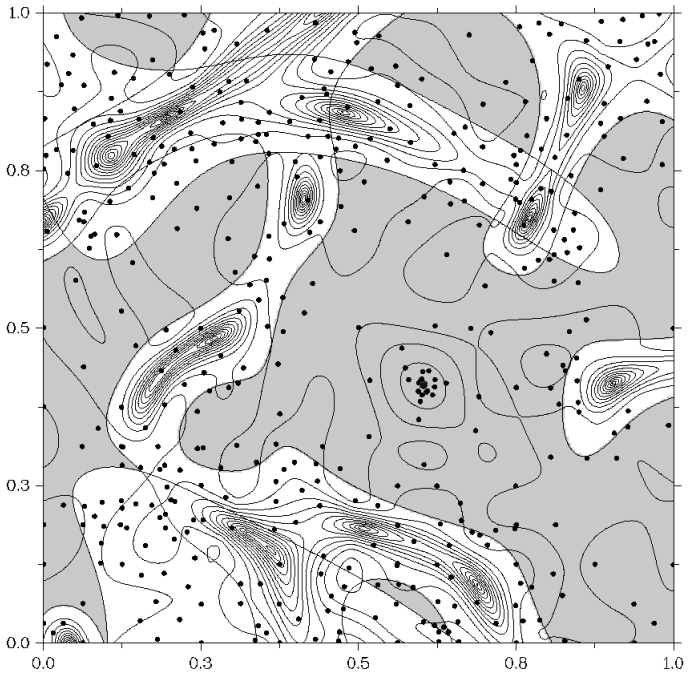

Рис. 1. Пример линий уровня задачи с решением внутри допустимой области

Будем считать, что тестовая задача решена, если метод оптимизации провел очередное испытание y^k в δ-окрестности глобального минимума y^*, т.е. $\left\|y^k - y^*\right\| \leqslant \delta = 0,01 \left\|b - a\right\|$, где a и b — левая и правая границы гиперкуба из (1). Если указанное соотношение не выполнено до истечения лимита на количество испытаний, то задача считается нерешенной.

При оценке качества метода и его реализации кроме ускорения от распараллеливания и времени выполнения также будем принимать во внимание среднее максимальное расстояния (в смысле l_{inf}-нормы) текущей оценки оптимума до его реального положения, вычисленное на множестве задач (4): D_{avg} и D_{max}. Динамика этих величин в процессе оптимизации показывает, насколько равномерно метод распределяет ресурсы между задачами.

Реализация параллельного метода была выполнена на языке C++ с использованием технологии OpenMP для распареллеливания процесса проведения испытаний на общей памяти. Все вычислительные эксперименты проведены на машине со следующей конфигурацией: Intel Core i7-7800X, 64GB RAM, Ubuntu 16.04 OS, GCC 5.5 compiler.

3.1. Результаты решения сгенерированных задач

Результаты решения тестовых задач последовательной и параллельной версией модифицированного ИАГП для решения множества задач представлены в табл. 1. Для всех

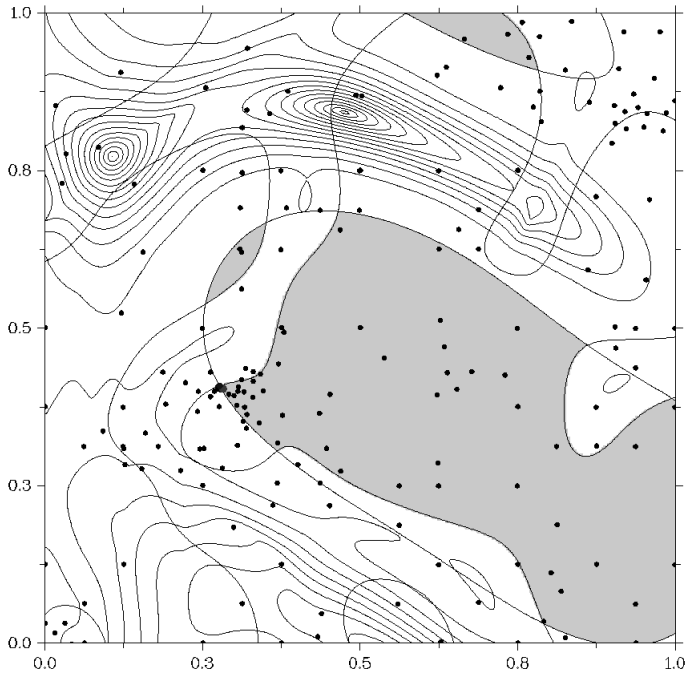

Рис. 2. Пример линий уровня задачи с решением на границе допустимой области

двухмерных классов задач параметр $r = 4,7$. В случае трехмерных задач $r = 4,7$, $\varepsilon_\nu = 0,1$. Во всех экспериментах в целевые функции и ограничения была внесена дополнительная вычислительная нагрузка так, чтобы время одного обращения к функции задачи было равно примерно 1 мс.

Из таблицы видно, что ускорение по итерациям S_i растет линейно с увеличением числа потоков p, в то время, как ускорение по времени S_p увеличивается не так быстро, что говорит о неидеальной реализации алгоритма. Увеличить реальное ускорение, верхней границей для которого является S_i, возможно путем оптимизации взаимодействий между копиями ИАГП и это планируется сделать в ходе будущей работы.

Таблица

Результаты экспериментов на наборах синтетических задач

Класс задач	p	Количество итераций	Время, с	S_i	S_t
GKLS & F_{GR} based	1	51434	90,20	-	-
	2	25698	56,96	2,00	1,58
	4	13015	36,67	3,95	2,46
	6	8332	26,85	6,17	3,36
GKLS based 2d	1	59066	97,53	-	-
	2	29060	60,56	2,04	1,61
	4	14266	38,92	4,14	2,51
	6	9436	29,53	6,26	3,30
GKLS based 3d	1	782544	1117,55	-	-
	2	397565	752,92	1,97	1,48
	4	208073	526,67	3,76	2,12
	6	142089	445,45	5,50	2,51

Для того, чтобы показать равномерную сходимость все тестовые задачи были также решены ИАГП в режиме решения отдельных задач. На рис. 3 и рис. 4 указаны графики величин средних и максимальных расстояний от реальных оптимумов до их текущих оценок при решении серии из задач, порожденных двумя разными генераторами, по отдельности (сплошная кривая) и совместно (пунктирная кривая). Не смотря на значительную разницу в структуре задач, модифицированный ИАГП гораздо быстрее уменьшает как максимальное, так и среднее отклонение оценок от оптимумов. Это говорит о наличии равномерной сходимости по всему множеству совместно решаемых задач. При этом в случае последовательного решения задач величина D_{max} имеет наибольшее значение вплоть до решения последней задачи.

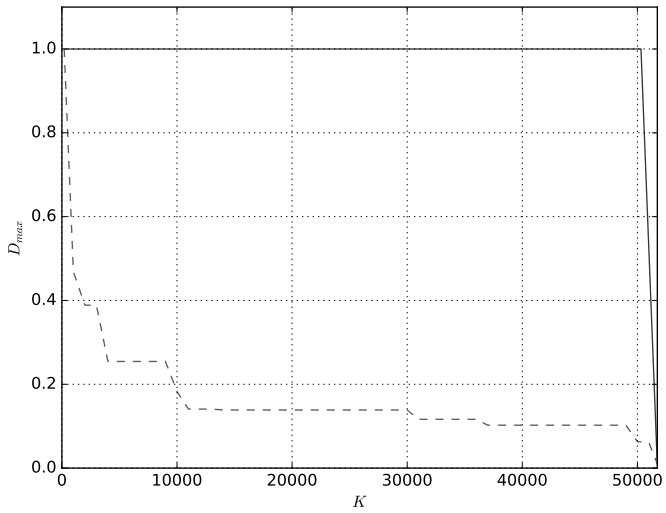

Рис. 3. Динамика величины D_{max} в процессе решения множества двухмерных задач, порождённых двумя разными генераторами GKLS и F_{GR}

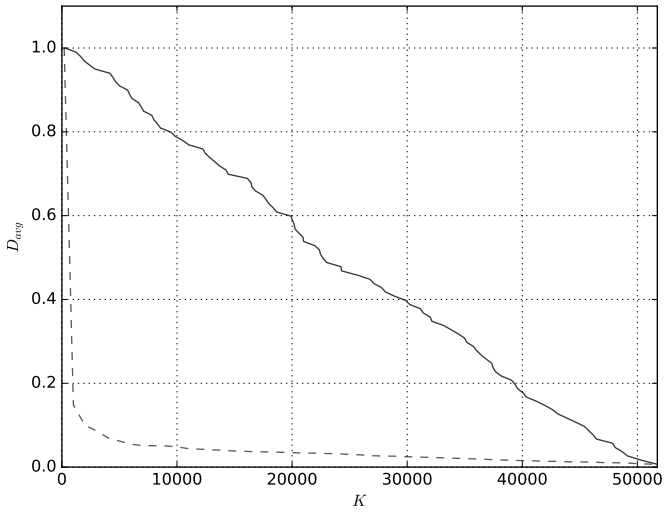

Рис. 4. Динамика величины D_{avg} в процессе решения множества двухмерных задач, порождённых двумя разными генераторами GKLS и F_{GR}

3.2. Пример решения многокритериальной задачи

Для демонстрации эффективности подхода к балансировке нагрузки рассмотрим пример, в котором множество задач вида (4) порождено в результате скаляризации многокритериальной задачи оптимизации с ограничениями.

Рассмотрим тестовую задачу, предложенную в [15]:

$$Minimize \begin{cases} f_1(y) = 4y_1^2 + 4y_2^2 \\ f_2(y) = (y_1 - 5)^2 + (y_2 - 5)^2 \end{cases} y_1 \in [-1; 2], y_2 \in [-2; 1]$$
$$s.t. \quad (9)$$
$$\begin{cases} g_1(y) = (y_1 - 5)^2 + y_2^2 - 25 \leqslant 0 \\ g_2(y) = -(y_1 - 8)^2 - (y_2 + 3)^2 + 7,7 \leqslant 0 \end{cases}$$

Будем использовать свертку Гермейера для скаляризации задачи (9). После свертки скалярная целевая функция имеет вид:

$$\varphi(y, \lambda_1, \lambda_2) = \max\{\lambda_1 f_1(y), \lambda_2 f_2(y)\}, \quad (10)$$

где $\lambda_1, \lambda_2 \in [0, 1]$, $\lambda_1 + \lambda_2 = 1$. Перебирая все возможные коэффициенты свертки, можно найти все множество парето-оптимальных решений в задаче (9). Для численного построения множества Парето выберем 100 наборов коэффициентов (λ_1, λ_2) таких, что $\lambda_1^i = ih$, $\lambda_2^i = 1 - \lambda_1^i$, $h = 10^{-2}$, $i = \overline{1, 100}$.

В качестве ограничения на вычислительные ресурсы был выбран лимит в 2500 испытаний. Множество вспомогательных скалярных задач решалось двумя способами:

- каждая задача решается отдельно с помощью ИАГП с установленным лимитом в 25 испытаний. Таким образом, вычислительные ресурсы равномерно распределены между задачами;
- все задачи решаются одновременно с помощью обобщенного ИАГП с установленным лимитом в 2500 испытаний.

В обоих случаях параметр $r = 4$.

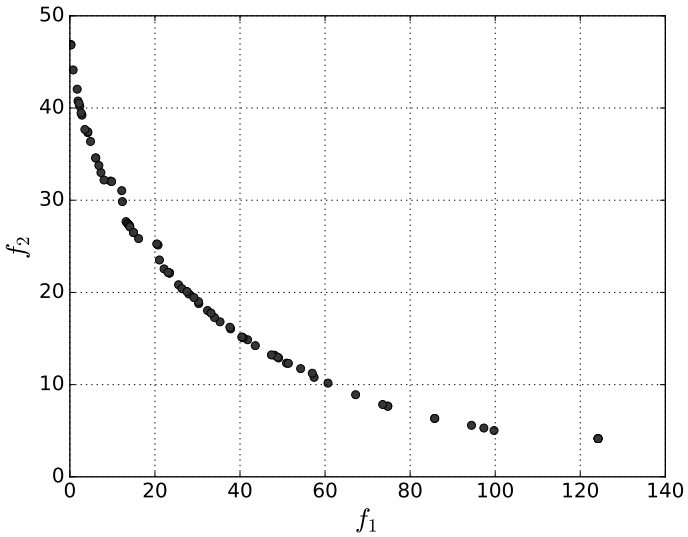

Рис. 5. Численная оценки множества Парето в задаче (9), полученная при раздельном решении задач

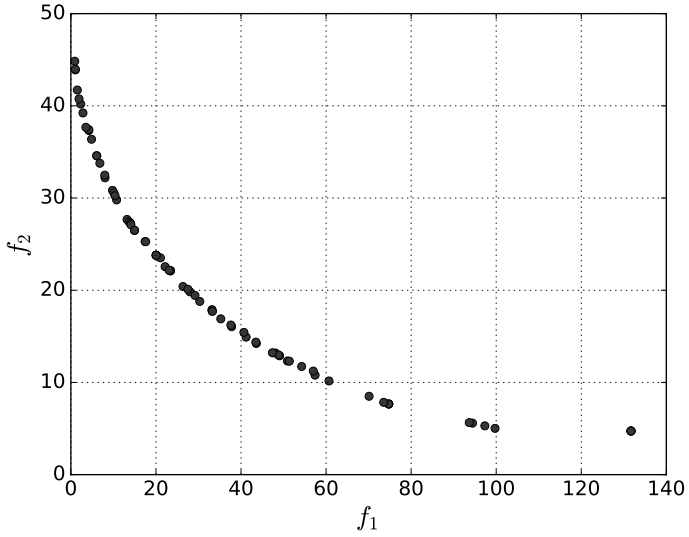

Рис. 6. Численная оценки множества Парето в задаче (9), полученная при совместном решении задач

На рис. 5 и рис. 6 представлены графики решений, полученных каждым из методов. Все графики качественно совпадают с указанным в [15] (авторы не предоставили другой информации и решениях для сравнения). Видно, что на рис. 5 кривая Парето имеет вогнутости, что не соответствует решению, указанному в [15] и означает нехватку ресурсов для решения некоторых из вспомогательных задач. Для оценки качества решения также был вычислен показатель $Spacing(SP)$ [11], характеризующий плотность точек аппроксимации множества Парето.

$$SP(S) = \sqrt{\frac{1}{|S|-1}\sum_{i=1}^{|S|}(\overline{d}-d_i)},\ \overline{d} = mean\{d_i\},\ d_i = \min_{s_i,s_j \in S: s_i \neq s_j} ||F(s_i) - F(s_j)||_1,\ F = (f_1, f_2)$$

В случае отдельного решения задач $SP_{single} = 0,984$, а при решении задач методом с балансировкой нагрузки $SP_{multi} = 0,749$, что говорит о более качественном приближении решения.

Заключение

В ходе работы была реализована поддержка нелинейных ограничений в алгоритме, решающем множество задач глобальной оптимизации. Проведены численные эксперименты, демонстрирующие преимущество рассматриваемого подхода над решением задач по отдельности. Показана эффективность совместного решения множества задач на примере решения многокритериальной задачи с нелинейными ограничениями.

В ходе дальнейшей работы планируется улучшить текущую реализацию алгоритма, сократив расходы на содержание поисковой информации для множества задач и тем самым улучшив показатели параллельного ускорения по времени. Также планируется реализовать версию рассматриваемого алгоритма, работающего на распределенной памяти по схеме, описанной в [2].

Исследование выполнено при поддержке РНФ, проект № 16-11-10150.

Литература

1. Barkalov K., Lebedev I. Comparing two approaches for solving constrained global optimization problems // Learning and Intelligent Optimization (Nizhny Novgorod, Russia, June, 19–21, 2017). Springer International Publishing, Cham. 2017. P. 301–306. DOI: 10.1007/978-3-319-69404-7_22.
2. Barkalov K., Lebedev I. Parallel algorithm for solving constrained global optimization problems // Parallel Computing Technologies (Nizhni Novgorod, Russia, Sept., 4–8, 2017). Springer International Publishing, Cham. 2017. P. 396–404. DOI: 10.1007/978-3-319-62932-2_38.
3. Barkalov K., Strongin R. Solving a set of global optimization problems by the parallel technique with uniform convergence // Journal of Global Optimization. 2018. Vol. 71, no. 1. P. 21–36. DOI: 10.1007/s10898-017-0555-4.
4. Beiranvand V., Hare W., Lucet Y. Best practices for comparing optimization algorithms // Optimization and Engineering. 2017. Vol. 18, no. 4. P. 815–848. DOI: 10.1007/s11081-017-9366-1.
5. Ehrgott M. Multicriteria Optimization. Springer-Verlag, Berlin, Heidelberg, 2005. 323 p. DOI: 10.1007/3-540-27659-9.
6. Evtushenko Y., Posypkin M. A deterministic approach to global box-constrained optimization // Optimization Letters. 2013. Vol. 7. P. 819–829. DOI: 10.1007/s11590-012-0452-1.
7. Gaviano M., Kvasov D.E., Lera D., Sergeev Ya.D. Software for generation of classes of test functions with known local and global minima for global optimization // ACM Transactions on Mathematical Software. 2003. Vol. 29, no. 4. P. 469–480. DOI: 10.1145/962437.962444.
8. Gergel V., Barkalov K., Lebedev I., Rachinskaya M., Sysoyev A. A flexible generator of constrained global optimization test problems // AIP Conference Proceedings. 2019. Vol. 2070, no. 1. P. 020009. DOI: 10.1063/1.5089976.
9. Jones D.R. The direct global optimization algorithm // The Encyclopedia of Optimization. Springer, Heidelberg, 2009. P. 725–735. DOI: 10.1007/978-0-387-74759-0_128.
10. Paulavivcius R., Zilinskas J., Grothey A. Parallel branch and bound for global optimization with combination of Lipschitz bounds // Optimization Methods and Software. 1997. Vol. 26, no. 3. P. 487–498. DOI: 10.1080/10556788.2010.551537.
11. Riquelme N., Von Lucken C., Baran B. Performance metrics in multi-objective optimization // 2015 Latin American Computing Conference (Arequipa, Peru, Oct., 19–23, 2015). IEEE. 2015. P. 1–11. DOI: 10.1109/clei.2015.7360024.
12. Sergeyev Y., Kvasov D. Deterministic Global Optimization. Springer, New York, 2017. 136 p. DOI: 10.1007/978-1-4939-7199-2.
13. Sergeyev Y.D., Strongin R.G., Lera D. Introduction to global optimization exploiting space-filling curves. Springer Briefs in Optimization, Springer, New York, 2013. 125 p. DOI: 10.1007/978-1-4614-8042-6.
14. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht, 2000. 704 p. DOI: 10.1007/978-1-4615-4677-1.

15. To T.B., Korn B. MOBES: A multiobjective evolution strategy for constrained optimization problems // Proceedings of the 3rd international conference on genetic algorithms (Mendel 97). 1997. P. 176–182.

16. Гришагин В.А. Операционные характеристики некоторых алгоритмов глобального поиска // Проблемы статистической оптимизации. 1978. № 7. С. 198–206.

17. Норкин В.И. О методе Пиявского для решения общей задачи глобальной оптимизации // Журнал вычислительной математики и математической физики. 1992. № 32. С. 992–1006.

Соврасов Владислав Валерьевич, аспирант, кафедра математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)

Баркалов Константин Александрович, к.ф.-м.н., доцент, кафедра математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)

DOI: 10.14529/cmse200201

PARALLEL GLOBAL OPTIMIZATION ALGORITHM FOR OBTAINING UNIFORM CONVERGENCE WHEN SIMULTANEOUSLY SOLVING A SET OF GLOBAL OPTIMIZATION PROBLEMS

© 2020 V.V. Sovrasov, K.A. Barkalov

Lobachevsky State University of Nizhni Novgorod
(pr. Gagarina 23(2), Nizhni Novgorod, 603022 Russia)
E-mail: sovrasov.vladislav@itmm.unn.ru, konstantin.barkalov@itmm.unn.ru

Received: 27.02.2020

In this work building of a parallel version of a method simultaneously solving a set of constrained global optimization problems is considered. This method converges uniformly to solutions of all the problems. That allows the method to arrange computational resources in an optimal way, since uniform convergence guarantees approximately equal precision of numerical solutions at the whole set of problems at the each iteration of optimization. The algorithm assigns a priority to each problem, and then at the each iteration carries out calculation of objective functions and constraints in several problems in parallel. If the method stops at any arbitrary moment, in all the problems numerical sulutions with the similar accuracy will be obtained. Sets of similar global optimization problems appear for an instance after scalarization of multi-objective problems or when a global optimization problem has a discrete parameter which takes a finite number of possible values. The considered method uses Peano-type curves to transform multidimensional problems into univariate ones. Efficiency of the implemented parallel algorithm is evaluated on several sets of synthetically generated constrained global optimization problems and on a scalarized multi-objective problem. Also the uniform convergence was confirmed numerically by validation quality of intermediate solutions during the optimization process.

Keywords: global optimization, parallel computations, derivative-free optimization, uniform convergence.

FOR CITATION

Sovrasov V.V., Barkalov K.A. Parallel Global Optimization Algorithm for Obtaining Uniform Convergence When Simultaneously Solving a Set of Global Optimization Problems. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 2. P. 5–18. (in Russian) DOI: 10.14529/cmse200201.

References

1. Barkalov K., Lebedev I. Comparing two approaches for solving constrained global optimization problems. Learning and Intelligent Optimization (Nizhny Novgorod, Russia, June, 19–21, 2017). Springer International Publishing, Cham. 2017. P. 301–306. DOI: 10.1007/978-3-319-69404-7_22.

2. Barkalov K., Lebedev I. Parallel algorithm for solving constrained global optimization problems. Parallel Computing Technologies (Nizhni Novgorod, Russia, Sept., 4–8, 2017). Springer International Publishing, Cham. 2017. P. 396–404. DOI: 10.1007/978-3-319-62932-2_38.

3. Barkalov K., Strongin R. Solving a set of global optimization problems by the parallel

technique with uniform convergence. Journal of Global Optimization. 2018. Vol. 71, no. 1. P. 21–36. DOI: 10.1007/s10898-017-0555-4.

4. Beiranvand V., Hare W., Lucet Y. Best practices for comparing optimization algorithms. Optimization and Engineering. 2017. Vol. 18, no. 4. P. 815–848. DOI: 10.1007/s11081-017-9366-1.
5. Ehrgott M. Multicriteria Optimization. Springer-Verlag, Berlin, Heidelberg, 2005. 323 p. DOI: 10.1007/3-540-27659-9.
6. Evtushenko Y., Posypkin M. A deterministic approach to global box-constrained optimization. Optimization Letters. 2013. Vol. 7. P. 819–829. DOI: 10.1007/s11590-012-0452-1.
7. Gaviano M., Kvasov D.E., Lera D., Sergeev Ya.D. Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software. 2003. Vol. 29, no. 4. P. 469–480. DOI: 10.1145/962437.962444.
8. Gergel V., Barkalov K., Lebedev I., Rachinskaya M., Sysoyev A. A flexible generator of constrained global optimization test problems. AIP Conference Proceedings. 2019. Vol. 2070, no. 1. P. 020009. DOI: 10.1063/1.5089976.
9. Jones D.R. The direct global optimization algorithm. The Encyclopedia of Optimization. Springer, Heidelberg. 2009. P. 725–735. DOI: 10.1007/978-0-387-74759-0_128.
10. Paulavivcius R., Zilinskas J., Grothey A. Parallel branch and bound for global optimization with combination of Lipschitz bounds. Optimization Methods and Software. 1997. Vol. 26, no. 3. P. 487–498. DOI: 10.1080/10556788.2010.551537.
11. Riquelme N., Von Lucken C., Baran B. Performance metrics in multi-objective optimization. 2015 Latin American Computing Conference (Arequipa, Peru, Oct., 19–23, 2015). IEEE. 2015. P. 1–11. DOI: 10.1109/clei.2015.7360024.
12. Sergeyev Y., Kvasov D. Deterministic Global Optimization. Springer, New York, 2017. 136 p. DOI: 10.1007/978-1-4939-7199-2.
13. Sergeyev Y.D., Strongin R.G., Lera D. Introduction to global optimization exploiting space-filling curves. Springer Briefs in Optimization, Springer, New York, 2013. 125 p. DOI: 10.1007/978-1-4614-8042-6.
14. Strongin R.G., Sergeyev Ya.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht, 2000. 704 p. DOI: 10.1007/978-1-4615-4677-1.
15. To T.B., Korn B. MOBES: A multiobjective evolution strategy for constrained optimization problems. Proceedings of the 3rd international conference on genetic algorithms (Mendel 97). 1997. P. 176–182.
16. Grishagin V. Operating characteristics of some global search algorithms. Problems of Stochastic Search. 1978. no. 7. P. 198–206. (in Russian)
17. Norkin V. I. Towards Pijavskyj's method for solving common global optimization problem. Computational Mathematics and Mathematical Physics. 1992. Vol. 32, no. 7. P. 992–1006. (in Russian)

УДК 004.021, 004.023 **DOI: 10.14529/cmse200202**

МЕТОДЫ ОПТИМИЗАЦИИ ОБОБЩЕННЫХ ТЕНЗОРНЫХ СВЕРТОК

© 2020 Р.А. Гареев

Уральский федеральный университет им. Б.Н. Ельцина
(620002 Екатеринбург, ул. Мира, д. 19)
E-mail: gareevroman@gmail.com
Поступила в редакцию: 18.03.2020

Свертка тензоров является одной из основных операций «Тензорного исчисления» — отдельного раздела математики, ставшего основным языком для описания фундаментальных законов таких областей науки, как теория относительности, механика, электродинамика и физика твердого тела. Эффективность выполнения свертки тензоров и ее обобщений имеет существенную практическую значимость для таких областей, как решение задач математической физики, машинного обучения, в спектральных методах, в квантовой химии, при интеллектуальном анализе данных, в высокопроизводительных вычислениях на многопроцессорных системах и др. В последние двадцать лет количество методов оптимизации тензорных сверток значительно увеличилось и продолжает возрастать. В статье представлен обзор активно используемых подходов к оптимизации свертки тензоров, применяемых при решении прикладных задач на однопроцессорных и многопроцессорных вычислительных системах с распределенной памятью. В работе представлены методы оптимизации важных частных случаев свертки тензоров — матричного и матрично-векторного произведения, использующихся для большинства оптимизаций сверток тензоров. Описанные оптимизации могут применяться в процессе компиляции программ, выполняемой промышленными компиляторами. Представленная информация может помочь при систематизации уже имеющихся знаний.

Ключевые слова: свертка тензоров, линейная алгебра, высокопроизводительные вычисления.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Гареев Р.А. Методы оптимизации обобщенных тензорных сверток // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 19–39. DOI: 10.14529/cmse200202.

Введение

Развитие вычислений с тензорами привели к созданию «Тензорные исчисления», отдельного активно развивающегося раздела математики, использующегося почти во всех областях механики, теоретической физики и прикладной математики. Можно выделить восемь основных операций, выполняемых над тензорами: сложение и вычитание, умножение на число, умножение тензоров, свертка, перестановка индексов, симметрирование и альтернирование. Каждая из перечисленных операций имеет широкое применение в различных научных дисциплинах [1].

Операция свертки тензоров (tensor contraction, TC) применяется в алгоритмах машинного обучения, таких как обучение и логический вывод в глубинных нейронных сетях (Deep Neural Networks, DNN) [2]. Квантовая химия широко использует TC в методах Хартри—Фока, а также при решении других задач, имеющих существенную практическую знчимость [3]. TC применяется при моделировании потоков несжимаемой жидкости на основе спектральных методов [4]. TC может быть использована для вычисления многомерного преобразования Фурье. В частности, трехмерное дискретное преобразование Фурье может быть вычислено через TC тензоров, полученных в результате применения двумерных дискретных преобразований Фурье [3].

Умножение матриц (matrix-matrix multiplication, MMM) и умножение матрицы на вектор (matrix-vector multiplication, MVM) могут рассматриваться как частные случаи ТС, широко применяющиеся при решении прикладных задач. Примером использования MMM служит шифрование баз данных с последующим интеллектуальным анализом данных (Data Mining) [5]. MMM вычисляется во время обрезки весов (weight pruning), использующимся в машинном обучении [6]. Энергия корреляции молекул, рассматриваемая в квантовой химии, рассчитывается с использованием MMM [7]. MMM служит основой для большинства операций с матрицами, изучаемых в высокопроизводительных вычислениях [9]. В качестве примеров применения MVM можно привести обрезку весов, используемую в машинном обучении [10]; анализ графов [11]; расчет коэффициентов скорости реакции [12]. Большая часть операций с векторами реализована с использованием MVM [13–15]. MVM применяется при решении задач математической физики численными методами, в частности, при решении задач математической геофизики. Например, MVM успешно используется при решении задач гравиметрии или магнитометрии о нахождении поверхностей раздела сред по гравитационным или магнитным данным методом Левенберга—Марквардта [8].

ТС может быть обобщена на произвольные полукольца матриц для решения задач о путях [16] (нахождение кратчайших путей, построение минимального остовного дерева, задача о самом широком пути и др.). Необходимость в решении таких задач возникает, в частности, в математических методах исследования операций, робототехнике, планировании размещения предприятий, транспортировке товаров, проектировании сверхбольших интегральных схем (VLSI design), при нахождении путей в картографических сервисах (MapQuest, Google Maps и др.).

В данной статье выполнен обзор методов оптимизации времени выполнения ТС. В разделе 1 дана классификация подходов к оптимизации MMM и MVM, использующимся в качестве основы для оптимизаций ТС. В разделе 2 описываются методы оптимизации ТС. В заключении приводится обобщение результатов, полученных в ходе исследования.

1. Оптимизация MMM и MVM

В силу широкой распространенности и практической значимости оптимизация MMM и MVM является отдельным предметом изучения высокопроизводительных вычислений [9]. Оптимизированные MMM и MVM могут применяться для оптимизации ТС для тензоров большей размерности (Раздел 2). В данном разделе рассматриваются основные подходы к оптимизации MMM и MVM.

Специалисты в области приложений линейной алгебры одними из первых предложили использовать единый интерфейс к высокопроизводительным фундаментальным операциям, реализованным как стороннее программное обеспечение [17]. Более сорока лет таким интерфейсом являются базовые подпрограммы линейной алгебры (Basic Linear Algebra Subprograms, BLAS) [13–15]. Описываемые подпрограммы разделяются на три уровня. Первый уровень описывает операции над векторами, такие как, например, скалярное произведение и векторные нормы. На втором уровне интерфейса BLAS содержится описание операций над матрицами и векторами, таких как матрично-векторное произведение, а также описание подпрограмм для нахождения решения системы алгебраических уравнений с треугольной матрицей. Третьим уровнем описывается матричное произведение для различных типов матриц, построение симметричной матрицы и другие операции над матрицами. Интерфейс BLAS продолжает развиваться. BLAS++ [18], обновленная версия интерфейса

BLAS, используется в проектах, направленных на достижение экзафлопсной производительности.

Применение BLAS требует новой реализации фундаментальных операций для каждой новой архитектуры процессора. В большинстве случаев они создаются экспертами в области высокопроизводительных вычислений, что требует значительных временных затрат. Реализации фундаментальных операций могут распространяться как открытое программное обеспечение (GotoBLAS [9, 19], OpenBLAS [20], BLIS [21]) и как проприетарные библиотеки, выпускаемые производителями аппаратного обеспечения (Intel's MKL [22], IBM's ESSL [23], ArmPL [24]).

Для сокращения времени разработки высокопроизводительных реализаций BLAS авторы библиотек ATLAS [25] и PHiPAC [26] предложили использование автонастройки (autotuninig). В ходе ее работы выполняется перебор всех возможных значений параметров рассматриваемой программы с целью получения реализации, имеющей наименьшее время выполнения. Как и в случае ручной настройки (manual-tuning), для выполнения автонастройки необходим доступ к целевой архитектуре, а также потенциально большие временные затраты. Было показано, что в случае ATLAS и MMM автонастройка может приводить к неоптимальным результатам [9].

