ПРОГРАММИРОВАНИЕ, 2020, № 2, с. 67–72

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 681.3.06

ВЫЧИСЛЕНИЕ ИНВОЛЮТИВНЫХ БАЗИСОВ И БАЗИСОВ ГРЁБНЕРА ИСПОЛЬЗУЯ ТАБЛИЧНОЕ ПРЕДСТАВЛЕНИЕ ПОЛИНОМОВ

© 2020 г. Д. А. Янович*

Объединенный институт ядерных исследований 141980 Дубна Московской области, Россия

**E-mail: yan@jinr.ru*

Поступила в редакцию 19.08.2019 г.
После доработки 20.10.2019 г.
Принята к публикации 19.11.2019 г.

При работе с полиномами обычно используются такие представления данных, как списки термов, геобакеты, кучи. В работе предпринята попытка по новому взглянуть на проблему представления полиномов для задач вычисления инволютивных базисов и базисов Грёбнера систем нелинейных полиномиальных уравнений. Также новый подход позволяет передать часть этой вычислительной задачи на GPU, что открывает перспективы решения более сложных проблем.

DOI: 10.31857/S0132347420020120

1. ВВЕДЕНИЕ

Обозначения, используемые в работе

X – множество переменных $X = \{x_1, \ldots, x_n\}$;

$\mathbb{R} = \mathbb{K}[X]$ – кольцо многочленов над кольцом $\mathbb{K}$;

f, g, h, q, r – многочлены из $\mathbb{R}$;

a, b, c – элементы из $\mathbb{K}$;

F, G, H – конечные подмножества из $\mathbb{R}$;

$\langle F \rangle$ – идеал из $\mathbb{R}$ порожденный F;

$\mathbb{Z}_{\geq 0}$ – множество неотрицательных чисел;

$\mathbb{M} = \{x_1^{d_1} \cdots x_n^{d_n} | d_i \in \mathbb{Z}_{\geq 0}\}$ – множество мономов из $\mathbb{R}$;

u, v, w, s, t – множество мономов или термов;

U, V, W – конечные подмножества из $\mathbb{M}$;

$\deg_i(u)$ – степень переменной x_i в u;

$\deg(u)$ – полная степень монома u;

$\succ$ – допустимое мономиальное упорядочение с порядком переменных $x_1 \succ \ldots \succ x_n$;

$\mathrm{lt}(f)$ – лидирующий терм относительно упорядочения $\succ$;

$\mathrm{lm}(f)$ – лидирующий моном f;

$\mathrm{lm}(F) = \{\mathrm{lm}(f) | f \in F\}$ – множество лидирующих мономов из F;

$\mathrm{lcm}(F)$ – наименьшее общее кратное множества мономов из $\mathrm{lm}(F)$;

$u|v$ означает, что моном u делит моном v.

Инволютивные базисы и базисы Грёбнера

Базис Грёбнера (и инволютивный базис) являются известным математическим инструментом, широко применяемым в задачах алгебраической геометрии, биоинформатики, вычислительной физики, робототехники и криптографии: везде, где необходимо решать системы нелинейных алгебраических, разностных и дифференциальных уравнений.

Определение и алгоритм построения базисов Грёбнера были предложены Б. Бухбергером [1].

Определение 1. [1, 2] *Пусть заданно допустимое мономиальное упорядочение. Конечное подмножество $G = \{g_1, \ldots, g_m\}$ элементов идеала I называется его базисом Грёбнера, если*

$$\langle \mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_m) \rangle = \langle \mathrm{lm}(I) \rangle.$$

В данной работе для вычисления базиса Грёбнера применяется инволютивный подход, который описан в работах [3, 4].

Если, некоторым самосогласованным способом запретить деление по некоторым переменным, называемыми *немультипликативными* и разрешить по другим, называемыми *мультипликативными*, то мы получим сужение операции деления – *инволютивное деление*.

Определение 2. [3, 4] *Мы будем говорить, что на множестве мономов $\mathbb{M}$ определено инволютивное деление L, если для любого конечного не пустого $U \subset \mathbb{M}$ и для любого $u \in U$ задано $M_L(u, U) \subseteq X$, генерирующее подмоноид $L(u, U) \subset M$, состоящий из мономов с переменными из $M_L(u, U)$, удовлетворяющий следующим условиям*:

(a) Из $u, v \in U$ и $uL(u,U) \cap vL(v,U) \neq \varnothing$ следует $u \in vL(v,U)$ или $v \in uL(u,U)$;

(b) Из $v \in U$ и $v \in uL(u,U)$ следует $L(v,U) \subseteq L(u, U)$;

(c) Из $u \in V$ и $V \subseteq U$ следует $L(u,U) \subseteq L(u,V)$ для всех $u \in V$.

Элементы $M_L(u,U)$ называются L-мультипликативными для u, элементы $NM_L(u, U) = X \backslash M_L(u, U)$ называются L-немультипликативными. Если $w \in uL(u,U)$, то u называется $(L-)$инволютивным делителем w и обозначается $u|_L w$. В свою очередь, моном w называется $(L-)$кратным u.

Определение 3. [3, 4] *Множество мономов называется L – инволютивным, если*

$$(\forall u \in U)(\forall w \in \mathbb{M})(\exists v \in U)[v|_L uw].$$

Определение 4. [3] *Авторедуцированное множество F называется частично инволютивным до монома v в допустимом мономиальном упорядочении $\succ$ если выполнено следующее условие*

$$(\forall