Алгоритм 1: Вычисление матричного произведения

```
1  begin
2      for j = 0...N step N_c do
3          for p = 0...K step K_c do
4              Упаковываем B(p:p + K_c − 1, j:j + N_c − 1) в массив B_c
5              for i = 0...M step M_c do
6                  Упаковываем A(i:i + M_c − 1, p:p + K_c − 1) в массив A_c
7                  for j_c = 0...N_c step N_r do
8                      for i_c = 0...M_c step M_r do
9                          for p_c = 0...K_c do
10                             𝕀 = i_c : i_c + M_r − 1
11                             𝕁 = j_c : j_c + N_r − 1
12                             C_c(𝕀, 𝕁) += A_c(𝕀, p_c) · B_c(p_c, 𝕁)
13                         end
14                     end
15                 end
16             end
17         end
18     end
19 end
```

В качестве примера рассмотрим параметризованный алгоритм 1, применяемый для выполнения матричного умножения в фреймворке BLIS [17]. Перемножаемые матрицы A и B имеют размерность $M \times K$ и $K \times N$, соответсвенно. Матрица C размерности $M \times N$ содержит результат выполнения матричного произведения. В процессе выполнения алгоритма матрицы A и B разбиваются на блоки (рис. 1). Это позволяет многократно использовать элементы блоков, хранящихся в кэш-памяти. Размеры блоков N_c, K_c, M_c, N_r и M_r являются

параметрами алгоритма. Их оптимальные значения могут определяться с помощью автонастройки и ручной настройки. С целью обеспечения последовательного доступа к операндам МММ используется упаковка данных. В процессе выполнения упаковки данных создаются временные выравненные в памяти массивы A_c и B_c, хранящие элементы перемножаемых матриц. Внутренний цикл вместе со своим содержимым называется микро-ядром МММ. Оно может генерироваться автоматически в процессе компиляции или реализовываться вручную в виде отдельной процедуры, написанной на языке ассемблера для целевой архитектуры [27, 28].

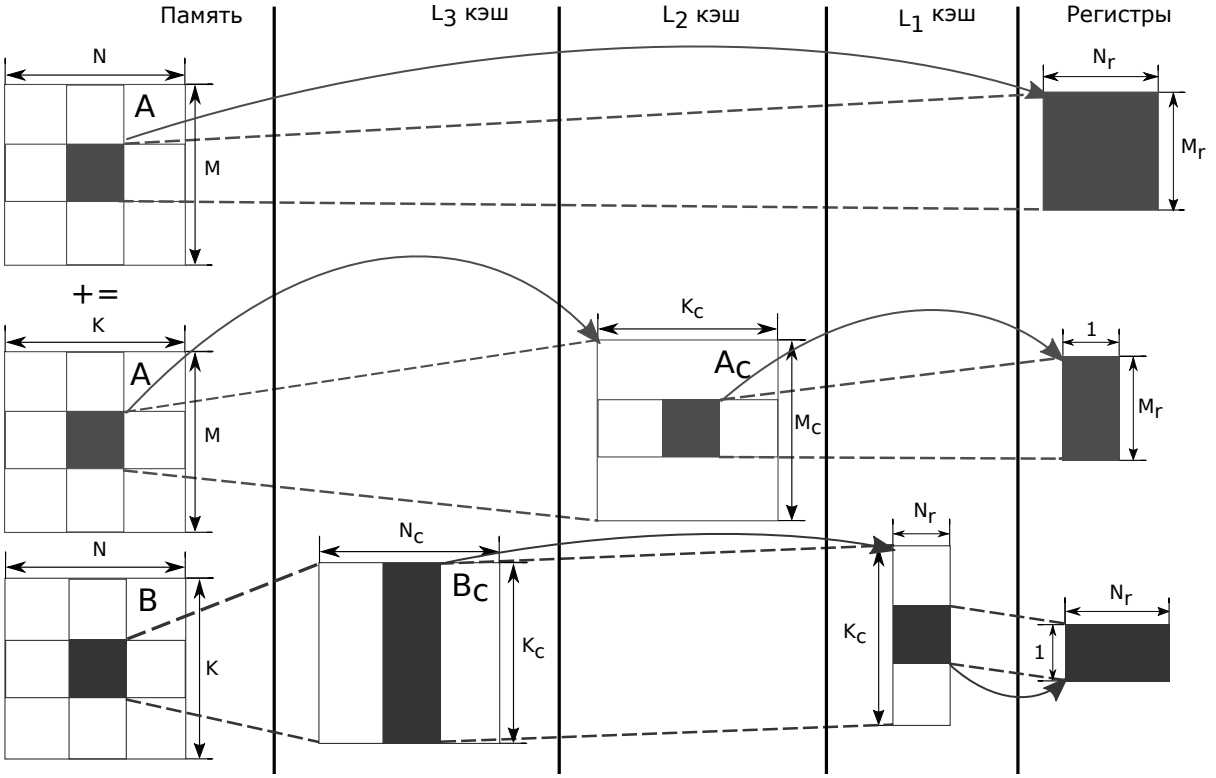

Рис. 1. Разбиение матриц A, B и C на блоки

Автонастройка и аналитические техники могут применяться совместно с целью определения наилучшей оптимизации. Такой подход используется в компиляторах LGen [29] и Build to Order BLAS (BTO) [30]. LGen — это компилятор для базовых операций линейной алгебры с операндами малой размерности, известной на этапе компиляции. Для выполнения разбиений циклов, векторизации и слияния циклов LGen использует специализированный язык, позволяющий построить математическое представление программы для ее дальнейшей оптимизации. Результатом работы LGen являются функции на языке C, содержащие SIMD инструкции. Пример выражения, подаваемого на вход компилятору LGen, приведен на рис. 2. BTO компилирует последовательность выражений, содержащих матрицы и векторы, написанную на языке MATLAB, в оптимизированный код на языке C++. В процессе оптимизации BTO выполняет слияние циклов, основываясь на аналитических моделях и эмпирических методах. На рис. 3 показан пример программы на языке MATLAB, компилируемой BTO.

Фреймворк AUGEM [31] использует автонастройку совместно с эвристическими алгоритмами векторизации, распределения регистров и планирования команд. С помощью библиотеки POET [32] фреймворк AUGEM распознает части абстрактного синтаксического

дерева программы на языке C, содержащие заранее определенный набор операций с матрицами, для их дальнейшей оптимизации. Пример операции с матрицами, распознаваемой фреймворком AUGEM показан на рис. 4.

```
alpha   =  Scalar()
beta    =  Scalar()
A       =  Matrix(6, 11)
x       =  Matrix(11, 1)
y       =  Matrix(6,  1)
z       =  Matrix(6,  1)
Generate(beta = (A*x+alpha*y)^T*z + Beta, opts)
```

Рис. 2. Пример выражения, подаваемого на вход компилятору LGen

```
GEMVER
in v1 : vector, v2 : vector,
   u1 : vector, u2 : vector,
   alpha : scalar, beta : scalar,
   x : vector, y : vector
inout A : dense column matrix
out z : vector, w : vector {
  A = A + v1 * u1' + v2 * u2'
  z = beta * (A' * x) + y
  w = alpha * (A * z)
}
```

Рис. 3. Пример выражения, подаваемого на вход компилятору BTO

```
mmCOMP(A, idx1, B, idx2, res):
tmp0=A[idx1]
tmp1=B[idx2]
tmp2=tmp0*tmp1
res=res+tmp2
```

Рис. 4. Пример операции с матрицами, распознаваемой фреймворком AUGEM

С целью сокращения времени автонастройки множество значений исследуемых параметров может быть уменьшено за счет использования дополнительных измерений, выполняемых на целевой архитектуре. Такой подход называется итеративной компиляцией (iterative compilation) [33]. В ходе ее работы алгоритм оценивает параметры целевой архитектуры, такие как, например, количество промахов кэша для данной реализации. В дальнейшем полученная информация применяется для определения значений оцениваемых параметров. Улучшенная автонастройка (auto-fine-tuning) [34] представляет собой отдельную модификацию автонастройки. Ее отличие в использовании аналитических моделей, уменьшающих пространство поиска. Такие модели могут быть созданы с применением знаний о современных архитектурах процессоров и оптимизуруемой операции. В общем случае указанные улучшения не гарантируют устранения проблем автонастройки.

Можно избежать перебора значений параметров программы за счет моделирования потребления ресурсов используемой архитектуры. Для этого применяются модели гипотети-

ческих процессоров, описывающих такие характеристики целевого процессора, как векторные регистры и векторные инструкции, иерархию кэш памяти [17, 34]. Аналитическое моделирование позволяет установить зависимости между потреблением ресурсов процессора и производительностью оптимизируемой программы. На момент написания данной работы не существует автоматического способа выполнения аналитического моделирования, позволяющего достичь высокой производительности оптимизируемой программы. Вследствие этого, как и в случае ручной настройки, аналитическое моделирование должно выполняться вручную специалистом.

Нахождение значений параметров программы, для которой выполнено аналитическое моделирование, и известны технические характеристики целевого процессора, выполняется за постоянное время и, вследствие этого, может использоваться в условиях ограниченного времени [28, 35]. Примером такой ситуации являются оптимизации, выполняемые промышленными компиляторами по умолчанию. С целью применения в процессе разработки и при компиляции готовых пакетов программ используемые оптимизации должны выполняться за наименьшее возможное время. Совместное применение автоматического распознавания кода, аналитических подходов и автоматической генерации кода позволяет автоматически получать высокопроизводительные реализации программ в процессе оптимизации, выполняемой компиляторами [28, 35].

В качестве примера модели гипотетического процессора рассмотрим модель, используемую в фреймворке BLIS для нахождения значений параметров реализации ММ [17]. Она может быть описана следующим образом [8]:

- «**Архитектура загрузки/сохранения и векторные регистры:** данные должны быть загружены в регистры процессора перед тем как с ними могут быть выполнены вычисления. Имеется N_{REG} векторных регистров. Каждый из них может содержать N_{VEC} значений размера S_{DATA}. Если N_{REG} равно 0, N_{VEC} присваивается значение 1. Предполагается, что команды для работы с памятью могут выполняться одновременно с командами арифметики с плавающей точкой».
- «**Векторные инструкции:** пропускная способность процессора составляет N_{VFMA} векторных операций смешанного сложения и умножения (vector fused multiply-add instruction, VFMA) за один такт. Отдельная VFMA выполняет N_{VEC} умножений и сложений, составляющих VFMA. Минимальное количество тактов L_{VFMA}, которое должно быть совершено перед началом выполнения новой зависимой по данным VFMA-инструкции, определяет задержку VFMA-инструкции».
- «**Кэш-память:** весь кэш данных — это множественно-ассоциативный кэш с политикой вытеснения последнего по времени использования (Least Recently Used, LRU). Каждый уровень кэша L_i характеризуется следующими параметрами: размер линии кэша C_{L_i}, степень ассоциативности W_{L_i}, количество множеств N_{L_i}, размер S_{L_i}, где $S_{L_i} = N_{L_i} C_{L_i} W_{L_i}$. В случае полностью ассоциативного кэша $N_{\mathrm{L}_i} = 1$».

Аналитическое моделирование может использоваться для минимизации перемещения данных и, как следствие, уменьшения времени работы приложения и его энергопотребления. Рассматриваемый алгоритм может быть изменен для достижения нижней асимптотической границы на перемещение данных в иерархии памяти, а также между процессорами [36, 37]. В частности могут изменяться формулы для нахождения значений параметров блочного алгоритма. Описанный подход обладает преимуществом в случае распределенных архитектур, имеющих большое количество отдельных узлов.

Кэш-независимые алгоритмы (cache-oblivious algorithms) [38] — это отдельный класс алгоритмов, позволяющих асимптотически оптимально использовать кэш-память, игнорируя ее отдельные параметры, такие как количество уровней кэша, длина кэш-линий, ассоциативность и т.д. Такие алгоритмы основываются на методе разделяй-и-властвуй (divide-and-conquer), разделяющем задачу на подзадачи до тех пор, пока данные отдельной подзадачи не смогут поместиться в каком-нибудь уровне кэш-памяти. Практическая оценка кэш-независимых алгоритмов показывает, что отсутствие в них таких механизмов, как предвыборка данных, затрудняет достижение производительности кода, созданного экспертами [39].

Несмотря на то, что MVM может рассматриваться как частный случай MMM, отношение количества вычислений, выполняемых MVM, к количеству перемещений данных из памяти меньше чем в случае MMM. Вследствие этого, методы для оптимизации MMM должны быть модифицированы для оптимизации MVM. В частности, из-за недопустимых издержек невозможно использование таких техник, как упаковка данных. В то же время возрастает необходимость в эффективной предвыборке данных, позволяющей уменьшить издержки чтения данных из памяти. Большинство подходов, используемых для оптимизации MVM, основаны на ручной настройке [9, 19–24, 40, 41] и автонастройке [25, 26]. Как и в случае MMM, аналитическое моделирование может применяться для оптимизации MVM с целью достижения производительности кода, оптимизированного вручную [8, 42]. Модификация представленной раннее гипотетической модели процессора может включать следующие дополнения, описывающее инструкции предвыборки [8]:

- «**Инструкции предвыборки:** за один такт процессора может быть выполнено N_{prefetch} инструкций. Задержка каждой инструкции составляет L_{prefetch} тактов процессора. Каждая инструкция может загрузить данные, размер которых равен C_{L_i}».

Особенности целевых архитектур процессоров и размерность рассматриваемой задачи могут использоваться совместно для автоматического получения оптимизированных реализаций BLAS. Так, например, библиотека LIBXSMM [43] генерирует низкоуровневые высокопроизводительные реализации MMM для некоторых архитектур от Intel и матриц, содержащих менее 80×80 элементов. Для этих целей используется специальный алгоритм предвыборки данных и модифицированный алгоритм векторизации.

Фреймворк POCA [27] позволят получить высокопроизводительные микро-ядра, части MMM, содержащие векторные инструкции. На рис. 5 представлен пример микро-ядра MMM для процессоров с микроархитектурой Sandy Bridge [27]. POCA использует эвристический алгоритм для распределения регистров процессора на основе анализа графа зависимостей и особенностей целевой архитектуры, таких как количество векторных регистров и количество тактов, требуемых для выполнения отдельных векторных инструкций микро-ядра.

2. Оптимизация ТС

За последние двадцать лет прогресс во многих научных дисциплинах, таких как квантовая химия и общая теория относительности, основывался на моделировании физических систем, использующем ТС. Несмотря на это, количество подходов к оптимизации ТС намного меньше чем количество подходов к оптимизации MMM и MVM [44]. В данном разделе описываются основные методы оптимизации времени выполнения ТС.

```
1 A0 = vload AP                  // <a0, a1, a2, a3>
2 B0 = vload BP                  // <b0, b1, b2, b3>
3 B1 = vshuffle B0 <1,0,3,2>     // <b1, b0, b3, b2>
4 B2 = vshuffle B1 <2,3,0,1>     // <b3, b2, b1, b0>
5 B3 = vshuffle B2 <1,0,3,2>     // <b2, b3, b0, b1>
6 M0 = fmul A0, B0               //
7 C0 = fadd C0, M0               // <c00, c11, c22, c33>
8 M1 = fmul A0, B1               //
9 C1 = fadd C1, M1               // <c01, c10, c23, c32>
10 M2 = fmul A0, B2              //
11 C2 = fadd C2, M2              // <c03, c12, c21, c30>
12 M3 = fmul A0, B3              //
13 C3 = fadd C3, M3              // <c02, c13, c20, c31>
14 AP = add AP, 4                // AP += 4
15 BP = add BP, 4                // BP += 4
16 prefetch_0 (BP+64)            // 8 * 64 = 512(bytes)
17 prefetch_0 AN                 // next A block
18 AN = add AN, 32               // Mr * 8 = 32(bytes)
```

Рис. 5. Пример микро-ядра MMM для процессора с микроархитектурой Sandy Bridge

Рассмотрим определения d-мерного тензора и TC, позволяющие свести оптимизацию этой операции к оптимизации MMM и MVM:

Определение 1. d-мерный тензор $\mathcal{T} \in \mathbb{R}^{n_{u_0} \times \ldots \times n_{u_{d-1}}}$ может быть определен как множество элементов из $\mathbb{R}$, описываемое следующим образом [45, 46]

$$\mathcal{T} \equiv \{A_{u_0 \ldots u_{d-1}} \in \mathbb{R} | (u_0, \ldots, u_{d-1}) \in n_{u_0} \times \ldots \times n_{u_{d-1}}\}.$$

В рамках данной работы будем предполагать, что элементы d-мерного тензора $\mathcal{T} \in \mathbb{R}^{n_{u_0} \times \ldots \times n_{u_{d-1}}}$ представлены многомерным массивом размерности $n_{u_0} \times \ldots \times n_{u_{d-1}}$. В общем случае это утверждение не всегда верно.

Определение 2. Пусть $\mathcal{A}$, $\mathcal{B}$, и $\mathcal{C}$ — это d_A-, d_B-, и d_C- мерные тензоры, соответственно. Сворачиваемые индексы $\mathcal{A}$ и $\mathcal{B}$ описываются кортежем $P = p_0 \ldots p_{t-1}$. Индексы $\mathcal{C}$, а также свободные индексы $\mathcal{A}$ и $\mathcal{B}$ описываются кортежами $I = i_0 \ldots i_{r-1}$ и $J = j_0 \ldots j_{s-1}$, соответственно. Операция свертывания или свертки $\mathcal{A}$ и $\mathcal{B}$ определяется как $\mathcal{C}_{\pi_C(IJ)} = \sum_P \alpha \cdot \mathcal{A}_{\pi_A(IP)} \cdot \mathcal{B}_{\pi_B(PJ)} + \beta \cdot \mathcal{C}_{\pi_C(IJ)}$, где $\sum_P = \sum_{p_0=0}^{n_{p_0}-1} \ldots \sum_{p_{t-1}=0}^{n_{p_{t-1}}-1}$, $\pi_C(IJ)$, $\pi_A(IP)$, и $\pi_B(PJ)$ — это перестановки индексов, α, $\beta \in \mathbb{R}$ [45, 46].

С целью оптимизации TC может быть использован программный код оптимизированных MMM и MVM, полученный в результате выполнения алгоритмов, описанных в предыдущем разделе. Так, например, метод Transpose-Transpose-GEMM-Transpose (TTGT) [47], применяемый в различных прикладных библиотеках (MATLAB Tensor Toolbox [48], NumPy [49], Eigen [50] и др.), выполняет перестановку индексов тензоров с целью их представления в виде матриц и использования MMM. Для выполнения таких перестановок требуются дополнительное время и память.

Метод Loops-over-GEMMS (LoG) [44] представляет сворачиваемые тензоры как наборы матриц, для которых выполняется MMM. Производительность LoG зависит от размера

таких матриц, а также расположения тензоров в памяти [44]. Так, например, в случае свертки 3-мерного тензора $\mathcal{A} \in \mathbb{R}^{m\times n\times k}$ и 2-мерного тензора $\mathcal{B} \in \mathbb{R}^{k\times l}$ в 3-мерный тензор $\mathcal{C} \in \mathbb{R}^{m\times n\times l}$, имеющей следующий вид $\mathcal{C}_{\alpha\beta\rho} = \sum\limits_{\delta=0}^{k-1} \mathcal{A}_{\alpha\beta\delta} \cdot \mathcal{B}_{\delta\rho} + \mathcal{C}_{\alpha\beta\rho}$, LoG вычислит МММ для каждого из n значений индекса β.

Для решения задач в таких научных областях, как, например, квантовая химия, созданы модификации метода TTGT для распределенных вычислений. Такие подходы применяют разбиение тензоров на блоки и другие техники для уменьшения передачи данных в иерархии памяти, а также между процессорами. Примерами библиотек, содержащих реализации описанных методов, служат Tensor Contraction Engine [47], Cyclops Tensor Framework [51], libtensor [52], TiledArray [53, 54] и др.

Для сокращения издержек, связанных с перестановкой индексов тензоров, можно использовать только микро-ядра, части реализации оптимизированного МММ, содержащие векторные инструкции на языке ассемблера. Такой подход применяется в методах GEMM-like Tensor-Tensor multiplication (GETT) [46] и Tensor-Based Library Instantiation Software (TBLIS) [45]. Как и в случае алгоритма 1 для вычисления МММ подходы GETT и TBLIS упаковывают элементы операндов ТС во временные одномерные массивы с целью дальнейшего использования кода микро-ядра МММ.

В отличии от GETT, TBLIS представляет тензоры в виде матриц, что позволяет явно выполнять разбиение матриц на блоки и другие трансформации, применяемые в алгоритме 1. В качестве примера применения такого представления рассмотрим d-мерный тензор $\mathcal{T} \in \mathbb{R}^{n_{u_0}\times\ldots\times n_{u_{d-1}}}$. $A_{u_0\ldots u_{d-1}}$, элемент тензора $\mathcal{T}$, расположен в памяти со смещением $\sum\limits_{k=0}^{d-1} u_k \prod\limits_{l=k+1}^{d-1} n_{u_l}$. Пусть свободные и сворачиваемые индексы тензора $\mathcal{T}$ описываются кортежами $I = i_0 \ldots i_{r-1}$ и $P = p_0 \ldots p_{s-1}$, соответсвенно. Тогда смещение может быть представлено в виде $\sum\limits_{k=0}^{d_I-1} i_k \left(\prod\limits_{l=k+1}^{d_I-1} n_{i_l}\right)\left(\prod\limits_{l=0}^{d_P-1} n_{p_l}\right) + \sum\limits_{k=0}^{d_P-1} p_k \prod\limits_{l=k+1}^{d_P-1} n_{p_l} = \left(\sum\limits_{k=0}^{d_I-1} i_k \prod\limits_{l=k+1}^{d_I-1} n_{i_l}\right) n_P + \sum\limits_{k=0}^{d_P-1} p_k \prod\limits_{l=k+1}^{d_P-1} n_{p_l} = \overline{I} n_P + \overline{P}$. Если рассмотреть смещение в памяти $iN + j$ элемента M_{ij} некой матрицы размерности M×N, то можно вывести формулы, позволяющие логически представить тензор $\mathcal{T}$ в виде матрицы размерности $n_I \times n_P$:

$$\overline{I} = \sum_{k=0}^{d_I-1} i_k \prod_{l=k+1}^{d_I-1} n_{i_l},$$

$$n_P = \prod_{l=0}^{d_P-1} n_{p_l}, \quad n_I = \prod_{l=0}^{d_I-1} n_{i_l},$$

$$\overline{P} = \sum_{k=0}^{d_P-1} p_k \prod_{l=k+1}^{d_P-1} n_{p_l}.$$

Другое отличие GETT в применении автонастройки для нахождения наилучших значений параметров разбиения и способов выполнить упаковку элементов тензора. Использование автонастройки позволяет повысить производительность получаемого кода, но делает невозможным применение GETT в условиях недоступности целевой архитектуры и ограниченности времени выполнения. Кроме этого GETT не поддерживает тензоры, размер которых неизвестен на этапе компиляции.

Подход, используемый в TBLIS, может быть объединен с аналитическим моделированием для устранения необходимости в коде MMM, оптимизированном вручную. Полученный метод совместно с алгоритмами для распознавания кода TC может применяться автоматически в процессе оптимизации, выполняемой промышленным компилятором по умолчанию во время компиляции и кросс-компиляции. На рис. 6 изображен комплекс автоматизированной оптимизации обобщенных тензорных операций (AOOTO) [8, 28]. В процессе работы AOOTO фронтенд Clang [55] строит промежуточное представление программы, используемое комплексом оптимизаций Polly [56]. Polly создает специальное математическое представление программы с целью распознавания TC и ее оптимизации с применением аналитического моделирования и метода, предложенного в TBLIS. Библиотека LLVM Core [55] использована для генерации ассемблерного кода целевой архитектуры процессора.

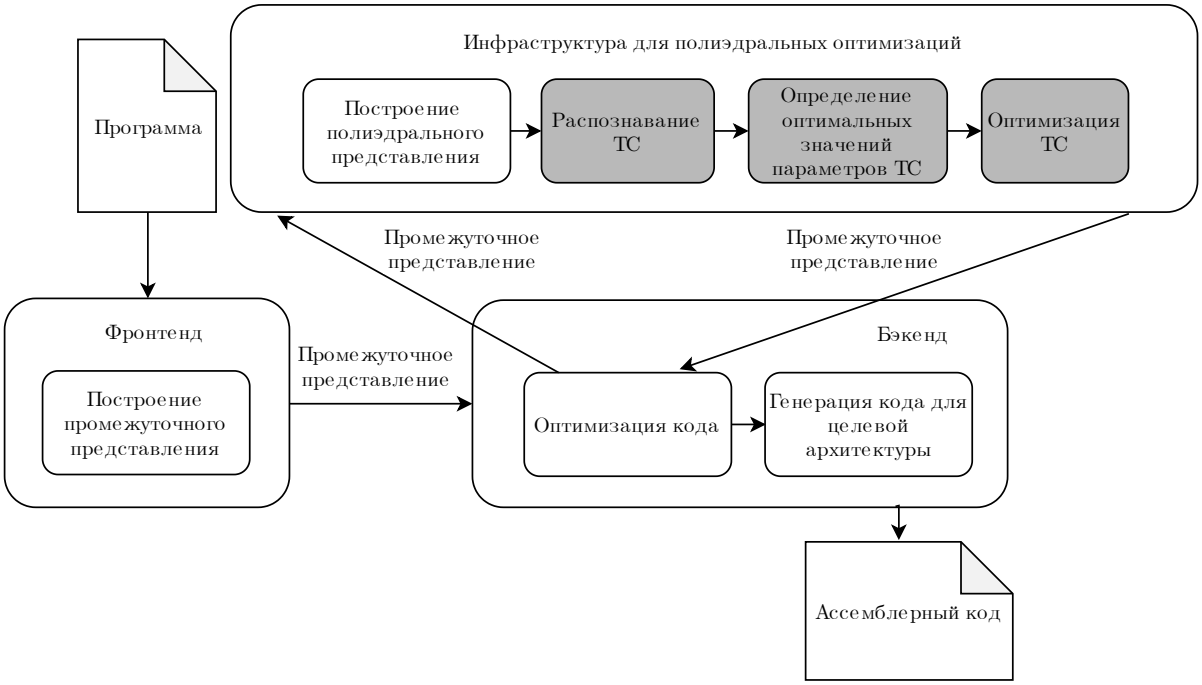

Рис. 6. Комплекс программ AOOTO

Для оптимизации TC могут применяться последовательности стандартных оптимизаций циклов (перестановка циклов, разбиение циклов на блоки, размотка циклов, векторизация и др.). Такие подходы зависят от размерности тензоров и их расположения в памяти [57]. В частности, производительность TC для задач малой размерности может быть улучшена за счет векторизации циклов [58]. Для задач большой размерности может применяться генератор кода циклов для графических ускорителей [59].

Заключение

TC является операцией «Тензорного исчисления», отдельного раздела математики, созданного для решения теоретических вопросов, требующих более общего исчисления чем векторное. TC, обобщение TC на произвольные полукольца матриц, а также MMM и MVM, частные случаи TC, широко применяются как в теории, так и на практике. Вследствие этого, эффективное выполнение TC имеет существенную практическую значимость.

В статье представлен обзор походов к оптимизации TC. Большинство таких методов основывается на преобразовании тензоров к матричной форме с целью применения оп-

тимизаций MMM и MVM, или использования отдельных частей оптимизированного кода MMM и MVM. Описывается представление тензоров, позволяющее свести оптимизацию ТС к оптимизации MMM и MVM.

Рассмотрены подходы к оптимизации MMM и MVM, реализованные в виде проприетарных библиотек и свободного программного обеспечения. Большая их часть основывается на ручной настройке и самонастройке, осложняющих использование в процессе кросс-компиляции, выполняемой промышленными компиляторами, и других условиях ограниченного времени и отсутствия доступа к целевой архитектуре. Время работы таких методов может быть уменьшено за счет применения аналитического моделирования, устанавливающего зависимость между использованием ресурсов гипотетического процессора и производительностью программы. В случае распределенных архитектур время выполнения MMM и MVM может быть уменьшено достижением нижней асимптотической границы на перемещение данных в иерархии памяти, а также между процессорами.

Литература

1. Коренев Г.В. Тензорное исчисление. Москва: Изд-во МФТИ, 2000. 240 с.
2. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. URL: `https://www.tensorflow.org/` (дата обращения: 02.02.2020).
3. Pekurovsky D. P3DFFT: A Framework for Parallel Computations of Fourier Transforms in Three Dimensions // SIAM Journal on Scientific Computing. 2012. Vol. 34, no. 4. P. 192–209. DOI: 10.1137/11082748X.
4. Deville M.O., Fischer P.F., Mund E.H High-Order Methods for Incompressible Fluid Flow. Cambridge University Press, 2002. 489 p.
5. Jayachandran S., Venkatachalam T. A Secure Scheme for Privacy Preserving Data Mining Using Matrix Encoding // Australian Journal of Basic and Applied Sciences (AJBAS). 2015. URL: `http://www.ajbasweb.com/old/ajbas/2015/July/741-744.pdf` (дата обращения: 17.03.2020).
6. Cong J., Xiao B. Minimizing Computation in Convolutional Neural Networks // Artificial Neural Networks and Machine Learning – ICANN. Springer International Publishing, Cham. 2014. P. 281–290. DOI: 10.1007/978-3-319-11179-7_36.
7. Watson M., Olivares-Amaya R., Edgar R.G., Aspuru-Guzik A. Accelerating Correlated Quantum Chemistry Calculations Using Graphical Processing Units // Computing in Science Engineering. 2010. Vol. 12, no. 4. P. 40–51. DOI: 10.1021/ct900543q.
8. Акимова Е.Н., Гареев Р.А. Аналитическое моделирование матрично-векторного произведения на многоядерных процессорах // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 1. С. 69–82. DOI: 10.14529/cmse200105.
9. Goto K., Geijn R. Anatomy of High-performance Matrix Multiplication // ACM Transactions on Mathematical Software. 2008. Vol. 34, no. 3. P. 1–25. DOI: 10.1145/1356052.1356053.
10. Yu J., Lukefahr A., Palframan D., Dasika G., Das R., Mahlke S. Scalpel: Customizing DNN pruning to the underlying hardware parallelism // 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (Ontario, Canada, June, 24–28, 2017). New York, ACM. 2017. P. 548–560. DOI: 10.1145/3079856.3080215.

11. Yang X., Parthasarathy S., Sadayappan P. Fast Sparse Matrix-Vector Multiplication on GPUs: Implications for Graph Mining // Proceedings of the VLDB Endowment. 2011. Vol. 4, no. 4. P. 231–242. DOI: 10.14778/1938545.1938548.

12. Kaushik D., Gropp W., Minkoff M., Smith B. Improving the Performance of Tensor Matrix Vector Multiplication in Cumulative Reaction Probability Based Quantum Chemistry Codes // High Performance Computing – HiPC 2008 (Bangalore, India, December, 17–20, 2008). Heidelberg, Springer-Verlag. 2008. P. 120–130. DOI: 10.1007/978-3-540-89894-8_14.

13. Lawson C.L., Hanson R.J., Kincaid D.R., Krogh F.T. Basic Linear Algebra Subprograms for Fortran Usage // ACM Transactions on Mathematical Software. 1979. Vol. 5. no. 3. P. 308–323. DOI: 10.1145/355841.355847.

14. Dongarra J.J., Du Croz J., Hammarling S., Hanson R.J. An Extended Set of FORTRAN Basic Linear Algebra Subprograms // ACM Transactions on Mathematical Software. 1988. Vol. 14, no. 1. P. 1–17. DOI: 10.1145/42288.42291.

15. Dongarra J.J., Du Croz J., Hammarling S., Duff I.S. A Set of Level 3 Basic Linear Algebra Subprograms // ACM Transactions on Mathematical Software. 1990. Vol. 16, no. 1. P. 1–17. DOI: 10.1145/77626.79170.

16. Sedukhin S.G., Paprzycki M. Generalizing Matrix Multiplication for Efficient Computations on Modern Computers // Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics – Volume Part I (Reykjavik, Iceland, September, 9–13, 2006). Heidelberg, Springer-Verlag. 2006. P. 225–234. DOI: 10.1007/978-3-642-31464-3_23.

17. Low T.M., Igual F.D., Smith T.M., Quintana-Orti E.S. Analytical Modeling Is Enough for High-Performance BLIS // ACM Transactions on Mathematical Software. 2016. Vol. 43, no. 2. P. 12:1–12:18. DOI: 10.1145/2925987.

18. Gates M., Luszczek P., Abdelfattah A., Kurzak J., Dongarra J., Arturov K., Cecka C., Freitag C. C++ API for BLAS and LAPACK. SLATE Working Notes. 2017. Vol. 2. P. 1–45. URL: https://www.icl.utk.edu/files/publications/2017/icl-utk-1031-2017.pdf (дата обращения: 17.03.2020).

19. Goto K., Van De Geijn R. High-performance Implementation of the Level-3 BLAS // ACM Transactions on Mathematical Software. 2008. Vol. 35. no. 1. P. 4:1–4:14. DOI: 10.1145/1377603.1377607.

20. Xianyi Z., Qian W., Yunquan Z. Model-driven level 3 BLAS performance optimization on Loongson 3A processor // 2012 IEEE 18th International Conference on Parallel and Distributed Systems (Singapore, December, 17–19, 2012). Massachusetts Ave. 2012. P. 684–691. DOI: 10.1109/ICPADS.2012.97.

21. Van Zee F., Van De Geijn R. BLIS: A Framework for Rapidly Instantiating BLAS Functionality // ACM Transactions on Mathematical Software. 2015. Vol. 41, no. 3. P. 14:1–14:33. DOI: 10.1145/2764454.

22. Intel Math Kernel Library (Intel MKL) URL: https://software.intel.com/ru-ru/intel-mkl/?cid=sem43700011401059448&intel_term=intel+mkl&gclid=CIjbtvqaqM8CFSoNcwodDPUAbw&gclsrc=aw.ds (дата обращения: 02.02.2020).

23. IBM Engineering and Scientific Subroutine Library Intel Math Kernel Library (Intel MKL). URL: https://www.ibm.com/support/knowledgecenter/SSFHY8_6.1/reference/essl_reference_pdf.pdf (дата обращения: 02.02.2020).

24. ARM Performance Libraries Reference Manual. URL: `https://ds.arm.com/media/uploads/arm_performance_libraries_reference_manual_arm-ecm-0493690.pdf` (дата обращения: 02.02.2020).

25. Whaley R.C., Dongarra J.J. Automatically Tuned Linear Algebra Software // Proceedings of the 1998 ACM/IEEE Conference on Supercomputing (Montreal, Canada, November, 7, 1998). New York, ACM. 1998. P. 1–27. DOI: 10.5555/509058.509096.

26. Bilmes J., Asanovic K., Chin C., Demmel J. Optimizing Matrix Multiply Using PHiPAC: A Portable, High-performance, ANSI C Coding Methodology // Proceedings of the 11th International Conference on Supercomputing (Vienna, Austria, July, 7–11, 1997). New York, ACM. 1997. P. 340–347. DOI: 10.1145/2591635.2667174.

27. Su X., Liao X., Xue J. Automatic Generation of Fast BLAS3-GEMM: A Portable Compiler Approach // Proceedings of the 2017 International Symposium on Code Generation and Optimization (Austin, TX, USA, February, 4–8, 2017). IEEE Press. 2017. P. 122–133. DOI: 10.1109/CGO.2017.7863734.

28. Gareev R., Grosser T., Michael K. High-Performance Generalized Tensor Operations: A Compiler-Oriented Approach // ACM Transactions on Architecture and Code Optimization. 2018. Vol. 15, no. 3. P. 34:1–34:27. DOI: 10.1145/3235029.

29. Spampinato D.G., Markus P. A Basic Linear Algebra Compiler // International Symposium on Code Generation and Optimization (Orlando, FL, USA, February, 15–19, 2014). New York, ACM. 2014. P. 23–32. DOI: 10.1145/2581122.2544155.

30. Belter G., Jessup E.R., Karlin I., Siek J.G. Automating the Generation of Composed Linear Algebra Kernels // Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (Portland, OR, USA, November, 14–20, 2009). New York, ACM. 2009. P. 59:1-59:12. DOI: 10.1145/1654059.1654119.

31. Wang Q., Zhang X., Zhang Y., Yi Q. AUGEM: Automatically Generate High Performance Dense Linear Algebra Kernels on x86 CPUs // Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (Denver, Colorado, USA, November, 17–22, 2013). New York, ACM. 2013. P. 25:1–25:12. DOI: 10.1145/2503210.2503219.

32. Yi Q., Wang Q., Cui H. Specializing Compiler Optimizations Through Programmable Composition for Dense Matrix Computations // Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (Cambridge, UK, December, 13–17, 2014). Washington, IEEE Computer Society. 2014. P. 596–608. DOI: 10.1109/MICRO.2014.14.

33. Knijnenburg P.M., Kisuki T., O'Boyle M.F.P. Iterative Compilation // Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation (SAMOS). Springer Berlin Heidelberg, 2002. P. 171–187. DOI: 10.1007/3-540-45874-3_10.

34. Yotov K., Xiaoming L., Gang R., Garzaran M.J.S., Padua D., Pingali K., Stodghill P. Is Search Really Necessary to Generate High-Performance BLAS? // Proceedings of the IEEE. 2005. Vol. 93, no. 2. P. 358–386. DOI: 10.1109/JPROC.2004.840444.

35. Akimova E.N., Gareev R.A. Algorithm of Automatic Parallelization of Generalized Matrix Multiplication // CEUR Workshop Proceedings. 2017. Vol. 2005. P. 1–10.

36. Demmel J. Communication Avoiding Algorithms // Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis (Salt Lake City, UT, USA,

November, 10–16, 2012). Massachusetts Ave., IEEE Computer Society. 2012. P. 1942–2000. DOI: 10.1109/SC.Companion.2012.351.

37. Ballard G., Carson E., Demmel J., Hoemmen M., Knight N., Schwartz O. Communication lower bounds and optimal algorithms for numerical linear algebra // Acta Numerica. 2014. Vol. 23. P. 1–155. DOI: 10.1017/S0962492914000038.

38. Frigo M., Leiserson C.E., Prokop H., Ramachandran S. Cache-Oblivious Algorithms // Proceedings of the 40th Annual Symposium on Foundations of Computer Science (New York City, New York, USA, October, 17–19, 1999). Massachusetts Ave., IEEE Computer Society. 2012. P. 285–298. DOI: 10.5555/795665.796479.

39. Yotov K., Roeder T., Pingali K., Gunnels J., Gustavson F. An Experimental Comparison of Cache-oblivious and Cache-conscious Programs // Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures (San Diego, California, USA, June, 9–11, 2007). New York, ACM. 2007. P. 93–104. DOI: 10.1145/1248377.1248394.

40. Liang J., Zhang Y. Optimization of GEMV on Intel AVX processor // International Journal of Database Theory and Application. 2016. Vol. 9. P. 47–60. DOI: 10.14257/ijdta.2016.9.2.06.

41. Hassan S.A., Mahmoud M.M.M., Hemeida A.M., Saber M.A. Effective Implementation of MatrixVector Multiplication on Intel's AVX Multicore Processor // Computer Languages, Systems & Structures. 2018. Vol. 51. P. 158–175. DOI: 10.1016/j.cl.2017.06.003.

42. Akimova E.N., Gareev R.A., Misilov V.E. Analytical Modeling of Matrix-Vector Multiplication on Multicore Processors: Solving Inverse Gravimetry Problem // 2019 International Multi-Conference on Engineering, Computer and Information Sciences (Novosibirsk, Russia, October, 21–27, 2019). Massachusetts, IEEE Xplore Digital Library. 2019. P. 0823–0827. DOI: 10.1109/SIBIRCON48586.2019.8958103.

43. Heinecke A., Pabst H., Henry G. LIBXSMM: A high performance library for small matrix multiplications // HPC transforms (Austin, TX, USA, November, 15–20, 2015). New York, ACM. 2015. P. 1–1. DOI: 10.1109/SC.2016.83.

44. Napoli E.D., Fabregat-Traver D., Quintana-Ortí G., Bientinesi P. Towards an efficient use of the BLAS library for multilinear tensor contractions // Applied Mathematics and Computation. 2014. Vol. 235. P. 454–468. DOI: 10.1016/j.amc.2014.02.051.

45. Matthews D. High-Performance Tensor Contraction without BLAS // SIAM Journal on Scientific Computing. 2018. Vol. 40, no. 1. P. 1–24. DOI: 10.1137/16m108968x.

46. Springer P., Bientinesi P. Design of a High-Performance GEMM-like Tensor-Tensor Multiplication // ACM Transactions on Mathematical Software. 2018. Vol. 44, no. 3. P. 28:1–28:29. DOI: 10.1145/3157733.

47. Hirata S. Tensor Contraction Engine: Abstraction and Automated Parallel Implementation of Configuration-Interaction, Coupled-Cluster, and Many-Body Perturbation Theories // The Journal of Physical Chemistry A. 2003. Vol. 107, no. 46. P. 9887–9897. DOI: 10.1021/jp034596z.

48. Bader B.W., Kolda G.T. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping // ACM Transactions on Mathematical Software. 2006. Vol. 32. P. 635–653. DOI: 10.1145/1186785.1186794.

49. Walt S., Colbert C., Varoquaux G. The NumPy Array: A Structure for Efficient Numerical Computation // Computing in Science & Engineering. 2011. Vol. 13. P. 22–30. DOI: 10.1109/MCSE.2011.37.

50. Eigen v3. URL: `http://eigen.tuxfamily.org` (дата обращения: 02.02.2020).

51. Solomonik E., Matthews D., Hammond J., Stanton J.F., Demmel J. A massively parallel tensor contraction framework for coupled-cluster computations // Journal of Parallel and Distributed Computing. 2014. Vol. 74, no. 12. P. 3176–3190. DOI: 10.1016/j.jpdc.2014.06.002.

52. Epifanovsky E., Wormit M., Ku T., Landau A., Zuev D., Khistyaev K., Manohar P., Kaliman I., Dreuw A., Krylov A. New implementation of high-level correlated methods using a general block tensor library for high-performance electronic structure calculations // Journal of computational chemistry. 2013. Vol. 34. P. 2293–2309. DOI: 10.1002/jcc.23377.

53. Calvin J., Lewis C.A., Valeev E.F. Scalable Task-Based Algorithm for Multiplication of Block-Rank-Sparse Matrices // Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms (Austin, Texas, USA, November, 15, 2015). New York, ACM. 2015. P. 4:1–4:8. DOI: 10.1145/2833179.2833186.

54. Calvin J., Valeev E.F. Task-Based Algorithm for Matrix Multiplication: A Step Towards Block-Sparse Tensor Computing // Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms (Austin, Texas, USA, November, 15, 2015). New York, ACM. 2015. P. 1–9.

55. Lattner C. LLVM: An Infrastructure for Multi-Stage Optimization // Master's Thesis. 2002. URL: `http://llvm.cs.uiuc.edu` (дата обращения: 27.10.2019).

56. Grosser T., Größlinger A., Lengauer C. Polly—Performing polyhedral optimizations on a low-level intermediate representation // Parallel Processing Letters. 2012. Vol. 22. No. 4. DOI: 10.1142/S0129626412500107.

57. Apra E., Klemm M., Kowalski K. Efficient Implementation of Many-Body Quantum Chemical Methods on the Intel E; Xeon Phi Coprocessor // International Conference for High Performance Computing, Networking, Storage and Analysis (New Orleans, LA, USA, November, 16–21, 2014). Massachusetts Ave., IEEE Computer Society. 2014. P. 674–684. DOI: 10.1109/SC.2014.60.

58. Stock K., Henretty T., Murugandi I., Sadayappan P., Harrison R. Model-Driven SIMD Code Generation for a Multi-resolution Tensor Kernel // Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium (New Orleans, LA, USA, May, 16–20, 2011). Massachusetts Ave., IEEE Computer Society. 2014. P. 1058–1067. DOI: 10.1109/IPDPS.2011.101.

59. Ma W., Krishnamoorthy S., Villa O., Kowalski K. GPU-Based Implementations of the Noniterative Regularized-CCSD(T) Corrections: Applications to Strongly Correlated Systems // Journal of Chemical Theory and Computation. 2011. Vol. 7, no. 5. P. 1316–1327. DOI: 10.1021/ct1007247.

Гареев Роман Альбертович, аспирант, Институт радиоэлектроники и информационных технологий-РтФ, УрФУ имени первого Президента России Б.Н. Ельцина (Екатеринбург, Российская Федерация)

DOI: 10.14529/cmse200202

OPTIMIZATION METHODS FOR GENERALIZED TENSOR CONTRACTION

© 2020 R.A. Gareev

Ural Federal University (Mira 19, Yekaterinburg, 620002 Russia)

E-mail: gareevroman@gmail.com

Received: 18.03.2020

Tensor contraction is one of major operations defined in tensor calculus, a separate branch of mathematics, which become a fundamental language of theory of relativity, mechanics, electrodynamics, and solid state physics. Effective implementation of tensor contraction is of considerable practical significance for such areas as solving of mathematical physics problems, machine learning, spectral element methods, quantum chemistry, data mining, and high performance computing. In the last twenty years, the number of optimization methods for tensor contraction has increased and continues to grow. In this article, the author reviews widespread approaches for optimization of tensor contraction, which are used on single processor as well as multiprocessor systems with distributed memory. The review contains the description of methods for optimization of matrix and matrix-vector multiplications, important particular cases of tensor contraction, which are used as a base for the most tensor contraction optimizations. The described optimizations can be applied during program compilation performed by production compilers. The information provided in this work could be useful for systematizing knowledge.

Keywords: tensor contraction, linear algebra, high-performance computing.

FOR CITATION

Gareev R.A. Optimization Methods for Generalized Tensor Contraction. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 2. P. 19–39. (in Russian) DOI: 10.14529/cmse200202.

References

1. Korenev G.V. Tensor analysis. Moscow, MIPT, 2000. 240 p.
2. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available at: `https://www.tensorflow.org/` (accessed: 02.02.2020).
3. Pekurovsky D. P3DFFT: A Framework for Parallel Computations of Fourier Transforms in Three Dimensions. SIAM Journal on Scientific Computing. 2012. Vol. 34, no. 4. P. 192–209. DOI: 10.1137/11082748X.
4. Deville M.O., Fischer P.F., Mund E.H High-Order Methods for Incompressible Fluid Flow. Cambridge University Press, 2002. 489 p.
5. Jayachandran S., Venkatachalam T. A Secure Scheme for Privacy Preserving Data Mining Using Matrix Encoding. Australian Journal of Basic and Applied Sciences (AJBAS). 2015. Available at: `http://www.ajbasweb.com/old/ajbas/2015/July/741-744.pdf` (accessed: 17.03.2020).
6. Cong J., Xiao B. Minimizing Computation in Convolutional Neural Networks. Artificial Neural Networks and Machine Learning – ICANN. Springer International Publishing, 2014. P. 281–290. DOI: 10.1007/978-3-319-11179-7_36.

7. Watson M., Olivares-Amaya R., Edgar R.G., Aspuru-Guzik A. Accelerating Correlated Quantum Chemistry Calculations Using Graphical Processing Units. Computing in Science Engineering. 2010. Vol. 12, no. 4. P. 40–51. DOI: 10.1021/ct900543q.

8. Akimova E.N., Gareev R.A. Application of Analytical Modeling of Matrix-Vector Multiplication on Multicore Processors. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2020. Vol. 9, no. 1. P. 69–82. (in Russian) DOI: 10.14529/cmse200105.

9. Goto K., Geijn R. Anatomy of High-performance Matrix Multiplication. ACM Transactions on Mathematical Software. 2008. Vol. 34, no. 3. P. 1–25. DOI: 10.1145/1356052.1356053.

10. Yu J., Lukefahr A., Palframan D., Dasika G., Das R., Mahlke S. Scalpel: Customizing DNN pruning to the underlying hardware parallelism. 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (Ontario, Canada, June, 24–28, 2017). New York, ACM. 2017. P. 548–560. DOI: 10.1145/3079856.3080215.

11. Yang X., Parthasarathy S., Sadayappan P. Fast Sparse Matrix-Vector Multiplication on GPUs: Implications for Graph Mining. Proceedings of the VLDB Endowment. 2011. Vol. 4, no. 4. P. 231–242. DOI: 10.14778/1938545.1938548.

12. Kaushik D., Gropp W., Minkoff M., Smith B. Improving the Performance of Tensor Matrix Vector Multiplication in Cumulative Reaction Probability Based Quantum Chemistry Codes. High Performance Computing – HiPC 2008 (Bangalore, India, December, 17–20, 2008). Heidelberg, Springer-Verlag. 2008. P. 120–130. DOI: 10.1007/978-3-540-89894-8_14.

13. Lawson C.L., Hanson R.J., Kincaid D.R., Krogh F.T. Basic Linear Algebra Subprograms for Fortran Usage. ACM Transactions on Mathematical Software. 1979. Vol. 5. no. 3. P. 308–323. DOI: 10.1145/355841.355847.

14. Dongarra J.J., Du Croz J., Hammarling S., Hanson R.J. An Extended Set of FORTRAN Basic Linear Algebra Subprograms. ACM Transactions on Mathematical Software. 1988. Vol. 14, no. 1. P. 1–17. DOI: 10.1145/42288.42291.

15. Dongarra J.J., Du Croz J., Hammarling S., Duff I.S. A Set of Level 3 Basic Linear Algebra Subprograms. ACM Transactions on Mathematical Software. 1990. Vol. 16, no. 1. P. 1–17. DOI: 10.1145/77626.79170.

16. Sedukhin S.G., Paprzycki M. Generalizing Matrix Multiplication for Efficient Computations on Modern Computers. Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics – Volume Part I (Reykjavik, Iceland, September, 9–13, 2006). Heidelberg, Springer-Verlag. 2006. P. 225–234. DOI: 10.1007/978-3-642-31464-3_23.

17. Low T.M., Igual F.D., Smith T.M., Quintana-Orti E.S. Analytical Modeling Is Enough for High-Performance BLIS. ACM Transactions on Mathematical Software. 2016. Vol. 43, no. 2. P. 12:1–12:18. DOI: 10.1145/2925987.

18. Gates M., Luszczek P., Abdelfattah A., Kurzak J., Dongarra J., Arturov K., Cecka C., Freitag C. C++ API for BLAS and LAPACK. SLATE Working Notes. 2017. Vol. 2. P. 1–45. Available at: `https://www.icl.utk.edu/files/publications/2017/icl-utk-1031-2017.pdf` (accessed: 17.03.2020).

19. Goto K., Van De Geijn R. High-performance Implementation of the Level-3 BLAS. ACM Transactions on Mathematical Software. 2008. Vol. 35. no. 1. P. 4:1–4:14. DOI: 10.1145/1377603.1377607.

20. Xianyi Z., Qian W., Yunquan Z. Model-driven level 3 BLAS performance optimization on Loongson 3A processor. 2012 IEEE 18th International Conference on Parallel and Distributed Systems (Singapore, December, 17–19, 2012). Massachusetts Ave. 2012. P. 684–691. DOI: 10.1109/ICPADS.2012.97.

21. Van Zee F., Van De Geijn R. BLIS: A Framework for Rapidly Instantiating BLAS Functionality. ACM Transactions on Mathematical Software. 2015. Vol. 41, no. 3. P. 14:1–14:33. DOI: 10.1145/2764454.

22. Intel Math Kernel Library (Intel MKL) Available at: `https://software.intel.com/ru-ru/intel-mkl/?cid=sem43700011401059448&intel_term=intel+mkl&gclid=CIjbtvqaqM8CFSoNcwodDPUAbw&gclsrc=aw.ds` (accessed: 02.02.2020).

23. IBM Engineering and Scientific Subroutine Library Intel Math Kernel Library (Intel MKL). Available at: `https://www.ibm.com/support/knowledgecenter/SSFHY8_6.1/reference/essl_reference_pdf.pdf` (accessed: 02.02.2020).

24. ARM Performance Libraries Reference Manual. Available at: `https://ds.arm.com/media/uploads/arm_performance_libraries_reference_manual_arm-ecm-0493690.pdf` (accessed: 02.02.2020).

25. Whaley R.C., Dongarra J.J. Automatically Tuned Linear Algebra Software. Proceedings of the 1998 ACM/IEEE Conference on Supercomputing (Montreal, Canada, November, 7, 1998). New York, ACM. 1998. P. 1–27. DOI: 10.5555/509058.509096.

26. Bilmes J., Asanovic K., Chin C., Demmel J. Optimizing Matrix Multiply Using PHiPAC: A Portable, High-performance, ANSI C Coding Methodology. Proceedings of the 11th International Conference on Supercomputing (Vienna, Austria, July, 7–11, 1997). New York, ACM. 1997. P. 340–347. DOI: 10.1145/2591635.2667174.

27. Su X., Liao X., Xue J. Automatic Generation of Fast BLAS3-GEMM: A Portable Compiler Approach. Proceedings of the 2017 International Symposium on Code Generation and Optimization (Austin, TX, USA, February, 4–8, 2017). IEEE Press. 2017. P. 122–133. DOI: 10.1109/CGO.2017.7863734.

28. Gareev R., Grosser T., Michael K. High-Performance Generalized Tensor Operations: A Compiler-Oriented Approach. ACM Transactions on Architecture and Code Optimization. 2018. Vol. 15, no. 3. P. 34:1–34:27. DOI: 10.1145/3235029.

29. Spampinato D.G., Markus P. A Basic Linear Algebra Compiler. International Symposium on Code Generation and Optimization (Orlando, FL, USA, February, 15–19, 2014). New York, ACM. 2014. P. 23–32. DOI: 10.1145/2581122.2544155.

30. Belter G., Jessup E.R., Karlin I., Siek J.G. Automating the Generation of Composed Linear Algebra Kernels. Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (Portland, OR, USA, November, 14–20, 2009). New York, ACM. 2009. P. 59:1-59:12. DOI: 10.1145/1654059.1654119.

31. Yi Q., Wang Q., Cui H. Specializing Compiler Optimizations Through Programmable Composition for Dense Matrix Computations. Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (Cambridge, UK, December, 13–17, 2014). Washington, IEEE Computer Society. 2014. P. 596–608. DOI: 10.1109/MICRO.2014.14.

32. Wang Q., Zhang X., Zhang Y., Yi Q. AUGEM: Automatically Generate High Performance Dense Linear Algebra Kernels on x86 CPUs. Proceedings of the International

Conference on High Performance Computing, Networking, Storage and Analysis (Denver, Colorado, USA, November, 17–22, 2013). New York, ACM. 2013. P. 25:1–25:12. DOI: 10.1145/2503210.2503219.

33. Knijnenburg P.M., Kisuki T., O'Boyle M.F.P. Iterative Compilation. Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation (SAMOS). Springer Berlin Heidelberg, 2002. P. 171–187. DOI: 10.1007/3-540-45874-3_10.

34. Yotov K., Xiaoming L., Gang R., Garzaran M.J.S., Padua D., Pingali K., Stodghill P. Is Search Really Necessary to Generate High-Performance BLAS? Proceedings of the IEEE. 2005. Vol. 93, no. 2. P. 358–386. DOI: 10.1109/JPROC.2004.840444.

35. Akimova E.N., Gareev R.A. Algorithm of Automatic Parallelization of Generalized Matrix Multiplication. CEUR Workshop Proceedings. 2017. Vol. 2005. P. 1–10.

36. Demmel J. Communication Avoiding Algorithms. Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis (Salt Lake City, UT, USA, November, 10–16, 2012). Massachusetts Ave., IEEE Computer Society. 2012. P. 1942–2000. DOI: 10.1109/SC.Companion.2012.351.

37. Ballard G., Carson E., Demmel J., Hoemmen M., Knight N., Schwartz O. Communication lower bounds and optimal algorithms for numerical linear algebra. Acta Numerica. 2014. Vol. 23. P. 1–155. DOI: 10.1017/S0962492914000038.

38. Frigo M., Leiserson C.E., Prokop H., Ramachandran S. Cache-Oblivious Algorithms. Proceedings of the 40th Annual Symposium on Foundations of Computer Science (New York City, New York, USA, October, 17–19, 1999). Massachusetts Ave., IEEE Computer Society. 2012. P. 285–298. DOI: 10.5555/795665.796479.

39. Yotov K., Roeder T., Pingali K., Gunnels J., Gustavson F. An Experimental Comparison of Cache-oblivious and Cache-conscious Programs. Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures (San Diego, California, USA, June, 9–11, 2007). New York, ACM. 2007. P. 93–104. DOI: 10.1145/1248377.1248394.

40. Liang J., Zhang Y. Optimization of GEMV on Intel AVX processor. International Journal of Database Theory and Application. 2016. Vol. 9. P. 47–60. DOI: 10.14257/ijdta.2016.9.2.06.

41. Hassan S.A., Mahmoud M.M.M., Hemeida A.M., Saber M.A. Effective Implementation of MatrixVector Multiplication on Intel's AVX Multicore Processor. Computer Languages, Systems & Structures. 2018. Vol. 51. P. 158–175. DOI: 10.1016/j.cl.2017.06.003.

42. Akimova E.N., Gareev R.A., Misilov V.E. Analytical Modeling of Matrix-Vector Multiplication on Multicore Processors: Solving Inverse Gravimetry Problem. 2019 International Multi-Conference on Engineering, Computer and Information Sciences (Novosibirsk, Russia, October, 21–27, 2019). Massachusetts, IEEE Xplore Digital Library. 2019. P. 0823–0827. DOI: 10.1109/SIBIRCON48586.2019.8958103.

43. Heinecke A., Pabst H., Henry G. LIBXSMM: A high performance library for small matrix multiplications. HPC transforms (Austin, TX, USA, November, 15–20, 2015). New York, ACM. 2015. P. 1–1. DOI: 10.1109/SC.2016.83.

44. Napoli E.D., Fabregat-Traver D., Quintana-Ortí G., Bientinesi P. Towards an efficient use of the BLAS library for multilinear tensor contractions. Applied Mathematics and Computation. 2014. Vol. 235. P. 454–468. DOI: 10.1016/j.amc.2014.02.051.

45. Matthews D. High-Performance Tensor Contraction without BLAS. SIAM Journal on Scientific Computing. 2018. Vol. 40, no. 1. P. 1–24. DOI: 10.1137/16m108968x.

46. Springer P., Bientinesi P. Design of a High-Performance GEMM-like Tensor-Tensor Multiplication. ACM Transactions on Mathematical Software. 2018. Vol. 44, no. 3. P. 28:1–28:29. DOI: 10.1145/3157733.

47. Hirata S. Tensor Contraction Engine: Abstraction and Automated Parallel Implementation of Configuration-Interaction, Coupled-Cluster, and Many-Body Perturbation Theories. The Journal of Physical Chemistry A. 2003. Vol. 107, no. 46. P. 9887–9897. DOI: 10.1021/jp034596z.

48. Bader B.W., Kolda G.T. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. ACM Transactions on Mathematical Software. 2006. Vol. 32. P. 635–653. DOI: 10.1145/1186785.1186794.

49. Walt S., Colbert C., Varoquaux G. The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering. 2011. Vol. 13. P. 22–30. DOI: 10.1109/MCSE.2011.37.

50. Eigen v3. Available at: `http://eigen.tuxfamily.org` (accessed: 02.02.2020).

51. Solomonik E., Matthews D., Hammond J., Stanton J.F., Demmel J. A massively parallel tensor contraction framework for coupled-cluster computations. Journal of Parallel and Distributed Computing. 2014. Vol. 74, no. 12. P. 3176–3190. DOI: 10.1016/j.jpdc.2014.06.002.

52. Epifanovsky E., Wormit M., Ku T., Landau A., Zuev D., Khistyaev K., Manohar P., Kaliman I., Dreuw A., Krylov A. New implementation of high-level correlated methods using a general block tensor library for high-performance electronic structure calculations. Journal of computational chemistry. 2013. Vol. 34. P. 2293–2309. DOI: 10.1002/jcc.23377.

53. Calvin J., Lewis C.A., Valeev E.F. Scalable Task-Based Algorithm for Multiplication of Block-Rank-Sparse Matrices. Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms (Austin, Texas, USA, November, 15, 2015). New York, ACM. 2015. P. 4:1–4:8. DOI: 10.1145/2833179.2833186.

54. Calvin J., Valeev E.F. Task-Based Algorithm for Matrix Multiplication: A Step Towards Block-Sparse Tensor Computing. Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms (Austin, Texas, USA, November, 15, 2015). New York, ACM. 2015. P. 1–9.

55. Lattner C. LLVM: An Infrastructure for Multi-Stage Optimization. Master's Thesis. 2002. Available at: `http://llvm.cs.uiuc.edu` (accessed: 27.10.2019).

56. Grosser T., Größlinger A., Lengauer C. Polly—Performing polyhedral optimizations on a low-level intermediate representation. Parallel Processing Letters. 2012. Vol. 22. No. 4. DOI: 10.1142/S0129626412500107.

57. Apra E., Klemm M., Kowalski K. Efficient Implementation of Many-Body Quantum Chemical Methods on the Intel E; Xeon Phi Coprocessor. International Conference for High Performance Computing, Networking, Storage and Analysis (New Orleans, LA, USA, November, 16–21, 2014). Massachusetts Ave., IEEE Computer Society. 2014. P. 674–684. DOI: 10.1109/SC.2014.60.

58. Stock K., Henretty T., Murugandi I., Sadayappan P., Harrison R. Model-Driven SIMD Code Generation for a Multi-resolution Tensor Kernel. Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium (New Orleans, LA, USA, May, 16–20, 2011). Massachusetts Ave., IEEE Computer Society. 2014. P. 1058–1067. DOI: 10.1109/IPDPS.2011.101.

59. Ma W., Krishnamoorthy S., Villa O., Kowalski K. GPU-Based Implementations of the Noniterative Regularized-CCSD(T) Corrections: Applications to Strongly Correlated Systems. Journal of Chemical Theory and Computation. 2011. Vol. 7, no. 5. P. 1316–1327. DOI: 10.1021/ct1007247.

УДК 519.612, 519.684.4 DOI: 10.14529/cmse200203

ПАРАЛЛЕЛЬНОЕ РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ НА ГИБРИДНОЙ АРХИТЕКТУРЕ CPU+GPU*

© 2020 Н.С. Недожогин, С.П. Копысов, А.К. Новиков

Удмуртский государственный университет

(426034 Ижевск, ул. Университетская, д. 1)

E-mail: nedozhogin07@gmail.com, s.kopysov@gmail.com, sc_work@mail.ru

Поступила в редакцию: 29.01.2020

В статье рассматривается параллельная реализация решения систем линейных алгебраических уравнений на вычислительных узлах, содержащих центральный процессор (CPU) и графические ускорители (GPU). Производительность параллельных алгоритмов для классических схем метода сопряженных градиентов при совместном использовании CPU и GPU существенно ограничивается наличием точек синхронизации. В статье исследуется конвейерный вариант метода сопряженных градиентов с одной точкой синхронизации и возможностью распределения нагрузки между CPU и GPU при решении систем уравнений большой размерности. Численные эксперименты проведены на тестовых матрицах и вычислительных узлах разной производительности гетерогенного кластера, что позволило оценить вклад коммуникационных затрат. Алгоритмы реализованы при совместном использовании технологий MPI, OpenMP и CUDA. Предложенные алгоритмы помимо сокращения времени выполнения позволяют решать системы линейных уравнений и большего порядка, для которых не обеспечиваются необходимые ресурсы памяти одним GPU или вычислительным узлом. При этом конвейерный блочный алгоритм сокращает общее время выполнения за счет уменьшения точек синхронизации и объединения коммуникаций в одно сообщение.

Ключевые слова: параллельные вычисления, метод сопряженных градиентов, сокращение коммуникаций.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Недожогин Н.С., Копысов С.П., Новиков А.К. Параллельное решение систем линейных уравнений на гибридной архитектуре CPU+GPU // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 40–54. DOI: 10.14529/cmse200203.

Введение

Ускорители вычислений содержатся в вычислительных узлах суперкомпьютеров и применяются достаточно успешно при решении многих вычислительных задач, несмотря на то, что центральный процессор (CPU) простаивает после запуска функций-ядер на ускорителе. Существует еще несколько важных условий, для которых многообещающим представляется совместное использование CPU и ускорителей (например, GPU) при параллельных вычислениях в рамках одной задачи.

Каждая архитектура CPU и GPU обладает уникальными особенностями и ориентирована на решение тех или иных задач, для которых характерна, например, высокая производительность или низкая латентность. Гибридные узлы, содержащие и совместно использующие CPU+GPU, могут обеспечить эффективное решение более широкого круга задач или одной задачи, для которой меняются параллельные свойства алгоритмов и определив

*Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2020».

Работа выполнена при финансовой поддержке УдГУ в рамках конкурса грантов «Научный потенциал».

для ее выполнения то или другое исполнительное устройство. Отметим также высокую энергоэффективность гибридных вычислительных систем.

Совместное использование CPU+GPU связано с решением ряда задач: разделения вычислительной нагрузки между исполнительными устройствами с учетом производительности и пропускной способности памяти и ее размера; выявления параллельных свойств алгоритма на каждом шаге его выполнения и назначения устройства для выполнения.

Одной из трудоемких операций в численных методах является решение систем алгебраических уравнений (СЛАУ). В настоящее время предложено много параллельных алгоритмов, обеспечивающих высокую производительность и масштабируемость при решении больших разреженных систем уравнений на современных многопроцессорных архитектурах с иерархической структурой.

Построение гибридных решателей с комбинированием прямых и итерационных методов для решения СЛАУ позволяет использовать несколько уровней параллелизма [1–5]. Так в [6] построен и реализован гибридный метод решения систем уравнений дополнения Шура предобусловленными итерационными методами из подпространств Крылова при совместном использовании центральных процессоров (CPU) и графических ускорителей (GPU). При параллельном решении системы уравнений для дополнения Шура применялся классический предобусловленный метод сопряженных градиентов [7] для блочной упорядоченной матрицы и разделением вычислений при матричных операциях между CPU и одним или несколькими GPU. В настоящей работе рассматривается подход, сокращающий затраты на обмен данными между CPU и GPU за счет уменьшения числа точек глобальной синхронизации и конвейеризации вычислений.

Методы подпространства Крылова являются одними из наиболее эффективных вариантов решения крупномасштабных задач линейной алгебры. Однако, классические алгоритмы подпространства Крылова плохо масштабируются на современных архитектурах из-за наличия узких мест, связанных с синхронизацией вычислений. Конвейерные методы подпространства Крылова [8] со скрытыми коммуникациями обеспечивают высокую параллельную масштабируемость за счет перекрытия глобальных коммуникаций с вычислениями матрично-векторных операций и скалярных произведений векторов. Первые работы по сокращению коммуникаций были связаны с вариантом метода сопряженных градиентов (CG), имеющих одну коммуникацию на каждой итерации [9], с применением трехчленных рекуррентных соотношений CG [10].

Следующим этапом развития стало появление s-шаговых методов из подпространств Крылова [11], в которых итерационный процесс в s-блоке использует различные базисы подпространств Крылова. В результате удалось сократить число точек синхронизации до одной на s итераций. Однако, для большого числа процессоров (ядер) коммуникации все же могут занимать существенно больше времени, чем вычисление одного матрично-векторного произведения. В работе [12] предложен алгоритм CG, использующий вспомогательные вектора и перенос последовательной зависимости между вычислением матрично-векторного произведения и скалярными произведениями векторов. В данном подходе латентность коммуникаций заменяется дополнительными вычислениями.

Статья организована следующим образом. В разделе 1 рассмотрен конвейерный вариант метода сопряженных градиентов, приведен алгоритм и отличия от классического подхода. Раздел 2 посвящен вопросу совместного использования CPU и GPU, декомпозиции матрицы и обсуждению блочного варианта метода сопряженных градиентов. В разделе 3

приведены результаты численных экспериментов на тестовых матрицах из библиотеки университета Флорида. В заключении приводится краткая сводка результатов, полученных в ходе численных экспериментов, и указаны направления дальнейших исследований.

1. Конвейерный вариант метода сопряженных градиентов

Рассмотрим конвейерный вариант метода сопряженных градиентов, который математически эквивалентен классической форме предобусловленного метода CG и имеет такую же скорость сходимости.

Алгоритм 1: Конвейерный алгоритм метода CGwO

```
1  r = b − Ax
2  u = M^{-1} r
3  w = Au
4  γ_1 = (r, u)
5  δ = (w, u)
6  j = 0
   while ||r||_2/||b||_2 > ε do
7      m = M^{-1} w
8      n = Am
       if (j = 0) then
9          β = 0
       else
10         β = γ_1/γ_0
11     α = γ_1/(δ − βγ_1/α)
12     z = n + βz
13     w = w − αz
14     q = m + βq
15     s = w + βs
16     p = u + βp
17     x = x + αp
18     r = r − αs
19     u = u + αq
20     γ_0 = γ_1
21     γ_1 = (r, u); δ = (w, u)
22     j = j + 1
```

В этом алгоритме модификация векторов r_{j+1}, x_{j+1}, s_{j+1}, p_{j+1} и матрично-векторных произведений обеспечивает конвейерные вычисления. Вычисление скалярных произведений (строки 4, 5) может быть перекрыто с вычислением произведения на предобуславливатель (строка 2) и матрично–векторным произведением (строка 3). Однако, число триад в алгоритме увеличивается до восьми, в отличие от трех для классического варианта и четырех в [11]. В этом случае, возможно параллельное вычисление триад и двух скалярных произведений в начале итерационного процесса с одной точкой синхронизации.

Представленный в данной работе конвейерный вариант CG, может быть использован с любым предобуславливателем. Существуют два способа организации вычислений в предо-

бусловленном конвейерном CG, обеспечивающих компромисс между масштабируемостью и общим числом операций [13].

Таким образом, конвейерная схема CG, характеризуется: другой последовательностью вычислений; наличием глобальной коммуникации, которая может перекрываться с локальными вычислениями, такими как матрично-векторное произведение и операциями с предобуславливателем; возможностью организации асинхронных коммуникаций.

Выполнено сравнение двух вариантов метода сопряженных градиентов: классической схемы и конвейерной. В численных экспериментах (см. табл. 1) представлены результаты сравнения времени выполнения последовательного варианта классического CG и конвейерной схемы CGwO (алг. 1), исполняемых на CPU и GPU. Отметим, что в вариантах для GPU реализовывалось совместное вычисление всех скалярных произведений векторов в одной функции-ядре, независимо друг от друга. Для этого, при запуске ядра CUDA, задавалась размерность `Grid` иерархии нитей CUDA в двумерном виде: 3 набора блоков, каждый для выполнения вычислений над своей парой векторов. Это позволило сократить число обменов между памятью CPU и GPU, объединив все результирующие скаляры в одной коммуникации.

В тестовых расчетах использовались матрицы из коллекции университета Флориды (https://sparse.tamu.edu/). Вектор правых частей формировался, как построчная сумма элементов матрицы. Таким образом, решение системы $Ax = b$, размерности $N \times N$ (с числом ненулевых элементов nnz) представляет из себя вектор $x = (1, 1, \ldots, 1)^T$.

Для систем уравнений малой размерности время решения на CPU по классической схеме CG значительно меньше времени выполнения GPU при одном и том же числе итераций (см. табл. 1). Для больших систем затраты на синхронизации и пересылку между CPU и GPU перекрываются быстродействием GPU. В конвейерном варианте CGwO вычислительные затраты выполнения на GPU уменьшаются для всех рассмотренных систем уравнений практически трехкратно только за счет сокращения обменов между GPU и CPU при вычислении скалярных произведений.

2. Метод сопряженных градиентов при совместном использовании CPU и GPU

Рассмотрим применение Алгоритма 1 для параллельного решения сверхбольших систем уравнений на вычислительных узлах, каждый из которых содержит несколько CPU и GPU.

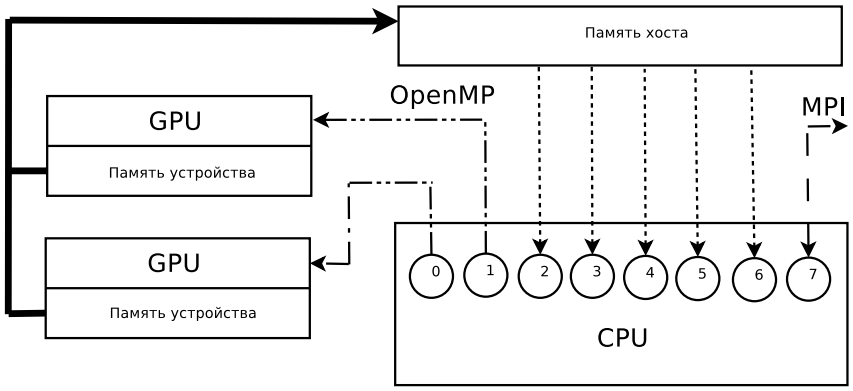

Рис. Гетерогенный вычислительный узел CPU+GPU

Для решения СЛАУ на нескольких GPU построим блочный алгоритм конвейерного метода сопряженных градиентов. Обмен данными между разными GPU в рамках одного вычислительного узла осуществляется с помощью технологии OpenMP, а обмен между разными вычислительными узлами — с помощью технологии MPI.

На рисунке представлена схема организации параллельных вычислений на гетерогенном вычислительном узле. Для примера рассмотрим узел, содержащий центральный восьмиядерный процессор и два графических ускорителя. Число OpenMP потоков выбирается по числу доступных ядер CPU. Первые два потока отвечают за обмен данными и запуск на двух GPU. Нити 2–6 обеспечивают вычисления на CPU и могут выполнять вычислительную работу над отдельным блоком матрицы СЛАУ. Последняя нить осуществляет MPI-коммуникации и обмен данными с другими вычислительными узлами.

2.1. Разделение матрицы

Для разделения матрицы A на блоки построим граф $G_A(V, E)$, где $V = \{i\}$ — множество вершин, связанных со строчным индексом матрицы (число вершин равно числу строк матрицы A); $E = \{(i, j)\}$ — множество ребер. Две вершины i и j считаются связанными, если в матрице A есть ненулевой элемент с индексами i и j. Полученный граф делится на блоки, число которых равно d. Например, для разделения графа можно использовать алгоритмом послойного разделения [14], который сокращает затраты на коммуникации за счет обменов только с двумя соседними узлами. После этого каждой вершине графа ставится в соответствие свой GPU или CPU. На каждом исполнительном устройстве вершины делятся на внутренние и граничные. Последние связаны хотя бы с одной вершиной, принадлежащей другому блоку.

После разделения каждый блок A_k исходной матрицы содержит в себе следующие подматрицы:

- $A_k^{[i_k,i_k]}$ — матрица связей между внутренними узлами;
- $A_k^{[i_k,b_k]}$, $A_k^{[b_k,i_k]}$ — матрицы связей между внутренними и граничными узлами;
- $A_k^{[b_k,b_l]}$ — матрица связи между граничными узлами k-го и l-го блоков.

Тогда матрица A может быть записана в следующем виде:

$$A = \begin{pmatrix} A_1^{[i_1,i_1]} & A_1^{[i_1,b_1]} & \cdots & 0 & 0 \\ A_1^{[b_1,i_1]} & A_1^{[b_1,b_1]} & \cdots & 0 & A_1^{[b_1,b_d]} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_d^{[i_d,i_d]} & A_d^{[i_d,b_d]} \\ 0 & A_d^{[b_d,b_1]} & \cdots & A_d^{[b_d,i_d]} & A_d^{[b_d,b_d]} \end{pmatrix}.$$

Используя полученное разделение, матрично-векторное произведение $n = Am$ разделим на две составляющие:

$$n_k^b = A_k^{[b_k,i_k]} m_k^i + \sum_{l=1}^{l \leqslant d} A_k^{[b_k,b_l]} m_l^b, \qquad n_k^i = A_k^{[i_k,i_k]} m_k^i + A_k^{[i_k,b_k]} n_k^b, \tag{1}$$

здесь k соответствует исполнительному устройству. Блочное представление векторов, участвующих в алгоритме, наследуется от разделения матрицы. Например, вектор m имеет вид $m^T = \left(m_1^i, m_1^b, \ldots, m_k^i, m_k^b, \ldots, m_d^i, m_d^b\right)$. Такая реализация матрично-векторного произведения уменьшает затраты на коммуникации между блоками на каждой итерации сопря-

женных градиентов. Для выполнения этой операции, требуется обмен векторами m_k^b, размер которых меньше размерности начального вектора m.

Разделение на блоки матрицы предобуславливателя M производится подобным образом.

2.2. Блочный конвейерный алгоритм

Используя блочное разделение матрицы и векторов, выполним распределение блоков матрицы на имеющиеся CPU и GPU. Число и размеры блоков позволяют распределять нагрузку в соответствии с производительностью имеющихся исполнительных устройств, в том числе и выделение нескольких блоков на одно устройство. Представим параллельную блочную схему метода CGwO, выполняемого каждым k-ом устройством в виде алг. 2, в котором выделены две параллельные ветви, выполняемые соответственно CPU и GPU/CPU.

Алгоритм 2: Блочный алгоритм CGwO выполняемый на k-ом устройстве

Data: Разделение матрицы на блоки $A_k^{[i_k,i_k]}$, $A_k^{[i_k,b_k]}$, $A_k^{[b_k,i_k]}$, $A_k^{[b_k,b_l]}$.

1	$r = b$	
2	$u = M^{-1}r$	
	`// Параллельно выполняемые ветви алгоритма`	
	`//` $(\texttt{CPU} \vee \texttt{GPU})_k$	`// CPU`
3	$w_k^i = A_k^{[i_k,i_k]} \cdot u_k^i + A_k^{[i_k,b_k]} \cdot u_k^b$	Сборка векторов u_k^b
4	$w_k^b = A_k^{[b_k,b_k]} \cdot u_k^b + A_k^{[b_k,i_k]} \cdot u_k^i$	$w_h^b = \sum_{l=1,l\neq k}^{l\leqslant d} A_k^{[b_k,b_l]} \cdot u_k^b$
5	$w_k^b = w_k^b + w_h^b$	Копирование w_h^b на $\texttt{GPU}_k$
6	$m = M^{-1}w$	Сборка векторов m_k^b
7	$\gamma_{1k} = (r_k, u_k)$; $\delta_k = (w_k, u_k)$	Сборка $\delta = \sum_k \delta_k; \gamma_1 = \sum_k \gamma_{1k}$
8	$j = 0$	
	while $\|\|r\|\|_2/\|\|b\|\|_2 > \varepsilon$ **do**	
9	$n_k^i = A_k^{[i_k,i_k]} \cdot m_k^i + A_k^{[i_k,b_k]} \cdot m_k^b$	$n_h^b = \sum_{l=1,l\neq k}^{l\leqslant d} A_k^{[b_k,b_l]} \cdot m_k^b$;
10	$n_k^b = A_k^{[b_k,b_k]} \cdot m_k^b + A_k^{[b_k,i_k]} \cdot m_k^i$	Копирование n_h^b на GPU_k;
11	$n_k^b = n_k^b + n_h^b$	$\beta = ((j = 0)\,?\,0 : \gamma_1/\gamma_0)$;
12	$z = n + \beta z$	$\alpha = \gamma_1/(\delta - \beta\gamma_1/\alpha)$;
13	$w = w - \alpha z$	
14	$q = m + \beta q$	
15	$s = w + \beta s$	Сборка векторов w_k^b;
16	$p = u + \beta p$	
17	$x = x + \alpha p$	
18	$r = r - \alpha s$	Сборка векторов m_k^b;
19	$u = u + \alpha q$	Сборка $\delta = \sum_k \delta_k; \gamma_1 = \sum_k \gamma_{1k}$;
20	$m = M^{-1}w$	
21	$\gamma_0 = \gamma_1$	
22	$\gamma_{1k} = (r_k, u_k)$; $\delta_k = (w_k, u_k)$	
23	$j = j + 1$	

Операции, выполняемые параллельно, записаны в одной строке алгоритма. Векторные операции на каждом исполняющем устройстве происходят в два этапа, для внутренних и граничных узлов. Обозначения внутренних и граничных узлов для векторов опущены, за

исключением матрично-векторного произведения. Скалярные произведения векторов выполняются независимо каждым исполняющим устройством над своими частями векторов. Суммирования промежуточных скаляров происходит в параллельных потоках, отвечающих за коммуникации, что является точкой синхронизации на каждой итерации алгоритма. В блочном CGwO по сравнению с алг. 1 был перенесен шаг предобуславливания (строка 7 в строку 20). Это сделано для того, чтобы совместить векторные операции на исполняющем устройстве и сборку вектора правых частей для выполнения матрично-векторного умножения в предобуславливании. В строке 11 справа используется тернарный оператор: если $j = 0$, то $\beta = 0$, в других случаях $\beta = \gamma_1/\gamma_0$. Нижний индекс h в алгоритме применяется для векторов, которые хранятся только в памяти CPU.

Численные эксперименты по применению алг. 2 проводились при различных вариантах конфигурации вычислительных узлов, содержащих несколько CPU и GPU. В самом общем случае, параллельные вычисления на нескольких гетерогенных вычислительных узлах, содержащих один и более CPU и несколько GPU, реализуются с помощью сочетания технологий: MPI, OpenMP и CUDA. Рассмотрим организацию вычислений на примере кластера, в составе которого имеется два вычислительных узла (8 ядер CPU и 2 GPU). Каждому вычислительному узлу ставится в соответствие параллельный процесс MPI. В параллельном процессе порождается 9 параллельных потоков OpenMP, что на один больше, чем доступные ядра CPU. Восьмой поток OpenMP отвечает за коммуникации между различными вычислительными узлами средствами технологии MPI (сборка векторов с помощью функции `Allgatherv`, сложение скаляров `Allreduce`) и различными GPU. В алг. 2 операции, выполняемые этим потоком, представлены справа. Нулевой и первый потоки OpenMP связываются с одним из доступных GPU-устройств и отвечают за пересылку данных между GPU-CPU (вызовы функций асинхронного копирования) и вспомогательные вычисления. Каждое доступное GPU-устройство (в дальнейшем рассматривается как исполняющее устройство) связывается с одним из параллельных потоков OpenMP, который отвечает за пересылку данных между GPU-CPU (вызовы функций асинхронного копирования) и участвует с восьмым потоком в операции матрично-векторного умножения на граничных узлах (строки 4, 9 правая колонка). Оставшиеся параллельные потоки (второй–седьмой) производят вычисления как отдельное исполняющее устройство для своего блока матрицы. Операции, выполняемые исполняющими устройствами в алг. 2 приведены слева.

Применение предобуславливателя строки 2, 6 и 20 подразумевает использование блочного матрично-векторного произведения вида (1) рассмотренного выше.

3. Численные эксперименты

Вычислительные эксперименты выполнялись на вычислительных узлах (ВУ) нескольких типов кластера «Уран» суперкомпьютерного центра ИММ УрО РАН (СКЦ ИММ УрО РАН).

Используемые вычислительные узлы имеют следующие характеристики:

- раздел «debug»: 4 узла tesla [31–32, 46–47] по два 8-ядерных процессора Intel Xeon E5-2660 (2,2 ГГц), оперативная память — 96 ГБ, 8 GPU Nvidia Tesla M2090 (6 ГБ графической памяти), коммуникационная сеть — Gigabit Ethernet (1 Гбит/с).
- раздел «tesla-v100»: два 18-ядерных процессора Intel Xeon Gold 6240 CPU (2,60 ГГц); оперативная память — 384 ГБ; 8 GPU Nvidia Tesla V100 (32 ГБ графической памяти).

- раздел «tesla [21–30]»: 10 узлов по два 6-ядерных процессора Intel Xeon X5675 (3,07 ГГц), оперативная память — 192 ГБ, 8 GPU Nvidia Tesla M2090 (6 ГБ графической памяти), коммуникационная сеть — Infiniband (20 Гбит/с).
- раздел «tesla [33–45]»: 13 узлов по два 8-ядерных процессора Intel Xeon E5-2660 (2,2 ГГц), оперативная память — 96 ГБ, 8 GPU Nvidia Tesla M2090 (6 ГБ графической памяти), коммуникационная сеть — Infiniband (20 Гб/с).
- раздел «tesla [48–52]»: 5 узлов по два 8-ядерных процессора Intel Xeon E5-2650 (2,6 ГГц), оперативная память — 64 ГБ, 3 GPU Nvidia Tesla K40m (12 ГБ графической памяти), коммуникационная сеть — Infiniband (20 Гб/с).

Таблица 1

Время решения алгоритмом CG на CPU и GPU, сек

Матрица	N	nnz	# итераций	ВУ	Время, сек	
					CG	CGwO
Plat362	362	5786	991	M2090	6,88E-01	**3,07E-01**
				K40m	4,13E-01	3,12E-01
1138_bus	1138	4054	717	M2090	3,81E-01	**1,84E-01**
				K40m	5,31E-01	2,01E-01
				debug	6,82E-01	1,90E-01
Muu	7102	170134	12	M2090	2,64E-01	4,68E-03
				K40m	3,31E-01	**4,55E-03**
Kuu	7102	340200	378	M2090	4,31E-01	**1,31E-01**
				K40m	4,39E-01	1,35E-01
Pres_Poisson	14822	715804	661	M2090	6,72E-01	3,13E-01
				K40m	6,346E-01	**2,73E-01**
Inline_1	503712	36816342	5642	M2090	4,74E+01	5,17E+01
				K40m	**3,06E+01**	3,37E+01
Fault_639	638802	28614564	4444	M2090	3,83E+01	4,32E+01
				K40m	**2,44E+01**	2,77E+01
				debug	2,44E+01	2,77E+01
thermal2	1228045	8580313	2493	M2090	1,35E+01	1,82E+01
				K40m	**8,33E+00**	1,18E+01
G3_circuit	1585478	7660826	592	M2090	3,43E+00	4,32E+00
				K40m	**1,94E+00**	2,92E+00
Quenn_4147	4147110	399499284	8257	M2090	5,46E+02	5,78E+02
				K40m	**3,55E+02**	3,75E+02

Результаты сравнения двух алгоритмов метода сопряженных градиентов на системах уравнений, содержащих тестовые матрицы, представлены в табл. 1. Результаты приведены для нескольких типов вычислительных узлов при использовании одного графического ускорителя. Матрицы упорядочены по увеличению порядка системы уравнений (N) и числа ненулевых элементов (nnz). Жирным выделено лучшее время решения системы в каждом случае.

Конвейерный алгоритм CGwO показал сокращение времени выполнения на небольших СЛАУ, для которых характерна небольшая вычислительная нагрузка, за счет чего сокращение коммуникаций обеспечивает меньшие временные затраты. Отметим, что классиче-

ский алгоритм CG был реализован на основе CUBLAS, а в варианте CGwO применяются матричные и векторные операции собственной GPU-реализации.

Для систем **Inline_1** и **Fault_639** время выполнения конвейерного алгоритма на 10 и 13,5 % больше относительно блочного варианта CG, что связано с выполнением дополнительных векторных операций, которые не перекрываются сокращением коммуникаций. С уменьшением числа итераций время выполнения алгоритмов CG и CgwO на одном GPU возрастает незначительно. Так, например, для матриц **G3_circuit** и **thermal2** с примерно равным числом уравнений значительно различаются результаты. Для системы с матрицей **thermal2** увеличение затрат становится более существенным, в сравнении с **G3_circuit**.

В табл. 2 и 3 представлены результаты выполнения блочных вариантов алгоритмов при вычислениях на нескольких вычислительных узлах. Здесь обозначение #CPU/#GPU означает число используемых графических ускорителей #GPU на каждом вычислительном узле, число которых равно #CPU. Как отмечалось ранее, при выполнении алгоритма на одном вычислительном узле с двумя ускорителями (например, 1CPU/2GPU) коммуникации обеспечивались только с помощью технологии OpenMP. В случае 8CPU/1GPU — на восьми вычислительных узлах при задействовании одного GPU на каждом, обмены осуществлялись на основе MPI.

Таблица 2

Время решения блочным алгоритмом CG на CPU/GPU, сек

Матрица/ВУ	BlockCG/число блоков			
	2	3	8	
	#CPU/#GPU			
	2/1	3/1	4/2	8/1
Plat362/M2090	**1,55E+00**	—	—	—
/K40m	1,92E+00	1,56E+00	—	—
1138_bus/M2090	1,84E+00	—	—	—
/K40m	1,90E+00	1,85E+00	—	—
/debug	**1,25E+01**	—	—	—
Muu/M2090	6,12E-01	—	1,84E+00	7,4E-01
/K40m	6,59E-01	**5,64E-01**	6,97E+00	—
Kuu/M2090	**1,30E+00**	—	1,35E+00	1,41E+00
/K40m	1,29E+00	1,36E+00	1,94E+00	—
Pres_Poisson/M2090	**1,55E+00**	—	1,57E+00	2,0E+00
/K40m	1,60E+00	1,66E+00	2,3E+00	—
Inline_1/M2090	3,76E+01	—	—	—
/K40m	2,66E+01	**2,20E+01**	—	—
Fault_639/M2090	3,18E+01	—	**1,55E+01**	2,09E+01
/K40m	2,20E+01	1,98E+01	2,89E+01	—
/debug	9,38E+01	—	—	—
thermal2/M2090	1,46E+01	—	9,77E+00	1,15E+01
/K40m	1,18E+01	**1,00E+01**	—	—
G3_circuit/M2090	4,27E+00	—	4,82E+00	**3,26E+00**
/K40m	4,04E+00	3,510E+00	4,06E+00	—
Quenn_4147/M2090	1,33E+02	1,55E+02	—	—

Значительное влияние характеристик сети на производительность блочных методов можно увидеть в табл. 2 и 3 для систем уравнений с матрицами **Fault_639** и **1138_bus**. Вычисления для этих систем уравнений выполнялись на различных вычислительных узлах (ВУ) с разной пропускной способностью и латентностью коммуникационной сети. При численных экспериментах на ВУ (раздел «debug»), соединенных коммуникационной сетью Gigabit Ethernet, затраты на коммуникации значительно увеличивают время выполнения алгоритма CG. Например, в варианте **1138_bus** на аппаратном разделе «debug» время выполнения конвейерного алгоритма в 3,6 раза меньше (строка «debug» в табл. 2 и 3 и любая строка в табл. 1). Использование коммуникационной сети Infiniband 20 Гб/с позволяет сократить время выполнения для всех представленных систем уравнений (строки «M2090» и «K40m»).

Сравнение строк таблиц, соответствующих «K40m» и «M2090», показывает, что использование GPU нового поколения обеспечивает существенное уменьшение времени выполнения. Особенно это заметно на системах больших размерностей **thermal2**, **G3_circuit**, где использование ускорителя Tesla K40m позволяет получить ускорение до 2,5 раз в сравнении с Tesla M2090 (в 2,5 раза в табл. 1 для системы **thermal2**, для остальных матриц в среднем в 1,5 раза быстрее).

При сокращении вычислительной нагрузки более явно сказывается уменьшение числа точек синхронизации и объединения пересылок за одну транзакцию. Это показывает сравнение систем с матрицами **Kuu** с **Muu**. При равном числе уравнений и ненулевых элементов обусловленность этих матриц существенно отличается и, как следствие, число итераций в методе сопряженных градиентов различно. Из табл. 1 видно, что использование конвейерного алгоритма для матрицы **Muu** дает ускорение в 70 раз по сравнению с матрицей **Kuu**, где ускорение только 2,8.

На матрицах большой размерности **Inline_1**, **Fault_639**, **thermal2** и **G3_circuit** ускорения не наблюдается, потому что сокращение коммуникаций не перекрывает затраты на дополнительные векторные операции (табл. 1). Использование блочных алгоритмов сокращает вычислительную нагрузку на один GPU, тем самым позволяет при некотором разделении матрицы и векторов (размерности ≈ 200000 на каждом GPU) получить ускорении вычислений. Например, для системы **Fault_639** порядка 638802 достигается невысокое ускорение (в 1,2–1,5 раза) при разделении на три блока, для системы **Inline_1** — на два блока.

Анализ результатов показал, что уменьшение объема данных за счет разделения матрицы и сокращение точек синхронизации незначительно снижают влияние коммуникационных затрат на общую производительность алгоритма. Только использование ВУ, соединенных Infiniband позволило получить ускорение при вычислениях на нескольких вычислительных узлах. Разделение матрицы на блоки позволило сократить время выполнения конвейерного блочного алгоритма по сравнению с классическим, выполняемом на одном узле, на матрицах **Inline_1**, **Fault_639** за счет сокращения вычислительной нагрузки, приходящейся на один графический ускоритель.

На системах большого порядка **thermal2**, **G3_circuit**, решаемых блочным вариантом алгоритма CGwO, сокращение коммуникаций и точек синхронизации так же не перекрывают возрастающие затраты на дополнительные векторные операции.

В табл. 4 представлены результаты по ускорению блочного конвейерного алгоритма метода сопряженных градиентов при разделении на большее число подобластей (12 и 16)

Таблица 3

Время решения блочным конвейерным алгоритмом CGwO на CPU/GPU, сек

Матрица/ВУ	BlockCGwO/число блоков			
	2	3	8	
	#CPU/#GPU			
	2/1	3/1	4/2	8/1
Plat362/M2090	**1,22E+00**	—	—	—
/K40m	1,28E+00	1,31E+00	—	
1138_bus/M2090	**9,28E-01**	—	—	—
/K40m	1,03E+00	1,04E+00	—	—
/debug	5,36E+00	—	—	—
Muu/M2090	2,29E-01	—	5,8E-01	**1,8E-01**
/K40m	2,89E-01	2,88E-01	—	—
Kuu/M2090	**6,43E-01**	—	0,80E+00	7,9E-01
/K40m	6,81E-01	7,95E-01	1,26E+00	—
Pres_Poisson/M2090	9,57E-01	—	1.08E+00	**9,5E-01**
/K40m	1,02E+00	1,19E+00	1,76E+00	—
Inline_1/M2090	3,65E+01	—	—	—
/K40m	2,48E+01	**1,97E+01**	—	—
Fault_639/M2090	3,03E+01	—	**1,49E+01**	1,69E+01
/K40m	2,05E+01	1,78E+01	2,13E+01	—
/debug	5,25E+01	—	—	—
thermal2/M2090	1,49E+01	—	9,33E+00	**8,92E+00**
/K40m	1,18E+01	9,65E+00	—	—
G3_circuit/M2090	3,99E+00		4,78E+00	**2,69E+00**
/K40m	3,27E+00	2,77E+00	3,86E+00	
Quenn_4147/M2090				1,4E+02

и вычислениях на узлах с GPU Nvidia Tesla M2090. Ускорение считалось относительно варианта на одном GPU из табл. 1. Все эксперименты производились при эксклюзивном использовании вычислительного узла, но не сети.

Таблица 4

Ускорение алгоритма CGwO

Матрица	Число блоков					
	12			16		
	#CPU/#GPU					
	2/6	3/4	12/1	2/8	4/4	16/1
Kuu	0,06	0,03	0,21	0,045	0,087	0,22
Inline_1	3,19	3,41	3,51	3,09	3,40	3,80
Fault_639	3,09	3,23	4,10	2.94	3,29	2,11
thermal2	1,56	1,95	2,15	1,50	1,55	2,22
Quenn_4147	4,73	5,01	5,07	4,77	5,50	5,34

Из представленных результатов видно, что использование многих узлов с одним GPU или небольшого числа узлов с несколькими GPU (не более 4-х на узел) дает примерно равное ускорение параллельного алгоритма. При использовании 8 ускорителей на узле ускорение снижается незначительно (7 %). Помимо этого большое число подобластей матрицы увеличивает затраты на коммуникации, которые не перекрываются за счет сокращения времени выполнения матричных и векторных операций. При разделении на 16 подобластей приложение выполняется на вычислительных узлах, содержащих CPU разных поколений (Intel Xeon X5675 и Intel Xeon E5-2660), что приводит к уменьшению ускорения для матриц **Quenn_4147** и **Fault_639**.

Заключение

Гибридные вычислительные узлы, содержащие и совместно использующие CPU+GPU позволяют обеспечить эффективное решение более широкого круга задач или одной задачи, для которой меняются параллельные свойства алгоритмов, и назначить на выполнение то или другое исполнительное устройство. Отметим также высокую энергоэффективность гибридных вычислительных систем в тех случаях, когда равномерно загружены центральные процессоры и графические ускорители вычислений.

В работе рассмотрена параллельная реализация решения систем линейных алгебраических уравнений на вычислительных узлах, содержащих центральный процессоры и графические ускорители. При совместном использовании CPU и GPU производительность параллельных алгоритмов для классических схем метода сопряженных градиентов существенно ограничивается наличием точек синхронизации. Предложен конвейерный вариант метода сопряженных градиентов с одной точкой синхронизации, возможностью асинхронных вычислений, распределения нагрузки между несколькими GPU, находящимися, как на одном вычислительном узле, так и для кластера GPU при решении систем уравнений большой размерности. Для дальнейшего увеличения эффективности вычислений предполагается исследовать не только коммуникационную нагрузку алгоритмов, но и распределение вычислительной нагрузки между CPU и GPU. Для получения более надежных временных оценок коммуникаций необходимо проведение серии вычислительных экспериментов на вычислительных системах с полностью монопольным режимом работы с большим числом гетерогенных узлов.

Из анализа полученных в ходе численных экспериментов данных можно сделать следующие выводы: использование конвейерного алгоритма снижает коммуникационные затраты, но увеличивает вычислительные. Для систем небольших размеров или с небольшим числом итераций это сокращает время выполнения алгоритма при использовании одного GPU. Для систем больших размерностей, сокращение времени выполнения в сравнении с CG, возможно только при разбиении матрицы на подобласти достаточно малой размерности, при котором уменьшение коммуникаций перекрывают увеличенные вычислительные затраты.

Предложенные блочные алгоритмы, помимо сокращения времени выполнения, позволяют решать системы линейных уравнений и большего порядка, для которых не обеспечиваются необходимые ресурсы памяти одним GPU или вычислительным узлом. При этом, конвейерный блочный алгоритм сокращает общее время выполнения за счет уменьшения точек синхронизации и объединения коммуникаций в одно сообщение.

Литература

1. Agullo E., Giraud L., Guermouche A., Roman J. Parallel hierarchical hybrid linear solvers for emerging computing platforms // Comptes Rendus Mecanique. 2011. Vol. 333. P. 96–103. DOI: 10.1016/j.crme.2010.11.005.
2. Gaidamour J., Henon P. A parallel direct/iterative solver based on a Schur complement approach // 11th IEEE International Conference on Computational Science and Engineering (San Paulo, Brazil, July, 16–18, 2008). IEEE. 2008. P. 98–105. DOI: 10.1109/CSE.2008.36.
3. Giraud L., Haidar A., Saad Y. Sparse approximations of the Schur complement for parallel algebraic hybrid solvers in 3D // Numerical Mathematics. 2010. Vol. 3, no. 3. P. 276–294. DOI: 10.4208/nmtma.2010.33.2.
4. Rajamanickam S., Boman E.G., Heroux M.A. ShyLU: A Hybrid-Hybrid Solver for Multicore Platforms // IEEE 26th International Parallel and Distributed Processing Symposium (Shanghai, China, May, 21–25, 2012). 2012. P. 631–643. DOI: 10.1109/IPDPS.2012.64.
5. Yamazaki I., Rajamanickam S., Boman E., Hoemmen M., Heroux M., Tomov S. Domain decomposition preconditioners for communication-avoiding Krylov methods on a hybrid CPU/GPU cluster // Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC14). 2014. P. 933–944. DOI: 10.1109/SC.2014.81.
6. Kopysov S., Kuzmin I., Nedozhogin N., Novikov A., Sagdeeva Y. Scalable hybrid implementation of the Schur complement method for multi-GPU systems // The Journal of Supercomputing. 2014. Vol. 69. P. 81–88. DOI: 10.1007/s11227-014-1209-7.
7. Hestenes M.R., Stiefel E. Methods of Conjugate Gradients for Solving Linear Systems // Journal of Research of the National Bureau of Standards. 1952. Vol. 49. P. 409–436. DOI: 10.6028/jres.049.044.
8. Cornelis J., Cools S., Vanroose W. The Communication-Hiding Conjugate Gradient Method with Deep Pipelines // CoRR abs/1801.04728. 2018. https://arxiv.org/abs/1801.04728.
9. D'Azevedo E.F., Romine C.H. Reducing communcation costs in the conjugate gradient algorithm on distributed memory multiprocessors // Technical Report ORNL/TM-12192, Oak Ridge National Lab, 1992. DOI: 10.2172/10176473.
10. Wilkinson J.H., Reinsch C. Linear Algebra. Springer, Berlin, Heidelberg, 1971. 441 p. DOI: 10.1007/978-3-662-39778-7
11. Chronopoulos A.T., Gear C.W. s-step iterative methods for symmetric linear systems // Journal of Computational and Applied Mathematics. 1989. Vol. 25, no. 2. P. 153–168. DOI: 10.1016/0377-0427(89)90045-9.
12. Ghysels P., Vanroose W. Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm // Parallel Computing. 2014. Vol. 40, no. 7. P. 224–238. DOI: 10.1016/j.parco.2013.06.001.
13. Gropp W. Update on libraries for blue waters. Bordeaux. France. 2010. URL: http://jointlab.ncsa.illinois.edu/events/workshop3/pdf/presentations/Gropp-Update-on-Libraries.pdf (дата обращения: 07.11.2019).
14. Кадыров И.Р., Копысов С.П., Новиков А.К. Разделение триангулированной многосвязной области на подобласти без ветвления внутренних границ // Ученые записки Казан-

ского университета. Серия: Физико-математические науки. 2018. Т. 160. № 3. С. 544–560. DOI: 10.26907/2541-7746.

Недожогин Никита Сергеевич, ст. преподаватель, кафедра вычислительной механики, Удмуртский государственный университет (Ижевск, Российская Федерация)

Копысов Сергей Петрович, д.ф.-м.н., профессор, кафедра вычислительной механики, Удмуртский государственный университет (Ижевск, Российская Федерация)

Новиков Александр Константинович, к.ф.-м.н., доцент, кафедра вычислительной механики, Удмуртский государственный университет (Ижевск, Российская Федерация)

DOI: 10.14529/cmse200203

PARALLEL SOLVING OF LINEAR EQUATIONS SYSTEMS ON HYBRID ARCHITECTURE CPU+GPU

© 2020 N.S. Nedozhogin, S.P. Kopysov, A.K. Novikov

Udmurt State University (Universitetskaya 1, Izhevsk, 426034 Russia)

E-mail: nedozhogin07@gmail.com, s.kopysov@gmail.com, sc_work@mail.ru

Received: 29.01.2020

The article discusses the parallel implementation of solving systems of linear algebraic equations on computational nodes containing a central processing unit (CPU) and graphic accelerators (GPU). The performance of parallel algorithms for the classical conjugate gradient method schemes when using the CPU and GPU together is significantly limited by the synchronization points. The article investigates the pipeline version of the conjugate gradient method with one synchronization point, the possibility of asynchronous calculations, load balancing between the CPU and GPU when solving the large linear systems. Numerical experiments were carried out on test matrices and computational nodes of different performance of a heterogeneous cluster, which allowed us to estimate the contribution of communication costs. The algorithms are implemented with the joint use of technologies: MPI, OpenMP and CUDA. The proposed algorithms, in addition to reducing the execution time, allow solving large linear systems, for which there are not enough memory resources of one GPU or a computing node. At the same time, block algorithm with the pipelining decreases the total execution time by reducing synchronization points and aggregating some messages in one.

Keywords: parallel calculations, the method of conjugate gradients, reduction of communications.

FOR CITATION

Nedozhogin N.S., Kopysov S.P., Novikov A.K. Parallel Solving of Linear Equations Systems on Hybrid Architecture CPU+GPU. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 2. P. 40–54. (in Russian) DOI: 10.14529/cmse200203.

References

1. Agullo E., Giraud L., Guermouche A., Roman J. Parallel hierarchical hybrid linear solvers for emerging computing platforms. Comptes Rendus Mecanique. 2011. Vol. 333. P. 96–103. DOI: 10.1016/j.crme.2010.11.005.

2. Gaidamour J., Henon P. A parallel direct/iterative solver based on a Schur complement

approach. 11th IEEE International Conference on Computational Science and Engineering (San Paulo, Brazil, July, 16–18, 2008). IEEE. 2008. P. 98–105. DOI: 10.1109/CSE.2008.36.

3. Giraud L., Haidar A., Saad Y. Sparse approximations of the Schur complement for parallel algebraic hybrid solvers in 3D. Numerical Mathematics. 2010. Vol. 3, no. 3. P. 276–294. DOI: 10.4208/nmtma.2010.33.2.

4. Rajamanickam S., Boman E.G., Heroux M.A. ShyLU: A Hybrid-Hybrid Solver for Multicore Platforms. IEEE 26th International Parallel and Distributed Processing Symposium (Shanghai, China, May, 21–25, 2012). 2012. P. 631–643. DOI: 10.1109/IPDPS.2012.64.

5. Yamazaki I., Rajamanickam S., Boman E., Hoemmen M., Heroux M., Tomov S. Domain decomposition preconditioners for communication-avoiding Krylov methods on a hybrid CPU/GPU cluster. Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC14). 2014. P. 933–944. DOI: 10.1109/SC.2014.81.

6. Kopysov S., Kuzmin I., Nedozhogin N., Novikov A., Sagdeeva Y. Scalable hybrid implementation of the Schur complement method for multi-GPU systems. The Journal of Supercomputing. 2014. Vol. 69. P. 81–88. DOI: 10.1007/s11227-014-1209-7.

7. Hestenes M.R., Stiefel E. Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards. 1952. Vol. 49. P. 409—436. DOI: 10.6028/jres.049.044.

8. Cornelis J., Cools S., Vanroose W. The Communication-Hiding Conjugate Gradient Method with Deep Pipelines. CoRR abs/1801.04728. 2018. https://arxiv.org/abs/1801.04728.

9. D'Azevedo E.F., Romine C.H. Reducing communcation costs in the conjugate gradient algorithm on distributed memory multiprocessors. Technical Report ORNL/TM-12192, Oak Ridge National Lab, 1992. DOI: 10.2172/10176473.

10. Wilkinson J.H., Reinsch C. Linear Algebra. Springer, Berlin, Heidelberg, 1971. 441 p. DOI: 10.1007/978-3-662-39778-7

11. Chronopoulos A.T., Gear C.W. s-step iterative methods for symmetric linear systems. Journal of Computational and Applied Mathematics. 1989. Vol. 25, no. 2. P. 153–168. DOI: 10.1016/0377-0427(89)90045-9.

12. Ghysels P., Vanroose W. Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm. Parallel Computing. 2014. Vol. 40, no. 7. P. 224–238. DOI: 10.1016/j.parco.2013.06.001.

13. Gropp W. Update on libraries for blue waters. Bordeaux. France. 2010. Available at: http://jointlab.ncsa.illinois.edu/events/workshop3/pdf/presentations/Gropp-Update-on-Libraries.pdf (accessed: 07.11.2019).

14. Kadyrov I.R., Kopysov S.P., Novikov A.K. Partitioning of triangulated multiply connected domain into subdomains without branching of inner boundaries. Uchenye Zapiski Kazanskogo Universiteta. Seriya Fiziko-Matematicheskie Nauki. 2018. Vol. 160, no. 3. P. 544–560. (in Russian) DOI: 10.26907/2541-7746.

УДК 519.876.5, 519.725 **DOI: 10.14529/cmse200204**

О ДЕКОДЕРЕ МЯГКИХ РЕШЕНИЙ ДВОИЧНЫХ КОДОВ РИДА—МАЛЛЕРА ВТОРОГО ПОРЯДКА

© 2020 В.М. Деундяк[1,2], Н.С. Могилевская[2]

[1]*ФГНУ НИИ «Спецвузавтоматика»*

(344002 Ростов-на-Дону, пер. Газетный, д. 51),

[2]*Южный федеральный университет*

(344090 Ростов-на-Дону, ул. Мильчакова, д. 8а)

E-mail: vl.deundyak@gmail.com, nadezhda.mogilevskaia@yandex.ru

Поступила в редакцию: 22.11.2019

Построена общая модель помехоустойчивого двоичного канала передачи данных, предназначенная для использования с различными декодерами мягких решений. Линия связи, рассматриваемая в модели, является дискретной по входу и непрерывной по выходу. На ее вход подаются дискретные сигналы из мультипликативного двоичного алфавита, а в результате непреднамеренных помех, возникающих в линии связи, на выходе после фильтрации формируются символы из мультипликативной группы поля вещественных чисел, которые затем подаются на вход декодера помехоустойчивого кода. Мягкие и вероятностные декодеры позволяют исправлять большее количество ошибок в кодовых словах, чем гарантируется минимальным расстоянием используемого помехоустойчивого кода. В работе рассмотрен вероятностный декодер мягких решений Сидельникова—Першакова для кодов Рида—Маллера второго порядка в модификации, предложенной П. Лоидрю и Б. Саккуром. Ранее эффективность этих декодеров была подтверждена с помощью имитационных экспериментов, но теоретическое обоснование отсутствовало. В настоящей работе сформулировано требование к каналу связи, названное гладкостью канала, при выполнении которого теоретически доказана корректность этого декодера в случае, когда количество ошибок в каждом кодовом слове ограничено половиной кодового расстояния. В основе доказательства лежит использование теории квадратичных форм и методов дифференциального исчисления в кольце полиномов нескольких переменных над полями Галуа.

Ключевые слова: коды Рида—Маллера, декодер, модель канала, доказательство корректности декодера.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Деундяк В.М., Могилевская Н.С. О декодере мягких решений двоичных кодов Рида—Маллера второго порядка // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 55–67. DOI: 10.14529/cmse200204.

Введение

В.М. Сидельниковым и А.С. Першаковым в [10] предложен вероятностный мягкий декодер для двоичных кодов Рида—Маллера второго порядка (далее СП-декодер). Напомним, что по сравнению с классическими детерминированными декодерами вероятностные и мягкие декодеры исправляют бо́льшее количество ошибок, но при этом обладают большей сложностью. Вероятностные декодеры, как правило, гарантировано исправляют ошибки в пределах половины кодового расстояния, а при дальнейшем увеличении числа ошибок делают это с некоторой вероятностью. Декодеры мягких решений получают на вход данные из канала без демодуляции [9], за счет чего не накапливаются ошибки квантования, и в силу этого качество декодирования растет.

В [1] была предложена модификация СП-декодера (далее СПМ-декодер), авторы которой отказались от использования некоторых параметров СП-декодера, тем самым понизив его сложность, от $O(\log_2 n + hs^3)n^2$ до $O(n^2 \log_2 n)$, где n — длина кода Рида—

Маллера, а s и h — параметры СП-декодера. СП и СПМ декодеры представлены их авторами без обоснования, однако корректирующая способность этих декодеров для дискретных ошибок подтверждена экспериментально (см., например, [1, 8]).

Декодеры СП и СПМ построены для линии связи, на выходе которой вместо входного бинарного алфавита появляются вещественные числа. В обоих декодерах на первом шаге фактически вычисляется производный вектор в мультипликативном виде, однако в соответствующих формулах вместо операции деления используется операция умножения, что, в конечном счете, способствует ухудшению процесса декодирования.

В настоящей работе построена модификация алгоритмов СП и СПМ, в которой производный вектор в мультипликативном виде вычисляется правильно с применением операции деления. Найдено условие для двоичного канала связи, названное гладкостью канала, при выполнении которого доказана корректность этого декодера кодов Рида—Маллера в случае, если число ошибок, повредившее кодовое слово, не превышает половины кодового расстояния.

Статья организована следующим образом. В разделе 1 построена общая модель помехоустойчивого двоичного канала передачи данных, предназначенная для использования с различными декодерами мягких решений, в которой оператор приемника фильтрует входные сигналы из R таким образом, чтобы при вычислении производного вектора не возникало проблемы деления на ноль. Раздел 2 содержит необходимые сведения о кодах Рида—Маллера и описание некоторых их свойств. В разделе 3 разработана модификация СПМ-декодера, предназначенная для использования в канале, описываемом построенной моделью. В разделе 4 сформулировано условие для двоичного канала связи, названное гладкостью канала, при выполнении которого доказана корректность построенного декодера кодов Рида—Маллера в случае, если число ошибок, повредившее кодовое слово, не превышает половины кодового расстояния. В заключении приводится краткая сводка результатов, полученных в работе и указаны направления дальнейших исследований.

1. Модель бинарного канала передачи данных

Рассмотрим модель бинарного канала передачи данных (см. рис. 1). Источник сообщений генерирует информационные векторы $\bar{m} = (m_1, m_2, \ldots, m_k)$, $\bar{m} \in F_2^k$, где F_2^k — линейное k-мерное пространство над полем Галуа F_2. Сгенерированные векторы поступают на вход кодера канала, где они кодируются с помощью некоторого двоичного линейного блочного кода длины n размерности $k(< n)$, для которого известен декодер мягких решений. На выходе кодера канала вырабатываются кодовые векторы $\bar{c}$ из пространства F_2^n. В качестве оператора кодера канала рассмотрим отображение $Cod: F_2^k \to F_2^n$.

Из кодера данные попадают на вход передатчика, который трансформирует символы кодовых векторов в сигналы, пригодные для передачи в линии связи. Именно, с помощью изоморфизма аддитивной группы поля F_2 на мультипликативную группу $C_2 = \{e^{i\pi j)}\}_{j\in F_2} = \{1; -1\}$:

$$\phi: F_2 \to C_2,\ \phi(j) = e^{i\pi j} = (-1)^j,\ j \in F_2,$$

передатчик в модели преобразовывает кодовые векторы $\bar{c} \in F_2^n$ в векторы $\bar{z} = (z_1, z_2, \ldots, z_n)(\in C_2^n)$.

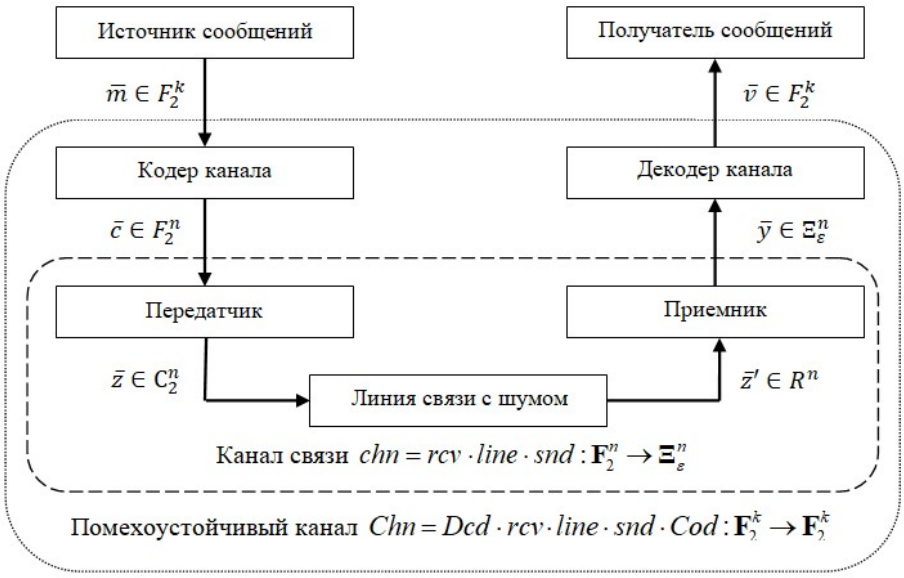

Рис.1. Схема моделируемого канала передачи данных

Введем оператор передатчика

$$snd: F_2^n \to C_2^n, \tag{1}$$

где $snd(\bar{a}) = (\phi(a_1), \ldots, \phi(a_n))$, $\bar{a} = (a_1, \ldots, a_n) \in F_2^n$, и рассмотрим отображение

$$\mu_n = (snd)^{-1}. \tag{2}$$

Полученные векторы $\bar{z} = (z_1, z_2, \ldots, z_n)$ направляются передатчиком в линию связи. Заметим, что физический аналог сигнала z_j можно получить, например, применяя двоичную фазовую манипуляцию, в которой фаза несущего колебания смещается на одно из двух значений: ноль или π [11].

На вход линии связи подаются дискретные сигналы из C_2, интерпретируемые как элементы R. В результате присутствующих в линии связи помех, на выходе вместо сигналов ±1 могут появиться произвольные вещественные числа из R. Отметим, что рассматриваемая линия связи является дискретной по входу и непрерывной по выходу. Таким образом, под воздействием шума координаты вектора $\bar{z}$ искажаются и формируется вектор $\bar{z}' = (z_1', z_2', \ldots, z_n') \in R^n$, и в результате получаем оператор линии связи $line: C_2^n \to R^n$.

Из линии связи вектор $\bar{z}' \in R^n$ поступает на вход приемника, который преобразует сигнал z'_s в y_s из допустимой области $\Xi_\varepsilon = \{\xi \in R | \varepsilon \le |\xi| \le 1/\varepsilon\}$, где $\varepsilon (\in (0;1])$ — параметр приемника. После такой фильтрации на выходе приемника появляется вектор $\bar{y} = (y_1, \ldots, y_n) \in \Xi_\varepsilon^n$. В результате, получаем оператор приемника с мягкими решениями:

$$rcv: R^n \to \Xi_\varepsilon^n. \tag{3}$$

Рассмотрим подробнее действие фильтра (см. рис. 2). Заштрихованные участки соответствуют допустимой области Ξ_ε, черные точки соответствуют корректным значениям сигнала из алфавита $C_2 = \{-1; 1\}$. Белые точки соответствуют искаженным, но оставшимся в допустимой области, сигналам. После фильтрации сигналы, принадлежащие области Ξ_ε, не изменяются. Если же искажение переводит сигнал за границы этой области (серые точки), то приемник смещает этот сигнал в область Ξ_ε по кратчайшему пути.

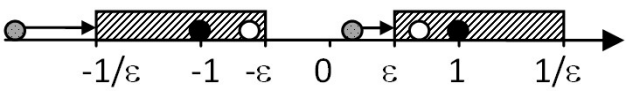

Рис. 2. Действие фильтра

Оператор непомехоустойчивого канала определим равенством

$$chn = rcv \cdot line \cdot snd\colon F_2^n \to \Xi_\varepsilon^n.$$

На вход декодера мягких решений поступает вектор $\bar{y}\,(\in \Xi_\varepsilon^n)$ с выхода приемника. Задача декодера состоит в восстановлении исходного информационного вектора $\bar{m} \in F_2^k$, через Dcd обозначим оператор декодера $Dcd\colon \Xi_\varepsilon^n \to F_2^n$. Результат декодирования $\bar{v}$ передается получателю сообщения, при этом, если помеховая обстановка такова, что декодирование прошло успешно, то $\bar{m} = \bar{v}$.

Оператор помехоустойчивого канала определим формулой:

$$Chn = Dcd \cdot rcv \cdot line \cdot snd \cdot Cod\colon F_2^k \to F_2^k.$$

Из определений описанных операторов вытекает, что $Chn = Dcd \cdot chn \cdot Cod$.

Если в модели приемник принимает жесткие решение, т.е. формирует на своем выходе элементы из C_2, то реализуется дискретный непомехоустойчивый канал. Оператор этого канала имеет вид

$$chn_d = rcv \cdot line \cdot snd\colon F_2^n \to C_2^n, \tag{4}$$

и поэтому $Chn = Dcd \cdot chn_d \cdot Cod$.

Отметим, что для троичного канала связи подобная модель была ранее рассмотрена в [3–5], однако двоичная модель имеет отличие в геометрическом представлении сигнала и фильтрации.

2. Основные определения и свойства двоичных кодов Рида—Маллера

Над полем F_2 рассмотрим кольцо $F_2[x_1,\dots,x_m]$ полиномов от m переменных. Далее будем полагать, что все мономы в полиномах этого кольца имеют вид $\phi = x_1^{\alpha_1}\dots x_m^{\alpha_m}$, $\alpha_i \in \{0;1\}$, $i = \overline{1,m}$.

Пусть $\rho(\bar{\alpha}) = \alpha_1 + \alpha_2 + \dots + \alpha_m$, где $\bar{\alpha} = (\alpha_1,\dots,\alpha_m) \in \{0;1\}^m$. Степень монома $\bar{x}^{\bar{\alpha}} = x_1^{\alpha_1}\dots x_m^{\alpha_m}$ определяется как $\deg(\bar{x}^{\bar{\alpha}}) = \rho(\bar{\alpha})$. Степень полинома $f \in F_2[x_1,x_2,\dots,x_m]$ определяется как максимум степеней его мономов с ненулевыми коэффициентами.

Зафиксируем упорядочение из $n = 2^m$ векторов

$$Order^m = \{\bar{\alpha}_1,\dots,\bar{\alpha}_n\}, \tag{5}$$

где $\bar{\alpha}_j = (\alpha_{j_1},\alpha_{j_2},\dots\alpha_{j_m})$, $\alpha_{j_i} \in \{0;1\}$, элементы которого расположены по возрастанию $\rho(\bar{\alpha})$, и упорядочены лексикографически при одинаковых $\rho(\bar{\alpha})$. Например,

$$Order^3 = \{(0,0,0);(0,0,1);(0,1,0);(1,0,0);(0,1,1);(1,0,1);(1,1,0);(1,1,1)\},$$

$$Order^2 = \{(0,0);\ (0,1);\ (1,0);\ (1,1)\ \}.$$

Далее полиномы $f \in F_2[x_1,\dots,x_m]$ будем записывать, располагая их слагаемые согласно упорядочению (5), и рассматривать их как сумму однородных полиномов со степенями от нуля до $\deg(f)$:

$$f(\bar{x}) = \sum\nolimits_{\rho(\bar{\alpha})\le\deg(f)} f_{\alpha_1\alpha_2\dots\alpha_m} x^{\alpha_1}x^{\alpha_2}\dots x^{\alpha_m} = f_{\bar{0}}x^{\bar{0}} + \sum\nolimits_{\rho(\bar{\alpha})=1} f_{\bar{\alpha}}\bar{x}^{\bar{\alpha}} + \sum\nolimits_{\rho(\bar{\alpha})=2} f_{\bar{\alpha}}\bar{x}^{\bar{\alpha}} + \dots + \sum\nolimits_{\rho(\bar{\alpha})=\deg(f)} f_{\bar{\alpha}}\bar{x}^{\bar{\alpha}} \tag{6}$$

Полиномы из $F_2[x_1,\dots,x_m]$, степени которых не превышают r, образуют линейное пространство $F_2^{(r)}[x_1,\dots,x_m]$:

$$F_2^{(r)}[x_1,\dots,x_m] = \{f(x_1,\dots,x_m) \in F_2\ [x_1,\dots,x_m] \mid \deg(f(x_1,\dots,x_m)) \le r\}.$$

Используя (5) для нумерации элементов кодового слова, определим двоичный код Рида—Маллера [2, 7] с параметрами r, m, где $m \ge r \ge 1$, $m \ge 2$, следующим образом:

$$RM(r,m) = \{(f(\bar{\alpha}_1),\dots,f(\bar{\alpha}_n)) | f \in F_2^{(r)}[x_1,\dots,x_m]\} \subset F_2^n. \tag{7}$$

Параметр r называют порядком кода. Информационными полиномами кода $RM(r,m)$ являются полиномы из кольца $F_2^{(r)}[x_1,\ldots,x_m]$. Вектор $\bar{v} \in F_2^k$ коэффициентов информационного полинома $v(x_1,\ldots,x_m)$, называется информационным вектором. Основные параметры кода $RM(r,m)$ длина n, размерность k, минимальное кодовое расстояние d и число исправляемых ошибок t вычисляются по формулам:

$$n = 2^m,\ k = \sum_{i=0}^{r} C_m^i,\ d = 2^{m-r},\ t = 2^{m-r-1} - 1.$$

В случае кода $RM(1,m)$

$$n = 2^m,\ k = 1 + m,\ d = 2^{m-1},\ t = 2^{m-2} - 1, \tag{8}$$

а информационные полиномы записываются в виде

$$f(\bar{x}) = \sum\nolimits_{\rho(\bar{\alpha})\le 1} f_{\bar{\alpha}}\bar{x}^{\bar{\alpha}} = a_{\bar{0}}\bar{x}^{\bar{\alpha}} + \sum\nolimits_{\rho(\bar{\alpha})=1} a_{\bar{\alpha}}\bar{x}^{\bar{\alpha}}.$$

В случае кода $RM(2,m)$

$$n = 2^m,\ k = 1 + m + C_m^2\ ,\ d = 2^{m-2},\ t = 2^{m-3} - 1, \tag{9}$$

а информационные полиномы записываются в виде

$$f(\bar{x}) = \sum\nolimits_{\rho(\bar{\alpha})\le 2} f_{\bar{\alpha}}\bar{x}^{\bar{\alpha}} = a_{\bar{0}}\bar{x}^{\bar{\alpha}} + \sum\nolimits_{\rho(\bar{\alpha})=1} a_{\bar{\alpha}}\bar{x}^{\bar{\alpha}} + \sum\nolimits_{\rho(\bar{\alpha})=2} a_{\bar{\alpha}}\bar{x}^{\bar{\alpha}}.$$

Определим оператор кодирования

$$C: F_2^{(r)}[x_1,\ldots,x_m] \to F_2^n,$$

формулой

$$C(f) = (f(\bar{\alpha}_1),\ldots,f(\bar{\alpha}_n)), \tag{10}$$

где $f \in F_2^{(r)}[x_1,\ldots,x_m]$ информационный полином, $\alpha_i \in Order^m$.

Согласно [3, 6, 7] оператор дифференцирования полинома $f \in F_2^{(r)}[x_1,\ldots,x_m]$ по направлению $\bar{b} \in F_2^m$:

$$D_{\bar{b}}: F_2[x_1,\ldots,x_m] \to F_2[x_1,\ldots,x_m]$$

определяется правилом:

$$(D_{\bar{b}}f)(\bar{x}) = f_{\bar{b}}(\bar{x}) - f(\bar{x}),\quad \bar{x} \in F_2^m, \tag{11}$$

где

$$f_{\bar{b}}(\bar{x}) = f(\bar{x} + \bar{b}).$$

Полином $D_{\bar{b}}f$ называется производным полиномом для полинома $f \in F_2^{(r)}[x_1,\ldots,x_m]$ по направлению $\bar{b} \in F_q^m$.

Рассмотрим $f \in F_2^{(r)}[x_1,\ldots,x_m]$, $\bar{b} \in F_2^m$, тогда $D_{\bar{b}}f \in F_2^{(r-1)}[x_1,\ldots,x_m]$, а ограничение $D_{\bar{b}}$ на $F_2^{(r)}[x_1,\ldots,x_m]$ задает линейный оператор:

$$D_{\bar{b}}:\ F_q^{(r)}[x_1,\ldots,x_m] \to F_q^{(r-1)}[x_1,\ldots,x_m].$$

В работе [6] показано, что, если $f(\bar{x}) \in F_2^{(2)}[x_1,\ldots,x_m]$ — полином в виде (6), $\bar{b} = (b_1,\ldots,b_m) \in F_2^m$, тогда полиномы $f(\bar{x})$ и $(D_{\bar{b}}f)(\bar{x})$ могут быть записаны с использованием квадратичных форм:

$$f(\bar{x}) = f_{\bar{0}} + \bar{x}\hat{\bar{f}}^T + \bar{x}A\bar{x}^T, \tag{12}$$

где

$$\hat{\bar{f}} = (f_{10\ldots00}, f_{01\ldots00}, \ldots, f_{00\ldots01}),$$

$$A = \begin{pmatrix} 0 & f_{110..00} & f_{101..00} & \cdots & f_{100..10} & f_{100..01} \\ 0 & 0 & f_{011..00} & \cdots & f_{010..10} & f_{010..01} \\ 0 & 0 & 0 & \cdots & f_{001..10} & f_{001..01} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & f_{000..11} \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix},$$

$$(D_{\bar{b}}f)(\bar{x}) = \bar{x}(A + A^T)\bar{b}^T + f(\bar{b}) - f_{\bar{0}}. \tag{13}$$

Построим аналог оператора $D_{\bar{b}}$ (см. (11)) для линейного пространства F_2^{n}, $n = 2^m$. Для нумерации элементов векторов из F_2^{n} будем использовать векторы из $\mathrm{F}_2^{\mathrm{m}}$, порядок которых определен в (5). Определим оператор сдвига $\tau_{\bar{b}}$, как перемешивающее биективное отображение

$$\tau_{\bar{b}}\colon F_2^{\mathrm{n}} \to F_2^{\mathrm{n}}, \quad \tau_{\bar{b}}(\bar{a}) = (a_{\bar{\alpha}_1+\bar{b}}, \ldots, a_{\bar{\alpha}_n+\bar{b}}),$$

где $\bar{a} = (a_{\bar{\alpha}_1}, \ldots, a_{\bar{\alpha}_n}) \in F_2^{\mathrm{n}}$, $\bar{b} = (b_1, \ldots, b_m) \in F_2^{\mathrm{m}}$. Разностный оператор $\Delta_{\bar{b}}$, являющийся аналогом $D_{\bar{b}}$, определим формулой:

$$\Delta_{\bar{b}}\colon F_2^{\mathrm{n}} \to F_2^{\mathrm{n}}, \Delta_{\bar{b}}(\bar{a}) = \tau_{\bar{b}}(\bar{a}) - \bar{a}, \bar{a} = (a_{\bar{\alpha}_1}, \ldots, a_{\bar{\alpha}_n}) \in F_2^{\mathrm{n}}. \tag{14}$$

Далее будем называть $\Delta_{\bar{b}}(\bar{a})$ производным вектором вектора $\bar{a}$ по направлению $\bar{b}$.

В работе [6] показано, что, если $f \in F_q^{(2)}[x_1, x_2, \ldots, x_m]$, $\bar{b} = (b_1, \ldots, \mathrm{b_m}) \in F_q^m$, то

$$\tau_{\bar{b}}(C(f)) = C(f_{\bar{b}}),\ C(D_{\bar{b}}f) = \Delta_{\bar{b}}(C(f)), \tag{15}$$

где $\Delta_{\bar{b}}$, $D_{\bar{b}}$ и C определены (10), (11) и (14) соответственно. Следовательно, если $C(w) \in RM(2,m)$, то и $\Delta_{\bar{b}}(C(w))) \in RM(1,m)$.

3. Декодер мягких решений для кодов $RM(2,m)$

Построим алгоритм декодирования мягких решений для кодов $RM(2,m)$, основываясь на модификации СПМ-декодера [1]. Напомним, что декодеры СП и СПМ построены для линии связи, на выходе которой появляются вещественные числа. При этом в обоих декодерах фактически вычисляется производный вектор в мультипликативном виде, хотя в соответствующих формулах вместо операции деления используется операция умножения. Ниже разработана модификация алгоритма СПМ, в которой производный вектор в мультипликативном виде вычисляется правильно с применением операции деления.

В пространстве Ξ_ε^n подобно (14) введем операторы

$$\xi_{\bar{b}}\colon \Xi_\varepsilon^n \to \Xi_\varepsilon^n,\ \nabla_{\bar{b}}\colon \Xi_\varepsilon^n \to \Xi_\varepsilon^n,$$

формулами

$$\xi_{\bar{b}}(\bar{Y}) = (y_{\bar{\beta}+\bar{\alpha}_1}, \ldots, y_{\bar{\beta}+\bar{\alpha}_n}), \nabla_{\bar{b}}(\bar{Y}) = (\zeta(y_{\bar{\beta}+\bar{\alpha}_1} y_{\bar{\alpha}_1}^{-1}), \ldots, \zeta(y_{\bar{\beta}+\bar{\alpha}n} y_{\bar{\alpha}n}^{-1})),$$

где $\bar{Y} = (y_{\bar{\alpha}_1}, \ldots, y_{\bar{\alpha}_n}) \in \Xi_\varepsilon^n$, ζ — фильтрующая функция, выполняемая оператором приемника rcv (см. (3)).

Вход: кодовый вектор $\bar{Y} = chn_d(C(f)) = (Y_{\bar{\alpha}_1}, \ldots, Y_{\bar{\alpha}_n}) \in C_2^n (\subset \Xi_\varepsilon^n)$, зашумленный в канале, а также параметры m, n, k кода $RM(2,m)$.

Выход: информационный вектор $\bar{f}$.

Шаг 1. По вектору $\bar{Y} \in \Xi_\varepsilon^n$ построим упорядоченый согласно с (5) набор векторов из R^n:

$$\left\{\nabla_{\bar{\gamma}}(\bar{Y}) = \zeta\left(Y_{\bar{\gamma}+\bar{\alpha}_1} Y_{\bar{\alpha}_1}^{-1}, \ldots, Y_{\bar{\gamma}+\bar{\alpha}_n} Y_{\bar{\alpha}_n}^{-1}\right)\right\}_{\bar{\gamma} \in F_2^m, \bar{\gamma} \neq 0},$$

где $\zeta = rcv$ (см. (3)).

Шаг 2. Рассмотрим все векторы $\bar{\gamma} \in F_2^m$, $\bar{\gamma} \neq \bar{0}$, и $\bar{P}^{\bar{\gamma}} = (P_{\bar{\alpha}_1}, \ldots, P_{\bar{\alpha}_n}) = \nabla_{\bar{\gamma}}(\bar{Y})$. Для каждого $\bar{\beta} = (\beta_1, \ldots, \beta_m) \in F_2^m$ вычислим функционал

$$\Psi(\bar{P}^{\bar{\gamma}}, \bar{\beta}) = \sum\nolimits_{s=1}^{n} \left| P_{\bar{\alpha}_s} (-1)^{\langle \bar{\beta}, \bar{\alpha}_s \rangle} \right| (\in R),$$

где $\langle \bar{\beta}, \bar{\alpha}_s \rangle (\in F_2)$ — скалярное произведение. Обозначим через $\Psi_{\bar{\gamma}}$ максимальное значение функционала для фиксированного $\bar{\gamma}$, а через $B_{\bar{\gamma}} = (\beta_1^{\bar{\gamma}}, \beta_2^{\bar{\gamma}}, \ldots, \beta_m^{\bar{\gamma}})$ вектор $\bar{\beta}$, на котором достигается $\Psi_{\bar{\gamma}}$. Если функционал $\Psi(\bar{P}^{\bar{\gamma}}, \bar{\beta})$ принимает одинаковое наибольшее значение на нескольких векторах $\bar{\beta}$, то $B_{\bar{\gamma}}$ выбирается случайно из таких векторов.

Сгенерируем, используя (5), набор $\Psi = (\Psi_{\bar{\alpha}_1 = 0}, \Psi_{\bar{\alpha}_2}, \ldots, \Psi_{\bar{\alpha}_n})$ и массив

$$B = \begin{pmatrix} B_{\bar{\alpha}_1} \\ B_{\bar{\alpha}_2} \\ \dots \\ B_{\bar{\alpha}_n} \end{pmatrix} = \begin{pmatrix} 0 & \dots & 0 \\ \beta_1^{\bar{\alpha}_2} & \dots & \beta_m^{\bar{\alpha}_2} \\ \dots & \dots & \dots \\ \beta_1^{\bar{\alpha}_n} & \dots & \beta_m^{\bar{\alpha}_n} \end{pmatrix}.$$

Шаг 3. Положим

$$\Theta = \begin{pmatrix} \Theta_{\bar{\alpha}_1} \\ \dots \\ \Theta_{\bar{\alpha}_n} \end{pmatrix} = \begin{pmatrix} \theta_{1,1} & \dots & \theta_{m,1} \\ \dots & \dots & \dots \\ \theta_{1,n} & \dots & \theta_{m,n} \end{pmatrix} := B$$

и пересчитаем строки Θ:

$$\Theta_{\bar{\alpha}_s} = Maj\left\{\Theta_{\bar{\alpha}_s+\bar{\beta}_j} - \Theta_{\bar{\beta}_j}\right\}_{\bar{\beta}_j \in F_2^m, \bar{\beta}_j \neq \bar{\alpha}_s, \bar{\beta}_j \neq \bar{0}},$$

где функция $Maj\{X\}$ находит элемент из конечного набора X, который встречается чаще остальных векторов в этом наборе.

Шаг 4. Для каждого $j = 1, \dots, m$ на множестве всех полиномов вида

$$\delta(\bar{x}) = \sum_{q=1}^{m} \delta_q x_q,\ \delta_q \in F_2,$$

находим максимум d_j функционала

$$T_j(\delta) = \sum_{s=1}^{n} \Psi_{\bar{\alpha}_s} (-1)^{\delta(\bar{\alpha}_s) - \theta_{j,s}} (\in R),$$

и полином $\omega^{(j)}(\bar{x}) = \sum_{q=1}^{m} \omega_q^{(j)} x_q$, на котором он достигается.

Шаг 5. Сформируем матрицу $(A + A^T) = (a_{qi})_{q,j \in [1,..,m]}$, где

$$a_{qj} = \begin{cases} \omega_j^{(q)}, \text{ если } d_q < d_j, \\ \omega_q^{(j)}, \text{ если } d_q \geq d_j. \end{cases}$$

Шаг 6. Вычислим однородный полином второго порядка $\psi(\bar{x})$

$$\psi(\bar{x}) = \sum_{q<j} a_{qj} x_q x_j, \text{ где } \bar{x} = (x_1, x_2, \dots, x_m) \in F_2^m,\ a_{qj} \in F_2,\ q, j \in [1, \dots, m].$$

Шаг 7. (5.) На множестве всех полиномов $\bar{\zeta}(\bar{x})(\in F_2^{(1)}[x_1, x_2, \dots, x_m])$ найдем полином $\varphi(\bar{x})$, минимизирующий функционал

$$\Phi(Y, \zeta) = \sum_{s=1}^{n} \left| Y_{\bar{\alpha}_s} (-1)^{\bar{\zeta}(\bar{\alpha}_s) + \psi(\bar{\alpha}_s)} \right| (\in R).$$

Искомый информационный вектор $\bar{f}$ определяется полиномом $f(\bar{x}) = \psi(\bar{x}) + \varphi(\bar{x})$.

4. Обоснование корректности декодера мягких решений для гладкого канала

Рассмотрим построенную выше модель помехоустойчивого канала связи, использующую коды $RM(2, m)$. По аналогии с [5] дискретный помехоустойчивый канал, действие которого описывается оператором chn_d (4), назовем *гладким*, если зашумленные в канале векторы $C(f)(\in RM(2, m))$ и $C(D_{\bar{b}}(w))(\in RM(1, m))$, где $\bar{b} \in F_2^m$, связаны равенством

$$\Delta_{\bar{b}}(\mu_n(chn_d(C(f)))) = \mu_n(chn_d(C(D_{\bar{b}} f))). \quad (16)$$

В [5] отмечено, что понятие гладкости канала возникло в связи с некоторой аналогией из теории гладких преобразований дифференцируемых многообразий [12].

Лемма. Рассмотрим дискретный помехоустойчивый канал, дейсвие которого описывается оператором chn_d (4) и для которого выполняется условие гладкости (16). Пусть $f \in F_2^{(2)}[x_1, \dots, x_m]$, $\bar{b} \in F_2^m$,

$$\bar{e} = \mu_n(chn_d(C(f))) - C(f),\ \bar{\varepsilon} = \mu_n(chn_d(C(D_{\bar{b}} f))) - C(D_{\bar{b}} f).$$

Тогда, если вес Хемминга вектора ошибки $\bar{e}$ ограничен сверху числом ошибок, исправление которых гарантируется параметрами кода $RM_2(2,m)$, т.е.

$$wt_h(\bar{e}) \le t_{RM_2(2,m)} = 2^{m-3} - 1,$$

то вес Хемминга вектора ошибки $\bar{\varepsilon}$ ограничен сверху числом ошибок, исправление которых гарантируется параметрами кода $RM_2(1,m)$, т.е.

$$wt_h(\bar{\varepsilon}) \le t_{RM_2(1,m)} = 2^{m-2} - 1.$$

Доказательство. Количество ошибок $t_{RM_2(2,m)}$ и $t_{RM_2(1,m)}$, которые исправляются кодами $RM_2(2,m)$, $RM_2(1,m)$ подсчитаны в (8–9). Из (16), свойства линейности оператора $\Delta_{\bar{b}}$ и (15) получаем

$$\mu_n\left(chn_d\left(C(D_{\bar{b}}f)\right)\right) = \Delta_{\bar{b}}\left(\mu_n\left(chn_d\left(C(f)\right)\right)\right) = \Delta_{\bar{b}}(C(f) + \bar{e}) =$$
$$= \Delta_{\bar{b}}\left(C(f)\right) + \Delta_{\bar{b}}(\bar{e}) = C(D_{\bar{b}}f) + \tau_{\bar{b}}(\bar{e}) - \bar{e}.$$

Следовательно,

$$\bar{\varepsilon} = chn_d\left(C(D_{\bar{b}}f)\right) - C(D_{\bar{b}}f) = \tau_{\bar{b}}(\bar{e}) - \bar{e}.$$

Используя неравенство $wt_h(\bar{e}) \le 2^{m-3} - 1$, оценим сверху вес вектора $\bar{\varepsilon}$:

$$wt_h(\bar{\varepsilon}) = wt_h(\tau_{\bar{b}}(\bar{e}) - \bar{e}) \le wt_h(\tau_{\bar{b}}(\bar{e})) + wt_h(\bar{e}) \le 2(2^{m-3} - 1) = 2^{m-2} - 2.$$

Легко видеть, что выполняется условие леммы

$$wt_h(\bar{\varepsilon}) \le 2^{m-2} - 1.$$

В доказательстве леммы фактически показано, что если в передаваемом по каналу кодовому слову кода Рида—Маллера второго порядка произойдет ошибок не более чем $t_{RM_2(2,m)} = 2^{m-3} - 1$, то в производном векторе, построенном на основе зашумленного кодового слова, ошибок будет не более $2^{m-2} - 2$. Но производный вектор является кодовым словом кода Рида—Маллера первого порядка, следовательно, в нем может быть гарантировано исправлено не более чем $2^{m-2} - 1$ ошибок, т.е. есть некоторый «запас» по исправлению ошибок. Однако если увеличить на единицу число «разрешенных» ошибок в кодовом слове кода второго порядка, то исправление всех ошибок в производном векторе не гарантируется.

Теорема. Рассмотрим двоичный канал, помехоустойчивость которого обеспечивается применением кодов Рида—Маллера $RM(2,m)$, для которого выполняется условие гладкости (15). Пусть $\bar{f} \in F_2^k$ и $f \in F_2^{(2)}[x_1,\dots,x_m]$ — соответствующие друг другу информационные вектор и полином. Предположим, что кодовое слово $C(f) \in RM_2(2,m)$ было отправлено в канал, а из канала принят вектор $\bar{Y} = chn_d(C(f)) \in C_2(\subset \Xi_\varepsilon^n)$, такой, что

$$wt_h(C(f) - \mu_n(\bar{Y})) \le 2^{m-3} - 1.$$

Тогда алгоритм декодирования на выходе строит полином f.

Доказательство. На первом шаге по $\bar{Y} = chn_d(C(f)) \in C_2^n(\subset \Xi_\varepsilon^n)$ строятся векторы

$$\nabla_{\bar{\gamma}}(\bar{Y}) = \nabla_{\bar{\gamma}}(chn_d(C(f))),$$

где $\bar{\gamma} \in F_2^m$, $\bar{\gamma} \ne \bar{0}$.

Покажем, что на этом шаге фактически вычисляются векторы, которые могли быть получены, если бы по каналу передавалось бы не кодовое слово $C(f)$, а производные от этого слова во всех направлениях, т.е. $\nabla_{\bar{\gamma}}(\bar{Y}) = chn_d(C(D_{\bar{\gamma}}f))$, $\bar{\gamma} \in F_2^m, \bar{\gamma} \ne \bar{0}$.

Введем ограничения на C_2^n отображений $\xi_{\bar{b}}$, $\Delta_{\bar{b}}$:

$$\tilde{\xi}_{\bar{b}}: C_2^n \to C_2^n,\ \tilde{\nabla}_{\bar{b}}: C_2^n \to C_2^n,$$

где $\bar{b} \in F_2^m$. Прямыми выкладками (см. (2), (14)) проверяется, что

$$\tau_{\bar{b}} \cdot \mu_n = \mu_n \cdot \tilde{\xi}_{\bar{b}}, \Delta_{\bar{b}} \cdot \mu_n = \mu_n \cdot \tilde{\nabla}_{\bar{b}}. \qquad (17)$$

Выше с использованием оператора μ_n пространство C_2^n отождествлялось с F_2^n (см. (1–2)), следовательно,

$$\mu_n(\nabla_{\bar{\gamma}}(\bar{Y})) = \mu_n(\tilde{\nabla}_{\bar{\gamma}}(\bar{Y})) = \mu_n(\tilde{\nabla}_{\bar{\gamma}}(chn_d(C(f)))),$$

используя (17), получаем

$$\mu_n(\tilde{\nabla}_{\bar{\gamma}}(chn_d(C(f)))) = \Delta_{\bar{\gamma}}(\mu_n(chn_d(C(f)))).$$

Из полученных равенств и условия гладкости (16) вытекает, что

$$\mu_n(\nabla_{\bar{\gamma}}(\bar{Y})) = \Delta_{\bar{\gamma}}(\mu_n(chn_d(C(f)))) = \mu_n(chn_d(C(D_{\bar{\gamma}}f))).$$

Вектор $C(D_{\bar{\gamma}}f)$ является кодовым словом $RM(1,m)$ и в силу леммы

$$wt_h(C(D_{\bar{\gamma}}f) - chn_d(C(D_{\bar{\gamma}}f))) < 2^{m-2} - 1.$$

На втором шаге из векторов $\nabla_{\bar{\gamma}}(\bar{Y}) = chn_d(C(D_{\bar{\gamma}}f)) = \tilde{\nabla}_{\bar{\gamma}}(\bar{Y})$, $\bar{\gamma} \in F_2^m$, $\bar{\gamma} \neq \bar{0}$, с помощью $\Psi(\bar{P},\bar{\beta})$, находится линейный однородный полином $\beta^{\bar{\gamma}}(\bar{x}) = \beta_1 x_1 + \ldots + \beta_m x_m$, который в закодированном виде $C(\beta^{\bar{\gamma}}(\bar{x})) = ((-1)^{\beta^{\bar{\gamma}}(\alpha_1)}, \ldots, (-1)^{\beta^{\bar{\gamma}}(\alpha_n)})(\in C_2^n)$ из всех подобных полиномов наиболее близок к $\nabla_{\bar{\gamma}}(\bar{Y})$. Как было показано в лемме, все ошибки в $\nabla_{\bar{\gamma}}(\bar{Y}) = chn_d(C(D_{\bar{\gamma}}f))$ исправляются кодом $RM(1,m)$. Фактически на шаге 2 производные полиномы декодируются по минимуму расстояния Хемминга, таким образом

$$\beta^{\bar{\gamma}}(\bar{x}) = \beta_1 x_1 + \ldots + \beta_m x_m = D_{\bar{\gamma}}f(\bar{x}).$$

Из (13)

$$\beta^{\bar{\gamma}}(\bar{x}) = \bar{x}(A + A^T)\bar{\gamma}^T + f(\bar{\gamma}) - f_{00\ldots00},$$

следовательно,

$$B_{\bar{\gamma}} = (\beta_1^{\bar{\gamma}}, \ldots, \beta_m^{\bar{\gamma}}) = ((A + A^T)\bar{\gamma}^T)^T = \bar{\gamma}(A + A^T). \quad (18)$$

Получаем, что строки матрицы B содержат верные значения коэффициентов однородной части $L_{\bar{\gamma}}f$ производной $D_{\bar{\gamma}}f$, $\bar{\gamma} \in F_2^m$. Тогда

$$B_{\bar{\gamma}} = \overline{L_{\bar{\gamma}}f}. \quad (19)$$

Обозначим s-тый столбец матрицы $(A + A^T)$ через $(A + A^T)^{(s)}$, тогда в силу (18)

$$B = \begin{pmatrix} B_{\bar{\alpha}_1} \\ B_{\bar{\alpha}_2} \\ \ldots \\ B_{\bar{\alpha}_n} \end{pmatrix} = \begin{pmatrix} \overline{L_{\bar{\alpha}_1}f} \\ \overline{L_{\bar{\alpha}_2}f} \\ \ldots \\ \overline{L_{\bar{\alpha}_n}f} \end{pmatrix} = \begin{pmatrix} \bar{\alpha}_1(A + A^T)^{(1)} & \ldots & \bar{\alpha}_1(A + A^T)^{(m)} \\ \bar{\alpha}_2(A + A^T)^{(1)} & \ldots & \bar{\alpha}_2(A + A^T)^{(m)} \\ \ldots & \ldots & \ldots \\ \bar{\alpha}_n(A + A^T)^{(1)} & \ldots & \bar{\alpha}_n(A + A^T)^{(m)} \end{pmatrix}.$$

В строках матрицы B, которые соответствуют векторам $\bar{\alpha}_i$ веса 1, т.е. $\bar{e}_1$=(1,0,...,0), $\bar{e}_2$=(0,1,...,0), ..., $\bar{e}_m$=(0,0,...,1), расположены неизменённые строки матрицы $(A + A^T)$. Согласно (19) в этих же строках расположены векторы коэффициентов однородной части $\overline{L_{\bar{e}_j}f}$ производной $D_{\bar{e}_j}f$. Тогда из симметричности матрицы $(A + A^T)$ вытекает, что ее столбцы так же содержат значения $\overline{L_{\bar{e}_j}f}$, $j = \overline{1,m}$.

В столбце s матрицы B расположены скалярные произведения вектора $(A + A^T)^s = \overline{L_{\bar{e}_s}f}$ на векторы $\bar{\alpha}_1, \ldots, \bar{\alpha}_n$ из (5). Другими словами, в столбце s матрицы B расположены закодированные значения однородной части производной $D_{\bar{e}_s}f$ (см. (19)).

Точность нахождения $B_{\bar{\gamma}}$ можно оценить по элементы $\Psi_{\bar{\gamma}}$ набора Ψ, а именно: чем больше параметр $\Psi_{\bar{\gamma}}$, тем точнее найдено $B_{\bar{\gamma}}$. Если декодируемое слов не содержит ошибок, то все элементы набора Ψ равны n.

На третьем шаге уточняются значения элементов B. При описанных в формулировке теоремы условиях на канал связи значения элементов B на этом шаге не изменяются. Покажем это. Согласно (17) для произвольных $\bar{\alpha}_s$, $\bar{\beta}_j \in F_2^m$ справедливо

$$\bar{\alpha}_s(A + A^T) + \bar{\beta}_j(A + A^T) = (\bar{\alpha}_s + \bar{\beta}_j)(A + A^T),$$

поэтому, должно выполняться

$$B_{\bar{\alpha}_s} + B_{\bar{\beta}_j} = B_{\bar{\alpha}_s + \bar{\beta}_j}.$$

Как уже было отмечено, векторы $B_{\bar{\gamma}}$ построены верно, следовательно, процедура пересчета строк не изменяет матрицу $\Theta = B$.

На вход четвертого шага поступает матрица $\Theta = B$. Выше было показано, что ее строки, соответствующие производным по базисным направлениям $\bar{e}_1$=(1,0,...,0), $\bar{e}_2$=(0,1,...,0), ..., $\bar{e}_m$=(0,0,...,1), формируют матрицу $(A + A^T)$ (см. (12)). Следовательно, в случае гладкого дискретного помехоустойчивого канала передачи данных, в котором количество ошибок не превосходит половины кодового расстояния, матрица $(A + A^T)$ уже построена. Но так как декодер разработан работы в ситуации с большим числом ошибок, то он продолжает работать и на 4, 5 и 6 шагах строит матрицу $(A + A^T)$. А именно, на четвертом шаге он строит вспомогательные полиномы $\delta(\bar{x}) = \sum_{q=1}^{m} \delta_q x_q$, $\delta(\bar{x}) \in F_2^m$, максимизирующие функционала $T_j(\delta)$.

На пятом шаге используя коэффициенты полиномов $\delta(\bar{x})$, декодер формирует матрицу $(A + A^T)$. При этом он учитывает ее симметричную структуру.

На шестом шаге из коэффициентов матрицы $(A + A^T)$ формируется полином $\pi(\bar{x})$, представляющий собой часть искомого полинома $f(\bar{x})$, содержащая квадратичные слагаемые. Обозначим эту часть полинома $\pi(\bar{x})$.

Покажем, что, с учетом условия теоремы, шаги 4, 5 и 6 не изменяют матрицу $(A + A^T)$. Функционал $T_j(\delta)$ для каждого $j \in \{1, \ldots, m\}$достигает максимального значения при $\delta(\bar{x}) = (L_{\bar{e}_j} f)(\bar{x})$:

$$\delta(\bar{\alpha}_s) = \left(L_{\bar{e}_j} f\right)(\bar{\alpha}_s) = \bar{\alpha}_s \left(\overline{L_{\bar{e}_j} f}\right)^T = \bar{\alpha}_s \left(B_{\bar{e}_j}^T\right) = \bar{\alpha}_s A \bar{e}_j^T = B_{\bar{\alpha}_s}^T \bar{e}_j^T = \overline{L_{\bar{\alpha}_s} f}\, \bar{e}_j^T = \theta_{js},$$

следовательно, найденные значения $\delta(\bar{x})$ совпадают со строками/столбцами матрицы $(A + A^T)$. Действия шага 5 направлены на симметризацию матрицы $(A + A^T)$, однако, при соблюдении условия теоремы построенная матрица $(A + A^T) = (a_{qi})_{q,j \in [1,..,m]}$ уже является симметричной. Следовательно, квадратичная часть ψ искомого информационного полинома f кода $RM(2, m)$ восстанавливается верно.

На вход седьмого шага алгоритма поступает полином ψ, который является квадратичной частью искомого информационного полинома f кода $RM(2, m)$.

Затем перебираются все возможные значения линейной части ϕ полинома $f = \phi + \psi$. Каждый полином f кодируется, и среди них всех полученных кодовых векторов $C(f)$ находится ближайший по L_1-метрике к вектору $\bar{Y}$.

Учитывая введенное в теореме ограничение на число ошибок в зашумленном кодовом векторе $\bar{Y}$, линейная часть ϕ и сам полином f находятся алгоритмом верно.

Заключение

В работе рассмотрен мягкий вероятностный декодер кодов Рида—Маллера, разработанный В.М. Сидельниковым и А.С. Першаковым, с изменениями, внесенными П. Лоидрю и Б. Саккуром, позволившими уменьшить вычислительную сложность декодирования. Для этого декодера известны результаты экспериментов, подтверждающие его высокую корректирующую способность [1, 8].

В разработанной общей модели канала предусматривается специальный фильтр, дающий возможность применять в декодере мультипликативный бинарный алфавит C_2 , что позволило построить такую модификацию алгоритма СПМ, в которой производный

вектор в мультипликативном виде вычисляется правильно. Получено теоретическое обоснование корректности этого декодера при выполнении условия гладкости канала.

Дальнейшие исследования могут быть связаны с использованием построенного декодера в кодовых криптосистемах и для распределенной передачи данных (см., например, [6]).

Литература

1. Loidreau P., Sakkour B. Modified version of Sidel'nikov-Pershakov decoding algorithm for binary second order Reed—Muller codes // Ninth International Workshop on Algebraic and Combinatorial Coding theory (ACCT'2004) (Kranevo, Bulgaria, 2004). 2004. P. 266–271.
2. Pellikaan R., Wu X.-W. List decoding of q-ary Reed—Muller Codes // IEEE Trans. On Information Theory. 2004. Vol. 50, no. 3. P. 679–682. DOI: 10.1109/tit.2004.825043.
3. Деундяк В.М., Могилевская Н.С. Дифференцирование полиномов нескольких переменных над полями Галуа нечетной мощности и приложения к кодам Рида-Маллера // Вестник Донского государственного технического университета. 2018. Т. 18, № 3. С. 339–348. DOI: 10.23947/1992-5980-2018-18-3-339-348.
4. Деундяк В.М., Могилевская Н.С. Модель троичного канала передачи данных с использованием декодера мягких решений кодов Рида—Маллера второго порядка // Известия высших учебных заведений. Северо-Кавказский регион. Серия: Технические науки. 2015. № 1(182). С. 3–10. DOI: 10.17213/0321-2653-2015-1-3-10.
5. Деундяк В.М., Могилевская Н.С. Об условиях корректности декодера мягких решений троичных кодов Рида—Маллера второго порядка // Владикавказский математический журнал. 2016. Т. 18, № 4. С. 23–33.
6. Деундяк В.М., Могилевская Н.С. Схема разделенной передачи конфиденциальных данных на основе дифференцирования полиномов нескольких переменных над простыми полями Галуа // Вопросы кибербезопасности. 2017. Т. 5, № 24. С. 64–71. DOI: 10.21681/2311-3456-2017-5-64-71.
7. Логачев О.А. Булевы функции в теории кодирования и криптологии. Москва: МЦНМО, 2004. 470 с.
8. Могилевская Н.С., Скоробогат В.Р., Чудаков В.С. Экспериментальное исследование декодеров кодов Рида—Маллера второго порядка // Вестник ДГТУ. 2008. Т. 8, № 3. С. 231–237.
9. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. Москва: Техносфера, 2005. 320 с.
10. Сидельников В.М., Першаков А.С. Декодирование кодов Рида—Маллера при большом числе ошибок // Пробл. передачи информ. 1992. Т. 28, № 3. С. 80–94.
11. Скляр Б. Цифровая связь. Теоретические основы и практическое применение, 2-е издание. Москва: Издательский дом «Вильямс», 2016. 1104 с.
12. Хирш М. Дифференциальная топология. Москва: Мир, 1979. 280 с.

Деундяк Владимир Михайлович, к.ф.-м.н., доцент, ФГНУ НИИ «Спецвузавтоматика», кафедра алгебры и дискретной математики, Южный федеральный университет (Ростов-на-Дону, Российская Федерация)

Могилевская Надежда Сергеевна, к.т.н., доцент, кафедра алгебры и дискретной математики, Южный федеральный университет (Ростов-на-Дону, Российская Федерация)

DOI: 10.14529/cmse200204

ON SOFT SOLUTIONS DECODER FOR REED–MULLER BINARY CODES OF THE SECOND ORDER

© 2020 V.M. Deundyak[1,2], N.S. Mogilevskaya[2]
[1]*FGNU NII "Specvuzavtomatika"*
(Gazetniy 51, Rostov-on-Don, 344002 Russia),
[2]*Southern Federal University*
(Milchakova 8a, Rostov-on-Don, 344090 Russia)
E-mail: vl.deundyak@gmail.com, nadezhda.mogilevskaia@yandex.ru
Received: 22.11.2019

A general model of a noise-resistant binary data channel is constructed, intended for use with various soft decision decoders. The communication line considered in the model is discrete in input and continuous in output. Discrete signals from the multiplicative binary alphabet are received at its input, and due to distortions acting in the communication line, symbols from the multiplicative group of the field of real numbers are formed at the output after filtering, which are then fed to the input of the error-correcting code decoder. Soft and probabilistic decoders of error-correcting codes allow correcting more errors in code words than is guaranteed by the minimum distance of the code used. The paper considers a probabilistic Sidelnikov—Pershakov decoder of soft solutions for Reed—Muller codes of the second order in the modification proposed by P. Loidreau and B. Sakkour. Earlier, the effectiveness of these decoders was confirmed by simulation experiments, but there was no theoretical justification. In this paper, the requirement to the communication channel, called the smoothness of the channel, is formulated, in which the correctness of this decoder is theoretically proved in the case when the number of errors per code word does not exceed half the code distance. The proof is based on the use of the theory of quadratic forms and methods of differential calculus in the polynomial ring of several variables over Galois fields.

Keywords: Reed–Muller codes, decoder, model of channel, proof of decoder correctness.

FOR CITATION

Deundyak V.M., Mogilevskaya N.S. On Soft Solutions Decoder for Reed–Muller Binary Codes of the Second Order. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 2. P. 55–67. (in Russian) DOI: 10.14529/cmse200204.

References

1. Loidreau P., Sakkour B. Modified version of Sidel'nikov–Pershakov decoding algorithm for binary second order Reed–Muller codes. Ninth International Workshop on Algebraic and Combinatorial Coding theory (ACCT'2004) (Kranevo, Bulgaria, 2004). 2004. P. 266–271.
2. Pellikaan R., Wu X.-W. List decoding of q-ary Reed–Muller Codes. IEEE Trans. On Information Theory. 2004. Vol. 50, no. 3. P. 679–682. DOI: 10.1109/tit.2004.825043.
3. Deundyak V.M., Mogilevskaya N.S. Differentiation of polynomials in several variables over Galois fields of odd cardinality and applications to Reed–Muller codes. Vestnik of Don

State Technical University. 2018. Vol. 18, no. 3. P. 339–348. (in Russian) DOI: 10.23947/1992-5980-2018-18-3-339-348.

4. Deundyak V.M., Mogilevskaya N.S. The model of the ternary communication channel with using the decoder of soft decision for Reed–Muller codes of the second order. University news. North-Caucasian region. Technical sciences series. 2015. Vol. 1, no. 182. P. 3–10. (in Russian) DOI: 10.17213/0321-2653-2015-1-3-10.
5. Deundyak V.M., Mogilevskaya N.S. On Correctness Conditions of a Soft-Decisions Decoder for Ternary Reed–Muller Codes of Second Order. Vladikavkaz Mathematical Journal. 2016. Vol. 18, no. 4. P. 23–33. (in Russian)
6. Deundyak V.M., Mogilevskaya N.S. The confidential data divided transmission scheme based on differential calculus of polynomials in several variables over prime Galois fields Voprosy kiberbezopasnosti. 2017. Vol. 5, no. 24. P. 64–71. (in Russian) DOI: 10.21681/2311-3456-2017-5-64-71.
7. Logachev O.A., Salnikov A.A., Iashchenko V.V. Boolean functions in coding theory and cryptology. MCCME, 2004. 470 p. (in Russian)
8. Mogilevskaya N.S., Skorobogat V.R., Chudakov V.S. Experimental research of second order Reed–Muller codes. Vestnik of Don State Technical University. 2008. Vol. 8, no. 3. P. 231–237. (in Russian)
9. Morelos-Saragosa R. The art of noiseless coding. Methods, Algorithms, Application. Tekhnosfera, 2005. 320 p. (in Russian)
10. Sidelnikov V.M., Pershakov A.S. Decoding of Reed–Muller codes with a large number of errors. Problems of Information Transmission. 1992. Vol. 28, no. 3. P. 80–94. (in Russian)
11. Skliar B. Digital communication. Theoretical foundations and practical application. Viliams Press, 2016. 1104 p. (in Russian)
12. Hirsch M. Differential topology. Mir Press, 1979. 280 p. (in Russian)

УДК 517.977.56 **DOI: 10.14529/cmse200205**

КВАЗИОСОБЫЕ УПРАВЛЕНИЯ В ОДНОЙ СТУПЕНЧАТОЙ ЗАДАЧЕ УПРАВЛЕНИЯ ДИСКРЕТНЫМИ ДВУХПАРАМЕТРИЧЕСКИМИ СИСТЕМАМИ

© 2020 К.Б. Мансимов[1,2], Т.Ф. Мамедова[2]

[1]*Бакинский государственный университет*
(Az1148 Баку, ул. ак. З. Халилова, д. 23),
[2]*Институт систем управления НАН Азербайджана*
(Az1141 Баку, ул. Б. Вахабзаде, д. 9)
E-mail: kamilbmansimov@gmail.com, kmansimov@mail.ru

Поступила в редакцию: 11.07.2019

Изучается одна ступенчатая (т.е. многоэтапная) задача оптимального управления терминального типа функционалом качества, описываемая дискретными двухпараметрическими системами уравнений типа Форназини—Маркезини при предположении выпуклости областей управления. Дискретная двухпараметрическая система уравнений типа Форназини—Маркезини представляет собой разностный аналог системы гиперболических уравнений второго порядка (иногда такие системы уравнений в западной литературе называют также 2D системами). Применяя модифицированный аналог метода приращений, получено специальное разложение функционала качества второго порядка с помощью линеаризованных разностных систем уравнений.

Применяя один вариант метода приращений, установлено необходимое условие оптимальности первого порядка в форме линеаризованного (дифференциального) условия максимума. Отдельно изучен случай вырождения линеаризованного условия максимума (квазиособый случай). Используя представления решений линеаризованных разностных систем уравнений с помощью специальных формул приращения функционала качества, выведены конструктивно проверяемые квадратичные необходимые условия оптимальности квазиособых управлений.

Ключевые слова: дискретная двухпараметрическая система уравнений типа Форназини—Маркезини, линеаризованное необходимое условие оптимальности, ступенчатая задача, оптимальное управление, квазиособое управление, выпуклая область управления.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мансимов К.Б., Мамедова Т.Ф. Квазиособые управления в одной ступенчатой задаче управления дискретными двухпараметрическими системами // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 68–83. DOI: 10.14529/cmse200205.

Введение

Различные аспекты задач оптимального управления, описываемые дискретными двухпараметрическими системами типа Форназини—Маркезини, изучены в работах [1–12] и др. Подобными системами описываются многие реальные процессы [1–4]. Заметим, что модели Форназини—Маркезини представляют собой разностный аналог системы Гурса—Дарбу, т.е. разностный аналог системы гиперболических уравнений второго порядка с краевыми условиями Гурса. Для таких задач оптимального управления изучены вопросы, связанные управляемостью и наблюдаемостью, а также с выводом различных необходимых и достаточных условий оптимальности [3–12]. Многие управляемые процессы являются многоэтапными. Задачи оптимального управления такими процессами называют задачами оптимального управления с переменной структурой составными задачами оптимального управления, или же ступенчатыми задачами оптимального управления. (см., например, [13–18]). К настоящему времени изучены ряд ступенчатых (составных, многоэтапных) задач оптимального управления, а также задачи оптимального

управления с переменной структурой, описываемые обыкновенными дифференциальными, а также разностными уравнениями (см., например, [13–18]). В этих работах, в основном, установлены различные необходимые условия оптимальности первого порядка. Исследованию особых управлений в ступенчатых задачах оптимального управления, описываемых обыкновенными дифференциальными или же разностными уравнениями посвящены, например, работы [16, 18].

В предлагаемой работе исследуется одна ступенчатая задача оптимального управления, описываемая дискретной двухпараметрической системой при выпуклости областей управления. Подобная ступенчатая задача оптимального управления рассматривается впервые. Целью настоящей работы является при помощи аналога метода приращений вывод необходимого условия оптимальности в форме линеаризованного (дифференциального) принципа максимума и исследование случая вырождения линеаризованного условия максимума (квазиособый случай). Используя специальное разложение приращения функционала качества, удалось получить конструктивно проверяемые необходимые условия оптимальности. Установлен аналог линеаризованного условия максимума. Заметим, что в отличие от непрерывного случая в дискретных задачах оптимального управления, линеаризованный принцип максимума не является следствием дискретного принципа максимума. Далее изучен случай вырождения линеаризованного условия максимума (квазиособый случай [11, 18, 19]). Доказано необходимое условие оптимальности квазиособых управлений.

Структура работы следующая. В разделе 1 приводится постановка задачи. Далее в разделе 2 вычисляется специальное приращение критерия качества и устанавливается необходимые условия оптимальности. В заключении подводятся итоги проведенного исследования и обсуждаются возможные направления продолжения работы.

1. Постановка задачи

Предположим, что управляемый процесс описывается системой нелинейных разностных уравнений

$$z\left(t+1,x+1\right)=f\left(t,x,z\left(t,x\right),u\left(t,x\right)\right),$$
$$\left(t,x\right)\in D_1=\left\{\left(t,x\right):\, t=t_0,t_0+1,...,t_1-1;\; x=x_0,x_0+1,...,X-1\right\}, \tag{1}$$
$$y\left(t+1,x+1\right)=g\left(t,x,y\left(t,x\right),v\left(t,x\right)\right),$$
$$\left(t,x\right)\in D_2=\left\{\left(t,x\right):\, t=t_1,t_1+1,...,t_2-1;\; x=x_0,x_0+1,...,X-1\right\}, \tag{2}$$

с краевыми условиями

$$z\left(t_0,x\right)=a\left(x\right),\quad x=x_0,x_0+1,...,X,$$
$$z\left(t,x_0\right)=b_1\left(t\right),\; t=t_0,t_0+1,...,t_1, \tag{3}$$
$$a\left(x_0\right)=b_1\left(t_0\right),$$
$$y\left(t_1,x\right)=G\left(z\left(t_1,x\right)\right),\quad x=x_0,x_0+1,...,X,$$
$$y\left(t,x_0\right)=b_2\left(t\right),\; t=t_1,t_1+1,...,t_2, \tag{4}$$
$$G\left(z\left(t_1,x_0\right)\right)=b_2\left(t_1\right).$$

Здесь $f\left(t,x,z,u\right)$, $\left(g\left(t,x,y,v\right)\right)$ — заданная n $\left(m\right)$-мерная вектор-функция, непрерывная по совокупности переменных вместе с частными производными по $\left(z,u\right)$ $\left(\left(y,v\right)\right)$

до второго порядка включительно, $a(x)$, $b_i(t)$, $i=1,2$ — заданные дискретные вектор-функции соответствующих размерностей, $G(z)$ — заданная дважды непрерывно дифференцируемая m-мерная вектор-функция, t_0, t_1, t_2, x_0, X — заданные числа, причем разности t_2-t_0 и $X-x_0$ есть натуральные числа, $u(t,x)$ $(v(t,x))$ — r (q)-мерный дискретный вектор управляющих воздействий, со значениями из заданного непустого, выпуклого и ограниченного множества $U(V)$, т.е.

$$\begin{aligned} &u(t,x)\in U\subset R^r,\ (t,x)\in D_1,\\ &v(t,x)\in V\subset R^q,\ (t,x)\in D_2. \end{aligned} \tag{5}$$

Пару $(u(t,x),v(t,x))$ с вышеприведенными свойствами назовем допустимым управлением.

На решениях краевой задачи (1)–(4), порожденных всевозможными допустимыми управлениями определим функционал

$$S(u,v)=\phi_1(z(t_1,X))+\phi_2(y(t_2,X)). \tag{6}$$

Здесь $\phi_1(z)$, $\phi_2(y)$ — заданные дважды непрерывно дифференцируемые скалярные функции.

Допустимое управление $(u(t,x),v(t,x))$, доставляющий минимум функционалу (6), при ограничениях (1)–(5) назовем оптимальным управлением, а соответствующий процесс $(u(t,x),v(t,x),z(t,x),y(t,x))$ — оптимальным процессом.

Заметим, что условия выпуклости областей управления позволят получить специальное приращение функционала качества.

2. Специальное разложение функционала качества и необходимые условия оптимальности

Пусть $(u^o(t,x),v^o(t,x),z^o(t,x),y^o(t,x))$ — фиксированный допустимый процесс. Обозначим $(\overline{u}(t,x)=u^o(t,x)+\Delta u(t,x),\ \overline{v}(t,x)=v^o(t,x)+\Delta v(t,x),\ \overline{z}(t,x)=z^o(t,x)+\Delta z(t,x),\ \overline{y}(t,x)=y^o(t,x)+\Delta y(t,x))$ произвольный допустимый процесс и введем функции Гамильтона—Понтрягина

$$\begin{aligned} &H(t,x,z,u,\psi^o)=\psi^{o'}\cdot f(t,x,z,u),\\ &M(t,x,y,v,p^o)=p^{o'}\cdot g(t,x,y,v), \end{aligned} \tag{6}$$

где ψ^o, p^o — пока неизвестные вектор-функции соответствующих размерностей, штрих $(')$ — операция транспонирования.

Запишем приращение функционала качества

$$\Delta S(u^o,v^o)=S(\overline{u},\overline{v})-S(u^o,v^o)=\left[\phi_1(\overline{z}(t_1,X))-\phi_1(z^o(t_1,X))\right]+\left[\phi_2(\overline{y}(t_2,X))-\phi_2(y^o(t_2,X))\right]. \tag{7}$$

Ясно, что приращение $(\Delta z(t,x),\Delta y(t,x))$ является решением краевой задачи

$$\Delta z(t+1,x+1)=f(t,x,\overline{z}(t,x),\overline{u}(t,x))-f(t,x,z^o(t,x),u^o(t,x)), \tag{8}$$

$$\Delta z\left(t_0,x\right)=0,\quad x=x_0,x_0+1,...,X,$$
$$\Delta z\left(t,x_0\right)=0,\ t=t_0,t_0+1,...,t_1, \tag{9}$$

$$\Delta y\left(t+1,x+1\right)=g\left(t,x,\overline{y}\left(t,x\right),\overline{v}\left(t,x\right)\right)-g\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right)\right), \tag{10}$$

$$\Delta y\left(t_1,x\right)=G\left(x,\overline{z}\left(t_1,x\right)\right)-G\left(x,z^o\left(t_1,x\right)\right),$$
$$\Delta y\left(t,x_0\right)=0,\ \ t=t_1,t_1+1,...,t_2. \tag{11}$$

С учетом (7), (8) получаем, что

$$\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\psi_1^{o'}\left(t,x\right)\Delta z\left(t+1,x+1\right)=$$
$$=\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\left[H\left(t,x,\overline{z}\left(t,x\right),\overline{u}\left(t,x\right),\psi_1^o\left(t,x\right)\right)-H\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi_1^o\left(t,x\right)\right)\right], \tag{12}$$

$$\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\psi_2^{o'}\left(t,x\right)\Delta y\left(t+1,x+1\right)=$$
$$=\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\left[M\left(t,x,\overline{y}\left(t,x\right),\overline{v}\left(t,x\right),\psi_2^o\left(t,x\right)\right)-M\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\right]. \tag{13}$$

Сделая замену переменных $t+1=\tau$, $x+1=s$, получим

$$\begin{aligned}
&\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\psi_1^{o'}\left(t,x\right)\Delta z\left(t+1,x+1\right)=\sum_{t=t_0+1}^{t_1}\sum_{x=x_0+1}^{X}\psi_1^{o'}\left(t-1,x-1\right)\Delta z\left(t,x\right)=\sum_{x=x_0+1}^{X}\psi_1^{o'}\left(t_1-1,x-1\right)\Delta z\left(t_1,x\right)-\\
&-\sum_{x=x_0+1}^{X}\psi_1^{o'}\left(t_0-1,x-1\right)\Delta z\left(t_0,x\right)+\sum_{t=t_0}^{t_1-1}\sum_{x=x_0+1}^{X}\psi_1^{o'}\left(t-1,x-1\right)\Delta z\left(t,x\right)=\psi_1^{o'}\left(t_1-1,X-1\right)\Delta z\left(t_1,X\right)-\\
&-\psi_1^{o'}\left(t_1-1,x_0-1\right)\Delta z\left(t_1,x_0\right)+\sum_{x=x_0}^{X-1}\psi_1^{o'}\left(t_1-1,x-1\right)\Delta z\left(t_1,x\right)+\sum_{t=t_0}^{t_1-1}\psi_1^{o'}\left(t-1,X-1\right)\Delta z\left(t,X\right)-\\
&-\sum_{t=t_0}^{t_1-1}\psi_1^{o'}\left(t-1,x_0-1\right)\Delta z\left(t,x_0\right)+\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\psi_1^{o'}\left(t-1,x-1\right)\Delta z\left(t,x\right),
\end{aligned} \tag{14}$$

$$\begin{aligned}
&\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\psi_2^{o'}\left(t,x\right)\Delta y\left(t+1,x+1\right)=\sum_{t=t_1+1}^{t_2}\sum_{x=x_0+1}^{X}\psi_2^{o'}\left(t-1,x-1\right)\Delta y\left(t,x\right)=\sum_{x=x_0+1}^{X}\psi_2^{o'}\left(t_2-1,x-1\right)\Delta y\left(t_2,x\right)-\\
&-\sum_{x=x_0+1}^{X}\psi_2^{o'}\left(t_1-1,x-1\right)\Delta y\left(t_1,x\right)+\sum_{t=t_1}^{t_2-1}\sum_{x=x_0+1}^{X}\psi_2^{o'}\left(t-1,x-1\right)\Delta y\left(t,x\right)=\psi_2^{o'}\left(t_2-1,X-1\right)\Delta y\left(t_2,X\right)-\\
&-\psi_2^{o'}\left(t_2-1,x_0-1\right)\Delta y\left(t_2,x_0\right)+\sum_{x=x_0}^{X-1}\psi_2^{o'}\left(t_2-1,x-1\right)\Delta y\left(t_2,x\right)-\psi_2^{o'}\left(t_1-1,X-1\right)\Delta y\left(t_1,X\right)+\\
&+\psi_2^{o'}\left(t_1-1,x_0-1\right)\Delta y\left(t_1,x_0\right)-\sum_{x=x_0}^{X-1}\psi_2^{o'}\left(t_1-1,x-1\right)\Delta y\left(t_1,x\right)+\sum_{t=t_1}^{t_2-1}\psi_2^{o'}\left(t-1,X-1\right)\Delta y\left(t,X\right)-\\
&-\sum_{t=t_1}^{t_2-1}\psi_2^{o'}\left(t-1,x_0-1\right)\Delta y\left(t,x_0\right)+\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\psi_2^{o'}\left(t-1,x-1\right)\Delta y\left(t,x\right).
\end{aligned} \tag{15}$$

Полагая

$$N\left(\psi_2^o,z,x\right)=\psi_2^{o'}\left(t_1-1,x-1\right)y\left(t_1,x\right)\equiv\psi_2^{o'}\left(t_1-1,x-1\right)G\left(x,z\left(t_1,x\right)\right),$$

учитывая тождества (14), (15) в (7) и используя формулу Тейлора, имеем

$$\Delta S\left(u^o,v^o\right)=\frac{\partial\phi_1\left(z^o\left(t_1,X\right)\right)}{\partial z}\Delta z\left(t_1,X\right)+\frac{1}{2}\Delta z'\left(t_1,X\right)\frac{\partial^2\phi_1\left(z^o\left(t_1,X\right)\right)}{\partial z^2}\Delta z\left(t_1,X\right)+$$
$$+\frac{\partial\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y}\Delta y\left(t_2,X\right)+\frac{1}{2}\Delta y'\left(t_2,X\right)\frac{\partial^2\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y^2}\Delta y\left(t_2,X\right)+\psi_1^{o'}\left(t_1-1,X-1\right)\Delta z\left(t_1,X\right)+$$
$$+\sum_{x=x_0}^{X-1}\psi_1^{o'}\left(t_1-1,x-1\right)\Delta z\left(t_1,x\right)+\sum_{t=t_0}^{t_1-1}\psi_1^{o'}\left(t-1,X-1\right)\Delta z\left(t,X\right)+\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\psi_1^{o'}\left(t-1,x-1\right)\Delta z\left(t,x\right)-$$
$$-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}H_u\left(t,x,z^o\left(t,x\right),\bar{u}\left(t,x\right),\psi^o\left(t,x\right)\right)\Delta u\left(t,x\right)-$$
$$-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}H_z'\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi^o\left(t,x\right)\right)\Delta z\left(t,x\right)-$$
$$-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\Delta u'\left(t,x\right)H_{uz}\left(t,x,z^o\left(t,x\right),\bar{u}\left(t,x\right),\psi^o\left(t,x\right)\right)\Delta z\left(t,x\right)-$$
$$-\frac{1}{2}\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\Delta z'\left(t,x\right)H_{zz}\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi_1^o\left(t,x\right)\right)\Delta z\left(t,x\right)-$$
$$-\frac{1}{2}\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\Delta u'\left(t,x\right)H_{uu}\left(t,x,z^o\left(t,x\right),\bar{u}\left(t,x\right),\psi_1^o\left(t,x\right)\right)\Delta u\left(t,x\right)+$$
$$+o_1\left(\left\|\Delta z\left(t_1,X\right)\right\|^2\right)+o_2\left(\left\|\Delta y\left(t_2,X\right)\right\|^2\right)-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}o_3\left(\left[\left\|\Delta z\left(t,x\right)\right\|+\left\|\Delta u\left(t,x\right)\right\|\right]^2\right)+$$
$$+\psi_2^{o'}\left(t_2-1,X-1\right)\Delta y\left(t_2,X\right)-\sum_{x=x_0}^{X-1}\psi_2^{o'}\left(t_2-1,x-1\right)\Delta y\left(t_2,x\right)-N_z\left(\psi_2^o,z^o,X\right)\Delta z\left(t_1,X\right)+$$
$$+\frac{1}{2}\Delta z'\left(t_1,X\right)N_{zz}\left(\psi_2^o,z^o\left(t_1,X\right)\right)\Delta z\left(t_1,X\right)-\sum_{x=x_0}^{X-1}N_z'\left(\psi_2^o,z^o\left(t_1,x\right),x\right)\Delta z\left(t_1,x\right)-$$
$$-\frac{1}{2}\sum_{x=x_0}^{X-1}\Delta z'\left(t_1,x\right)N_{zz}\left(\psi_2^o,z^o\left(t_1,x\right),x\right)\Delta z\left(t_1,x\right)-\sum_{x=x_0}^{X-1}o_4\left(\left\|\Delta z\left(t_1,x\right)\right\|^2\right)+\sum_{t=t_1}^{t_2-1}\psi_2^{o'}\left(t-1,X-1\right)\Delta y\left(t,X\right)+$$
$$+\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\psi_2^{o'}\left(t-1,x-1\right)\Delta y\left(t,x\right)--\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}M_u'\left(t,x,y^o\left(t,x\right),\bar{v}\left(t,x\right),\psi_2^o\left(t,x\right)\right)\Delta v\left(t,x\right)-$$
$$-\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\Delta v'\left(t,x\right)M_y\left(t,x,y^o\left(t,x\right),\bar{v}\left(t,x\right),\psi_2^o\left(t,x\right)\right)\Delta y\left(t,x\right)-$$
$$-\frac{1}{2}\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\Delta z'\left(t,x\right)M_{yy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\Delta y\left(t,x\right)-$$
$$-\frac{1}{2}\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\Delta v'\left(t,x\right)M_{yy}\left(t,x,y^o\left(t,x\right),\bar{v}\left(t,x\right),\psi_2^o\left(t,x\right)\right)\Delta v\left(t,x\right)-$$
$$-\sum_{t=t_0}^{t_2-1}\sum_{x=x_0}^{X-1}o_5\left(\left[\left\|\Delta y\left(t,x\right)+\Delta v\left(t,x\right)\right\|\right]^2\right)-\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}M_y'\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\Delta y\left(t,x\right).\tag{16}$$

Здесь и далее $o_i(\alpha),i=\overline{1,5}$ означает, что $o_i(\alpha)/\alpha^2\to 0$ при $\alpha\to 0.$

Если предполагать, что $\left(\psi_1^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)$ является решением системы разностных уравнений

$$\psi_1^o\left(t-1,x-1\right)=H_z\left(t,x\right),\tag{17}$$
$$\psi_1^o\left(t_1-1,x-1\right)=G_z'\left(x,z^o\left(t_1,x\right)\right)\psi_1^o\left(t_1-1,x\right),$$

$$\psi_1^o(t-1,X-1)=0,$$

$$\psi_1^o(t_1-1,X-1)=-\frac{\partial\phi_1(z^o(t_1,X))}{\partial z}+G_z'(x,z^o(t_1,x))\psi_2^o(t_1-1,X-1),$$

$$\psi_2^o(t-1,x-1)=M_y(t,x),$$

$$\psi_2^o(t-1,X-1)=0,$$

$$\psi_2^o(t_2-1,x-1)=0,$$

$$\psi_2^o(t_2-1,X-1)=-\frac{\partial\phi_2(y^o(t_2,X))}{\partial y},$$

(18)

то формула приращения (16) примет вид

$$\Delta S(u^o,v^o)=-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}H_u'(t,x,z^o(t,x),\bar{u}(t,x),\psi_1^o(t,x))\Delta u(t,x)+$$

$$+\frac{1}{2}\Delta z'(t_1,X)\frac{\partial^2\phi_1(z^o(t_1,X))}{\partial z^2}\Delta z(t_1,X)-\frac{1}{2}\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\Big[\Delta z'(t,x)H_{zz}(t,x,z^o(t,x),\bar{u}(t,x),\psi_1^o(t,x))\Delta z(t,x)+$$

$$+2\Delta u'(t,x)H_{uz}(t,x,z^o(t,x),\bar{u}(t,x),\psi^o(t,x))\Delta z(t,x)+$$

$$+\Delta u'(t,x)H_{uu}(t,x,z^o(t,x),u^o(t,x),\psi^o(t,x))\Delta u(t,x)\Big]+$$

$$+\frac{1}{2}\Delta y'(t_2,X)\frac{\partial^2\phi_2(y^o(t_2,X))}{\partial y^2}\Delta y(t_2,X)-\frac{1}{2}\Delta z'(t_1,X)N_{zz}(\psi_2^o,z^o(t_1,X),X)\Delta z(t_1,X)-$$

$$-\frac{1}{2}\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\Big[\Delta y'(t,x)M_{yy}(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x))\Delta y(t,x)+$$

$$+2\Delta v'(t,x)M_{vy}(t,x,y^o(t,x),\bar{v}(t,x),\psi_2^o(t,x))\Delta y(t,x)+$$

$$+\Delta v'(t,x)M_{vv}(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x))\Delta vv(t,x)\Big]+o_1\left(\left\|\Delta z(t_1,X)\right\|^2\right)-$$

$$-\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}M_v'(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x))\Delta v(t,x)+$$

$$+o_2\left(\left\|\Delta y(t_2,X)\right\|^2\right)-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}o_3\left(\left[\left\|\Delta z(t,x)\right\|+\left\|\Delta u(t,x)\right\|\right]^2\right)-\sum_{x=x_0}^{X-1}o_4\left(\left\|\Delta z(t_1,x)\right\|^2\right)-$$

$$-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}o_5\left(\left[\left\|\Delta y(t,x)\right\|+\left\|\Delta v(t,x)\right\|\right]^2\right).$$

(19)

Пусть $\varepsilon\in[0,1]$ — произвольное число, а $u(t,x)\in U$, $(t,x)\in D_1$, $v(t,x)\in V$, $(t,x)\in D_2$ — произвольные допустимые управляющие функции. Используя произвольность допустимых управляющих функций $\bar{u}(t,x)$, $\bar{v}(t,x)$, вместо них возьмем такие специальные допустимые дискретные управляющие функции $\bar{u}(t,x;\varepsilon)$, $\bar{v}(t,x;\varepsilon)$, чтобы выполнялись соотношения

$$\bar{z}(t+1,x+1;\varepsilon)\equiv f(t,x,\bar{z}(t,x;\varepsilon),\bar{u}(t,x;\varepsilon))\equiv f(t,x,\bar{z}(t,x;\varepsilon),u^o(t,x))+\varepsilon\left[u(t,x)-u^o(t,x)\right], \quad (20)$$

$$\bar{z}(t_0,x;\varepsilon)=\alpha(x),\ x=x_0,x_0+1,\ldots,X,$$

$$\bar{z}(t,x_0;\varepsilon)=\beta_1(t),\ t=t_0,t_0+1,\ldots,t_1, \quad (21)$$

$$\bar{y}(t+1,x+1;\varepsilon)\equiv g(t,x,\bar{y}(t,x;\varepsilon),\bar{v}(t,x;\varepsilon))\equiv g(t,x,\bar{y}(t,x;\varepsilon),v^o(t,x))+\varepsilon\left[v(t,x)-v^o(t,x)\right], \quad (22)$$

$$\overline{y}\left(t_1,x;\varepsilon\right)=G\left(x,\overline{z}\left(t_1,x;\varepsilon\right)\right),\ y\left(t,x_0;\varepsilon\right)=\beta_2\left(t\right). \tag{23}$$

Это возможно в силу выпуклости множеств U и V.

Положим по определению

$$\alpha\left(t,x\right)=\left.\frac{\partial\overline{z}\left(t,x;\varepsilon\right)}{\partial\varepsilon}\right|_{\varepsilon=0},$$

$$\beta\left(t,x\right)=\left.\frac{\partial\overline{y}\left(t,x;\varepsilon\right)}{\partial\varepsilon}\right|_{\varepsilon=0}.$$

Учитывая условия, наложенные на правые части уравнений (1), (2), при помощи (20)–(21) получаем, что

$$\Delta z\left(t,x;\varepsilon\right)=\overline{z}\left(t,x;\varepsilon\right)-z^o\left(t,x\right)=\varepsilon\alpha\left(t,x\right)+o\left(\varepsilon;t,x\right), \tag{24}$$

$$\Delta y\left(t,x;\varepsilon\right)=\overline{y}\left(t,x;\varepsilon\right)-y^o\left(t,x\right)=\varepsilon\beta\left(t,x\right)+o\left(\varepsilon;t,x\right), \tag{25}$$

где $\alpha\left(t,x\right)$ и $\beta\left(t,x\right)$ являются решениями краевых задач

$$\alpha\left(t+1,x+1\right)=\frac{\partial f\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right)\right)}{\partial z}\alpha\left(t,x\right)+\frac{\partial f\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right)\right)}{\partial u}\left(u\left(t,x\right)-u^o\left(t,x\right)\right), \tag{26}$$

$$\alpha\left(t_0,x\right)=0,\ \alpha\left(t,x_0\right)=0, \tag{27}$$

$$\beta\left(t+1,x+1\right)=\frac{\partial g\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right)\right)}{\partial y}\beta\left(t,x\right)+\frac{\partial g\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right)\right)}{\partial v}\left(v\left(t,x\right)-v^o\left(t,x\right)\right), \tag{28}$$

$$\beta\left(t_1,x\right)=G_z\left(x,z\left(t_1,x\right)\right)\alpha\left(t_1,x\right),\ \beta\left(t,x_0\right)=0. \tag{29}$$

Учитывая разложения (24), (25), в формуле приращение (19) получим

$$\begin{aligned}
&\Delta S_\varepsilon\left(u^o\left(t,x\right),v^o\left(t,x\right)\right)=S\left(\overline{u}\left(t,x;\varepsilon\right),v^o\left(t,x;\varepsilon\right)\right)-S\left(u^o\left(t,x\right),v^o\left(t,x\right)\right)=\\
&=-\varepsilon\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}H'_u\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi_1^o\left(t,x\right)\right)\left(u\left(t,x\right)-u^o\left(t,x\right)\right)-\\
&-\varepsilon\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}M'_v\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\left(v\left(t,x\right)-v^o\left(t,x\right)\right)+\\
&+\frac{\varepsilon^2}{2}\Bigg[\alpha'\left(t_1,X\right)\frac{\partial^2\phi_1\left(z^o\left(t_1,X\right)\right)}{\partial z^2}\alpha\left(t_1,X\right)-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\Big[\alpha'\left(t,x\right)H_{zz}\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi^o\left(t,x\right)\right)\alpha\left(t,x\right)+\\
&+2\left(u\left(t,x\right)-u^o\left(t,x\right)\right)'H_{uz}\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi^o\left(t,x\right)\right)\alpha\left(t,x\right)+\\
&+\left(u\left(t,x\right)-u^o\left(t,x\right)\right)'H_{uu}\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi^o\left(t,x\right)\right)\alpha\left(t,x\right)\left(u\left(t,x\right)-u^o\left(t,x\right)\right)\Big]+\\
&+\frac{\varepsilon^2}{2}\Bigg[\beta'\left(t_2,X\right)\frac{\partial^2\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y^2}\beta\left(t_2,X\right)-\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\beta'\left(t,x\right)M_{yy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\times\\
&\times\beta\left(t,x\right)+2\left(v\left(t,x\right)-v^o\left(t,x\right)\right)'M_{vy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\beta\left(t,x\right)+\\
&+\left(v\left(t,x\right)-v^o\left(t,x\right)\right)'M_{vv}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\left(v\left(t,x\right)-v^o\left(t,x\right)\right)\Big]-\\
&-\frac{\varepsilon^2}{2}\alpha'\left(t_1,X\right)N_{zz}\left(\psi_2^o,z^o\left(t_1,X\right),X\right)\alpha\left(t_1,X\right)+o\left(\varepsilon^2\right).
\end{aligned} \tag{30}$$

Из разложения (33) в силу независимости и произвольности допустимых управлений $u(t,x)$ и $v(t,x)$ получаем справедливость следующего утверждения.

Теорема 1. Если множества U и V выпуклы, то для оптимальности допустимого управления $\left(u^o(t,x), v^o(t,x)\right)$ в задаче (1)–(6) необходимо, чтобы неравенства

$$\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1} H'_u\left(t,x,z^o(t,x),u^o(t,x),\psi_1^o(t,x)\right)\left(u(t,x)-u^o(t,x)\right)\le 0, \tag{31}$$

$$\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1} M'_v\left(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x)\right)\left(v(t,x)-v^o(t,x)\right)\le 0, \tag{32}$$

выполнялись для всех $u(t,x)\in U$, $(t,x)\in D_1$, $v(t,x)\in V$, $(t,x)\in D_2$ соответственно.

Пара соотношений (31), (36) есть аналог линеаризованного условия максимума Понтрягина для рассматриваемой задачи.

Теперь рассмотрим случай вырождения аналога линеаризованного условия максимума.

Определение. Допустимое управление $\left(u^o(t,x), v^o(t,x)\right)$ назовем квазиособым управлением в задаче (1)–(6), если для всех $u(t,x)\in U$, $(t,x)\in D_1$, $v(t,x)\in V$, $(t,x)\in D_2$ выполняются соотношения

$$\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1} H'_u\left(t,x,z^o(t,x),u^o(t,x),\psi_1^o(t,x)\right)\left(u(t,x)-u^o(t,x)\right)=0, \tag{33}$$

$$\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1} M'_v\left(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x)\right)\left(v(t,x)-v^o(t,x)\right)=0. \tag{34}$$

Случай выполнения тождеств (33), (34) назовем квазиособым случаем.

В квазиособом случае из разложения (33) вытекает справедливость утверждения.

Теорема 2. При сделанных предположениях для оптимальности квазиособого управления $\left(u^o(t,x), v^o(t,x)\right)$ необходимо, чтобы вдоль процесса $\left(u^o(t,x),v^o(t,x),z^o(t,x),y^o(t,x)\right)$ выполнялись неравенства

$$\begin{aligned}
&1)\quad \alpha'(t_1,X)\left[\frac{\partial^2\phi_1\left(z^o(t_1,X)\right)}{\partial z^2}-N_{zz}\left(\psi_2^o,z^o(t_1,X)\right)\right]\alpha(t_1,X)-\\
&-\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\Big[\alpha'(t,x)H_{zz}\left(t,x,z^o(t,x),u^o(t,x),\psi^o(t,x)\right)\alpha(t,x)+\\
&+2\left(u(t,x)-u^o(t,x)\right)' H_{uz}\left(t,x,z^o(t,x),u^o(t,x),\psi^o(t,x)\right)\alpha(t,x)+\\
&+\left(u(t,x)-u^o(t,x)\right)' H_{uu}\left(t,x,z^o(t,x),u^o(t,x),\psi^o(t,x)\right)\left(u(t,x)-u^o(t,x)\right)\Big]-\\
&-\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\beta_1'(t,x)M_{yy}\left(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x)\right)\beta_1(t,x)+\beta_1'(t_2,X)\frac{\partial^2\phi_2\left(y^o(t_2,X)\right)}{\partial y^2}\beta_1(t_2,X)\ge 0.
\end{aligned} \tag{35}$$

где $\alpha(t,x)$ есть решение краевой задачи (26)–(27), а $\beta_1(t,x)$ есть решение задачи

$$\beta_1(t+1,x+1)=g_y\left(t,x,y^o(t,x),v^o(t,x)\right)\beta_1(t,x), \tag{36}$$

$$\begin{gathered}\beta_1(t_1,x)=G_z\left(x,z^o(t_1,x)\right)\alpha(t_1,x),\\ \beta_1(t,x_0)=0,\end{gathered} \tag{37}$$

$$\begin{aligned}2)\ \beta_2'(t_2,X)\frac{\partial^2\phi_2\left(y^o(t_2,X)\right)}{\partial y^2}\beta_2(t_2,X)-\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\Big[\beta_2'(t,x)M_{yy}\left(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x)\right)\times\\ \times\beta_2(t,x)+2\left(v(t,x)-v^o(t,x)\right)'M_{vy}\left(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x)\right)\beta_2(t,x)+\\ +\left(v(t,x)-v^o(t,x)\right)'M_{vv}\left(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x)\right)\left(v(t,x)-v^o(t,x)\right)\Big]\ge 0,\end{aligned} \tag{38}$$

где $\beta_2(t,x)$ есть решение задачи

$$\beta_2(t+1,x+1)=g_z\left(t,x,y^o(t,x),v^o(t,x)\right)\beta_2(t,x)+g_v\left(t,x,y^o(t,x),v^o(t,x)\right)\left(v(t,x)-v^o(t,x)\right), \tag{39}$$

$$\begin{gathered}\beta_2(t_1,x)=0,\\ \beta_2(t,x_0)=0.\end{gathered} \tag{40}$$

Неравенства (35), (38) являются неявными необходимыми условиями оптимальности особых управлений.

Используя их, получим явное необходимое условие оптимальности.

Решение $\alpha(t,x)$ краевой задачи (26)–(27) допускает представление [8]

$$\alpha(t,x)=\sum_{\tau=t_0}^{t-1}\sum_{s=x_0}^{x-1}R_1(t,x;\tau,s)f_u\left(\tau,s,z^o(\tau,s),u^o(\tau,s)\right)\left(u(\tau,s)-u^o(\tau,s)\right), \tag{41}$$

где $R_1(t,x;\tau,s)$ $(n\times n)$ матричная функция — решение задачи

$$\begin{gathered}R_1(t,x;\tau-1,s-1)=R(t,x;\tau,s)f_z\left(\tau,s,z^o(\tau,s),u^o(\tau,s)\right),\\ R_1(t,x;\tau-1,x-1)=0\\ R_1(t,x;t-1,s-1)=0\\ R_1(t,x;t-1,x-1)=E,\end{gathered}$$

(E_1 — $(n\times n)$ единичная матрица).

Через $R_2(t,x;\tau,s)$ обозначим $(m\times m)$ матричную функцию, являющуюся решением задачи

$$\begin{gathered}R_2(t,x;\tau-1,s-1)=R_2(t,x;\tau,s)g_y\left(\tau,s,y^o(\tau,s),v^o(\tau,s)\right),\\ R_2(t,x;\tau-1,x-1)=0\\ R_2(t,x;t-1,s-1)=0\\ R_2(t,x;t-1,x-1)=E_2,\end{gathered}$$

(E_2— $(m\times m)$ единичная матрица).

Тогда решения задач (36)–(37) и (39)–(40) допускают соответственно представления

$$\beta_1(t,x)=\sum_{\tau=t_0}^{t_1-1}\sum_{s=x_0}^{x-1}Q(t,x;\tau,s)f_u(\tau,s,z^o(\tau,s),u^o(\tau,s))(u(\tau,s)-u^o(\tau,s)), \tag{42}$$

$$\beta_2(t,x)=\sum_{\tau=t_1}^{t-1}\sum_{s=x_0}^{x-1}R_2(t,x;\tau,s)g_v(\tau,s,y^o(\tau,s),v^o(\tau,s))(v(\tau,s)-v^o(\tau,s)), \tag{43}$$

где $Q(t,x;\tau,s)$ определяется формулой

$$Q(t,x;\tau,s)=R_2(t,x;t_1-1,x-1)G_z(x,z^o(t_1,x))R_1(t_1,x;\tau,s)+$$
$$+\sum_{\beta=s+1}^{x-1}R_2(t,x;t_1-1,\beta-1)G_z(x,z^o(t_1,x))R_1(t_1,\beta;\tau,s).$$

Используя представления (41), (42), будем иметь

$$\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\alpha'(t,x)H_{zz}(t,x,z^o(t,x),u^o(t,x),\psi_1^o(t,x))\alpha(t,x)=$$
$$=\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\left(\sum_{\tau=t_0}^{t-1}\sum_{s=x_0}^{x-1}R_1(t,x;\tau,s)f_u(\tau,s,z^o(\tau,s),u^o(\tau,s))(u(\tau,s)-u^o(\tau,s))\right)'\times$$
$$\times H_{zz}(t,x,z^o(t,x),u^o(t,x),\psi_1^o(t,x))\left(\sum_{\ell=t_0}^{t-1}\sum_{m=x_0}^{x-1}R(t,x;\ell,m)f_u(\ell,m,z^o(\ell,m),u^o(\ell,m))(u(m,\ell)-u^o(m,\ell))\right)'$$
$$=\sum_{\tau=t_0}^{t_1-1}\sum_{s=x_0}^{x-1}\sum_{\ell=t_0}^{t_1-1}\sum_{m=x_0}^{x-1}f_u'(\tau,s,z^o(\tau,s),u^o(\tau,s))(u(\tau,s)-u^o(\tau,s))'\times$$
$$\times\left\{\sum_{t=\max(\tau,\ell)+1}^{t_1-1}\sum_{x=\max(s,m)+1}^{X-1}R'(t,x;\tau,s)H_{zz}(t,x,z^o(t,x),u^o(t,x),\psi_1^o(t,x))R(t,x;\ell,m)\right\}\times \tag{44}$$
$$\times\Delta_{u(\ell,m)}f(\ell,m,z^o(\ell,m),u^o(\ell,m))(u(m,\ell)-u^o(m,\ell)),$$

$$\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\Delta_{v(t,x)}H_z'(t,x,z^o(t,x),u^o(t,x),\psi_1^o(t,x))\alpha(t,x)= \tag{45}$$
$$=\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\left(\sum_{\tau=t+1}^{t_1-1}\sum_{s=x+1}^{X-1}(u(\tau,s)-u^o(\tau,s))H_{uz}(\tau,s,z^o(\tau,s),u^o(\tau,s),\psi_1^o(\tau,s))R_1(\tau,s;t,x)\right)\times$$
$$\times f_u(t,x,z^o(t,x),u^o(t,x))(u(t,x)-u^o(t,x)),$$

$$\alpha'(t_1,X)\frac{\partial^2\phi_1(z^o(t_1,X))}{\partial z^2}\alpha(t_1,X)=\sum_{\tau=t_0}^{t_1-1}\sum_{s=x_0}^{X-1}\sum_{\ell=t_0}^{t_1-1}\sum_{m=x_0}^{X-1}(u(\tau,s)-u^o(\tau,s))'f_u'(\tau,s,z^o(\tau,s),u^o(\tau,s))\times \tag{46}$$
$$\times R_1(t,X;\tau,s)\frac{\partial^2\phi_1(z^o(t_1,X))}{\partial z^2}R_1(t_1,X;\ell,m)f_u(\ell,m,z^o(\ell,m),u^o(\ell,m))(u(m,\ell)-u^o(m,\ell)),$$

$$\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\beta_1'(t,x)M_{yy}(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x))\beta_1(t,x)=$$
$$=\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\left(\sum_{\tau=t_0}^{t_1-1}\sum_{s=x_0}^{x-1}Q_1(t,x;\tau,s)f_u(\tau,s,z^o(\tau,s),u^o(\tau,s))(u(\tau,s)-u^o(\tau,s))\right)'\times$$
$$\times M_{yy}(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x))\left(\sum_{\ell=t_0}^{t_1-1}\sum_{m=x_0}^{x-1}Q(t,x;\ell,m)f(\ell,m,z^o(\ell,m),u^o(\ell,m))(u(m,\ell)-u^o(m,\ell))\right)=$$

$$=\sum_{\tau=t_0}^{t_1-1}\sum_{s=x_0}^{X-1}\sum_{\ell=t_0}^{t_1-1}\sum_{m=x_0}^{X-1}\left(u\left(\tau,s\right)-u^o\left(\tau,s\right)\right)f_u'\left(\tau,s,z^o\left(\tau,s\right),u^o\left(\tau,s\right)\right)\times$$

$$\times\left\{\sum_{t=t_1}^{t_2-1}\sum_{x=\max\left(s,m\right)+1}^{X-1}Q_1'\left(t,x;\tau,s\right)M_{yy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\right\}f_u\left(\ell,m,z^o\left(\ell,m\right),u^o\left(\ell,m\right)\right)\times \tag{47}$$

$$\times\left(u\left(m,\ell\right)-u^o\left(m,\ell\right)\right),$$

$$\beta_1'\left(t_2,X\right)\frac{\partial^2\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y^2}\beta_1\left(t_2,X\right)=\sum_{\tau=t_0}^{t_1-1}\sum_{s=x_0}^{X-1}\left(u\left(\tau,s\right)-u^o\left(\tau,s\right)\right)'f\left(\tau,s,z^o\left(\tau,s\right),u^o\left(\tau,s\right)\right)\times$$

$$\times Q'\left(t_2,X;\tau,s\right)\frac{\partial^2\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y^2}\times$$

$$\times\sum_{\ell=t_0}^{t_1-1}\sum_{m=x_0}^{X-1}Q\left(t_2,X;\ell,m\right)f\left(\ell,m,z^o\left(\ell,m\right),u^o\left(\ell,m\right)\right)\left(u\left(m,\ell\right)-u^o\left(m,\ell\right)\right). \tag{48}$$

Далее при помощи представления (46) получаем, что

$$\beta_2'\left(t_2,X\right)\frac{\partial^2\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y^2}\beta_2\left(t_2,X\right)=\sum_{\tau=t_1}^{t_2-1}\sum_{s=x_0}^{X-1}\sum_{\ell=t_1}^{t_2-1}\sum_{m=x_0}^{X-1}\left(v\left(\tau,s\right)-v^o\left(\tau,s\right)\right)'g_v'\left(\tau,s,y^o\left(\tau,s\right),v^o\left(\tau,s\right)\right)\times \tag{49}$$

$$\times R_2'\left(t_2,X;\tau,s\right)\frac{\partial^2\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y^2}R_2\left(t_2,X;\ell,m\right)g_v\left(\ell,m,y^o\left(\ell,m\right),v^o\left(\ell,m\right)\right)\left(v\left(m,\ell\right)-v^o\left(m,\ell\right)\right),$$

$$\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\left(v\left(t,x\right)-v^o\left(t,x\right)\right)'M_{vy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\beta_2\left(t,x\right)= \tag{50}$$

$$=\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\left[\sum_{\tau=t+1}^{t_2-1}\sum_{s=x+1}^{X-1}\left(v\left(\tau,s\right)-v^o\left(\tau,s\right)\right)'M_{vy}'\left(\tau,s,y^o\left(\tau,s\right),v^o\left(\tau,s\right),\psi_2^o\left(\tau,s\right)\right)R_2\left(\tau,s;t,x\right)\right]\times$$

$$\times g_v\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right)\right)\left(v\left(t,x\right)-v^o\left(t,x\right)\right),$$

$$\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\beta_2'\left(t,x\right)M_{yy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)\beta_2\left(t,x\right)=$$

$$=\sum_{\tau=t_1}^{t_2-1}\sum_{s=x_0}^{X-1}\sum_{\ell=t_1}^{t_2-1}\sum_{m=x_0}^{X-1}\left(v\left(\tau,s\right)-v^o\left(\tau,s\right)\right)'g_v'\left(\tau,s,y^o\left(\tau,s\right),v^o\left(\tau,s\right)\right)\times$$

$$\times\left\{\sum_{t=\max\left(\tau,\ell\right)+1}^{t_2-1}\sum_{x=\max\left(s,m\right)+1}^{X-1}R_2'\left(t,x;\tau,s\right)M_{yy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)R_2\left(t,x;\ell,m\right)\right\}\times \tag{51}$$

$$\times g_v\left(\ell,m,y^o\left(\ell,m\right),v^o\left(\ell,m\right)\right)\left(v\left(m,\ell\right)-v^o\left(m,\ell\right)\right).$$

Введем матричные функции $K\left(\tau,s,\ell,m\right)$, $L\left(\tau,s,\ell,m\right)$ посредством формул

$$K\left(\tau,s,\ell,m\right)=\sum_{t=\max\left(\tau,\ell\right)+1}^{t_1-1}\sum_{x=\max\left(s,m\right)+1}^{X-1}R_1'\left(t,x;\tau,s\right)H_{zz}\left(t,x,z^o\left(t,x\right),u^o\left(t,x\right),\psi_1^o\left(t,x\right)\right)R\left(t,x;\ell,m\right)-$$

$$-R_1'\left(t_1,X;\tau,s\right)\frac{\partial^2\phi_1\left(z^o\left(t_1,X\right)\right)}{\partial z^2}R_1\left(t_1,X;\ell,m\right)+$$

$$+\sum_{t=t_1}^{t_2-1}\sum_{x=\max\left(s,m\right)+1}^{X-1}Q_1'\left(t,x;\tau,s\right)M_{yy}\left(t,x,y^o\left(t,x\right),v^o\left(t,x\right),\psi_2^o\left(t,x\right)\right)-$$

$$-Q'\left(t_2,X;\tau,s\right)\frac{\partial^2\phi_2\left(y^o\left(t_2,X\right)\right)}{\partial y^2}Q\left(t_2,X;\ell,m\right),$$

$$L(\tau,s,\ell,m)=-R_2'(t_2,X;\tau,s)\frac{\partial^2\phi_2(y^o(t_2,X))}{\partial y^2}R_2(t_2,X;\ell,m)+$$

$$+\sum_{t=\max(\tau,\ell)+1}^{t_2-1}\sum_{x=\max(s,m)+1}^{X-1}R_2'(t_2,X;\tau,s)M_{yy}(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x))R_2(t_2,X;\ell,m).$$

Учитывая тождества (47)–(48) и выражения для $K(\tau,s,\ell,m)$, $L(\tau,s,\ell,m)$, неравенства (38), (39) записываются в виде

$$\begin{gathered}\sum_{\tau=t_0}^{t_1-1}\sum_{s=x_0}^{X-1}\sum_{\ell=t_0}^{t_1-1}\sum_{m=x_0}^{X-1}\left(u(m,\ell)-u^o(m,\ell)\right)' f_u'(\tau,s,z^o(\tau,s),u^o(\tau,s))K(\tau,s,\ell,m)\times\\ \times f_u(\ell,m,z^o(\ell,m),u^o(\ell,m))\left(u(m,\ell)-u^o(m,\ell)\right)+\\ +\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\left(u(t,x)-u^o(t,x)\right)' H_{uu}(t,x,z^o(t,x),u^o(t,x),\psi_1^o(t,x))\left(u(t,x)-u^o(t,x)\right)+\\ +2\sum_{t=t_0}^{t_1-1}\sum_{x=x_0}^{X-1}\left[\sum_{\tau=t+1}^{t_1-1}\sum_{s=x+1}^{X-1}\left(u(\tau,s)-u^o(\tau,s)\right)H_{uz}'(\tau,s,z^o(\tau,s),u^o(\tau,s),\psi_1^o(\tau,s))R_1(\tau,s;t,x)\right]\times\\ \times f_u(t,x,z^o(t,x),u^o(t,x))\left(u(t,x)-u^o(t,x)\right)\le 0,\\ \sum_{\tau=t_1}^{t_2-1}\sum_{s=x_0}^{X-1}\sum_{\ell=t_1}^{t_2-1}\sum_{m=x_0}^{X-1}\left(v(\tau,s)-v^o(\tau,s)\right)' g_v'(\tau,s,y^o(\tau,s),v^o(\tau,s))L(\tau,s,\ell,m)g_v(\ell,m,y^o(\ell,m),v^o(\ell,m))+\\ +2\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\left[\sum_{\tau=t+1}^{t_2-1}\sum_{s=x+1}^{X-1}\left(v(\tau,s)-v^o(\tau,s)\right)' M_{vy}'(\tau,s,y^o(\tau,s),v^o(\tau,s),\psi_2^o(\tau,s))R_2(\tau,s;t,x)\right]\times\\ \times g_v(t,x,y^o(t,x),v^o(t,x))\left(v(t,x)-v^o(t,x)\right)+\\ +\sum_{t=t_1}^{t_2-1}\sum_{x=x_0}^{X-1}\left(v(t,x)-v^o(t,x)\right)' M_{vv}(t,x,y^o(t,x),v^o(t,x),\psi_2^o(t,x))\left(v(t,x)-v^o(t,x)\right)\le 0.\end{gathered}\tag{52}$$

Теорема 3. Пусть множества U и V выпуклы. Тогда для оптимальности квазиособого, управления $(u^o(t,x),v^o(t,x))$ необходимо, чтобы неравенства (52), (53) выполнялись для всех $u(t,x)\in U$, $(t,x)\in D_1$, $v(t,x)\in V$, $(t,x)\in D_2$ соответственно.

Отметим, что неравенства (52) и (53) являются конструктивными, но довольно общими необходимыми условиями оптимальности квазиособых управлений. Тем не менее, из них, в частности, можно получить ряд более легко проверяемых, но менее информативных одноточечных необходимых условий оптимальности квазиособых управлений, используя произвольность допустимых управлений $u(t,x),v(t,x)$.

Приведем одно из них.

Следствие. При сделанных предположениях для оптимальности квазиособого управления $(u^o(t,x),v^o(t,x))$ в рассматриваемой задаче необходимо, чтобы соотношения

$$\begin{gathered}(u-u^o(\theta,\xi))\left[f_u'(\theta,\xi,z^0(\theta,\xi),u(\theta,\xi))K(\theta,\xi,\theta,\xi)f_u(\theta,\xi,z^0(\theta,\xi),u(\theta,\xi))+\right.\\ \left.+H_{uu}(\theta,\xi,z^o(\theta,\xi),u^o(\theta,\xi),\psi_1^o(\theta,\theta))\right](u-u^0(\theta,\xi))\le 0,\\ (v-v^o(\theta,\xi))\left[g_u'(\theta,\xi,y^0(\theta,\xi),v^0(\theta,\xi))L(\theta,\xi,\theta,\xi)g_u(\theta,\xi,y^0(\theta,\xi),v(\theta,\xi))+\right.\\ \left.+M_{vv}(\theta,\xi,y^o(\theta,\xi),v^o(\theta,\xi),\psi_2^o(\theta,\theta))\right](v-v^0(\theta,\xi))\le 0,\end{gathered}$$

выполнялись для всех $u\in U$, $(\theta,\xi)\in D_1$, $v\in V$, $(\theta,\xi)\in D_2$, соответственно.

Заключение

В работе рассматривается одна ступенчатая задача оптимального управления, описываемая дискретным аналогом системы Гурса—Дарбу (модель Форназини—Маркезини). При предположении выпуклости области управления, применяя модифицированный вариант метода приращений, установлено необходимое условия оптимальности первого порядка в форме линеаризованного условие максимума, которое, в отличие от непрерывного случая, не является следствием дискретного принципа максимума.

Далее изучен случай вырождения линеаризованного условия максимума (квазиособый случай). При помощи новых специальных формул приращений критерия качества выведены необходимые условия оптимальности квазиособых управлений. Заметим, что условия оптимальности для таких задач оптимального управления очень мало установлены. В дальнейшем предполагается исследование поставленной задачи в предположении открытости областей управления, замкнутости областей управления, а также при наличии наложенных на состояния системы различных функциональных ограничений типа равенств и неравенств. Представляет интерес также исследование поставленной задачи с негладким критерием качества, а также изучение задачи на минимакс. Одной из актуальных задач является доказательство в рассматриваемой задаче достаточного условия оптимальности типа условий Кротова с помощью формализма Беллмана—Кротова.

Авторы выражают благодарность рецензентам за полезные замечания.

Литература

1. Fornazini E., Marchesini G. State-space realization theory of two-dimensional filters // IEEE Trans. Automat. Contr. 1976. Vol. 21, no. 4. P. 484–492. DOI: 10.1109/TAC.1976.1101305.
2. Kaczorek T. Two-dimensional linear systems. Springer, 1985. 399 p. DOI: 10.1007/BFb0005617.
3. Гайшун И.В., Хоанг В. Условия полной управляемости дискретных двухпараметрических систем // Дифференц. уравнения. 1991. Т. 27, № 2. С. 187–193.
4. Гайшун И.В. Многопараметрические системы управления. Минск: Изд-во ИМ НАН Белоруси, 1996. 200 с.
5. Васильев О.В., Кириллова Ф.М. Об оптимальных процессах в двухпараметрических дискретных системах // Докл. АН СССР. 1967. Т. 175, № 1. С. 17–19.
6. Васильев О.В. К оптимальным процессам в непрерывных и дискретных двухпараметрических системах // Информ. сб. трудов ВЦ Иркутского госуниверситета. 1968. № 2. С. 87–104.
7. Степанюк Н.Н. Некоторые задачи управляемости и наблюдаемости двухпараметрических дискретных систем // Дифференц. уравнения. 1978. Т. 14, № 12. С. 2190–2195.
8. Мансимов К.Б. Дискретные системы. Баку: Изд–во БГУ, 2013. 151 с.
9. Мансимов К.Б., Насияти М.М. Необходимые условия оптимальности в одной многоэтапной дискретной задаче управления // Матем. и компьютерное моделирование. 2011. № 5. С. 162–179.
10. Насияти М.М. Условия оптимальности в ступенчатых дискретных двухпараметрических задачах управления // Автореф. дисс. на соиск. уч. степени доктора философии по математике. Баку. 2015. 22 с.
11. Мансимов К.Б., Марданов М.Дж. Качественная теория оптимального управления системами Гурса—Дарбу. Баку: Изд-во ЭЛМ, 2010. 362 с.

12. Мансимов К.Б. Достаточные условия оптимальности типа условий Кротова в дискретных двухпараметрических системах // Автоматика и телемеханика. 1985. № 8. С. 15–20.
13. Величенко В.В. Оптимальное управление составными системами // Докл. АН СССР. 1967. Т. 176. № 4. С. 754–756.
14. Захаров Г.К. Оптимизация ступенчатых систем с управляемыми условиями перехода // Автоматика и телемеханика. 1993. № 1. С. 32–36.
15. Никольский М.С. Об одной вариационной задаче с переменной структурой // Вестник МГУ. Серия: Вычислительная математика и кибернетика. 1987. № 4. С. 31–34.
16. Мансимов К.Б., Исмайлов Р.Р. Об условиях оптимальности в одной ступенчатой задаче управления // Журнал вычисл. матем. и матем. физики. 2006. Т. 46, № 10. С. 1758–1770.
17. Розова В.Н. Оптимальное управление ступенчатыми системами // Вестник Российского университета дружбы народов. 2006. № 1. С. 27–32.
18. Марданов М.Дж., Мансимов К.Б., Меликов Т.К. Исследование особых управлений и необходимые условия оптимальности второго порядка в системах с запаздыванием. Баку: Изд-во ЭЛМ, 2013. 363 с.
19. Габасов Р., Кириллова Ф.М. Особые оптимальные управления. М.: Либроком, 2013. 256 с.

Мансимов Камиль Байрамали оглы, д.ф.-м.н., профессор, кафедра математической кибернетики, Бакинский государственный университет (Баку, Азербайджан)

Мамедова Туркан Фикрет кызы, диссертант, Институт систем управления НАН Азербайджана (Баку, Азербайджан)

DOI: 10.14529/cmse200205

QUASISINGULAR CONTROL IN A ONE STEP CONTROL PROBLEM OF DISCRETE TWO-PARAMETRIC SYSTEMS

© 2020 K.B. Mansimov[1,2], T.F. Mamedova[2]
[1]*Baku State University (Z. Khalilov 23, Baku, Az1148 Azerbaijan),*
[2]*Institute of Control Systems of NAS of Azerbaijan*
(B. Wahabzadeh 9, Baku, Az1141 Azerbaijan)
E-mail: kamilbmansimov@gmail.com, kmansimov@mail.ru
Received: 11.07.2019

We study one stepwise (i.e., multi-stage) optimal control problem of a terminal type by a quality functional, described by discrete two-parameter systems of equations of the Fornasini–Marchezini type under the assumption of convexity of the control domains. A discrete two-parameter system of equations of the Fornasini–Marchezine type is a difference analogue of the system second-order hyperbolic equations (sometimes such systems of equations in the Western literature are also called 2D systems). Using a modified analogue of the increment method, a special decomposition of the second-order quality functional, using linearized difference systems of equations is obtained.

Using one version of the increment method, the first-order necessary optimality condition is established in the form of a linearized (differential) maximum condition. The case of degeneration of the linearized maximum condition (a quasi-singular case) separately is studied. Using constructive verifiable quadratic necessary optimality conditions for quasi-singular controls, using representations of solutions of linearized difference systems of equations using special formulas for incrementing the quality functional.

Keywords: Fornazini–Marchesini type discrete two-parameter system , linearized necessary optimality condition, step problem, optimal control, quasi-singular control, convex control domain.

FOR CITATION

Mansimov K.B., Mamedova T.F. Quasisingular Control in a One Step Control Problem of Discrete Two-Parametric Systems. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 2. P. 68–83. (in Russian) DOI: 10.14529/cmse200205.

References

1. Fornazini E., Marchesini G. State-space realization theory of two-dimensional filters. IEEE Trans. Automat. Contr. 1976. Vol. 21, no. 4. P. 484–492. DOI: 10.1109/TAC.1976.1101305.
2. Kaczorek T. Two-dimensional linear systems. Springer, 1985. 399 p. DOI: 10.1007/BFb0005617.
3. Gayshun I.V., Khoang V. Conditions for complete controllability of discrete two-parameter systems. Differential Equations. 1991. Vol. 27, no. 2. P. 187–193. (in Russian)
4. Gayshun I.V. Multiparameter control systems. Minsk, Publishing of the National Academy of Sciences of Belarus, 1996. 200 p. (in Russian)
5. Vasil'yev O.V., Kirillova F.M. On Optimal Processes in Two-Parameter Discrete Systems. Dokl. AN SSSR. 1967. Vol. 175, no. 1. P. 17–19. (in Russian)
6. Vasil'yev O.V. To optimal processes in continuous and discrete two-parameter systems. Inform. sb. trudov vts. Irkutskogo Gosuniversiteta. 1968. no. 2. P. 87–104. (in Russian)

7. Stepanyuk N.N. Some problems of controllability and observability of two-parameter discrete systems. Differents. uravneniya. 1978. Vol. 14, no. 12. P. 2190–2195. (in Russian)
8. Mansimov K.B. Discrete systems. Baku, Publishing of the Baku State University, 2013. 151 p. (in Russian)
9. Mansimov K.B., Nasiyati M.M. Necessary Optimality Conditions in a Multi-Stage Discrete Control Problem. Mathematical and computer modelling. 2011. no. 5. P. 162–179. (in Russian)
10. Nasiyati M.M. Optimality conditions in stepwise discrete two-parameter control problems. Avtoref. diss. na soisk. uch. stepeni doktora filosofii po matematike. Baku. 2015. 22 p. (in Russian)
11. Mansimov K.B., Mardanov M.J. Qualitative theory of optimal control of Goursat-Darboux systems. Baku, ELM, 2010. 362 p. (in Russian)
12. Mansimov K.B. Sufficient conditions for optimality of the type of Krotov conditions in discrete two-parameter systems. Autom. Remote Control. 1985. no. 8. P. 15–20. (in Russian)
13. Velichenko V.V. Optimal control of composite systems. Dokl. AN SSSR. 1967. Vol. 176, no. 4. P. 754–756. (in Russian)
14. Zakharov G.K. Optimization of step systems with controllable transition conditions. Autom. Remote Control. 1993. no. 1. P. 32–36. (in Russian)
15. Nikol'skiy M.S. On a variational problem with a variable structure. Moscow University Computational Mathematics and Cybernetics. 1987. no. 4. P. 31–34. (in Russian)
16. Mansimov K.B., Ismaylov R.R. Conditions of Optimality in a Single Step Control Problem. Comput. Math. Math. Phys. 2006. Vol. 46, no. 10. P. 1758–1770.
17. Rozova V.N. Optimal control of incremental systems. Vestnik Rossiyskogo universiteta druzhby narodov. 2006. no. 1. P. 27–32. (in Russian)
18. Mardanov M.J., Mansimov K.B., Melikov T.K. Investigation of special controls and necessary conditions for second-order optimality in systems with delay. Baku, ELM, 2013. 363 p. (in Russian).
19. Gabasov R., Kirillova F.M. Special optimal controls. Moscow, Librocom, 2013. 256 p. (in Russian)

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки), 05.13.17 – Теоретические основы информатики (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,

   ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
Том 9, № 2
2020

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 29.05.2020. Дата выхода в свет 22.06.2020. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 10,23. Тираж 500 экз. Заказ 128/191. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.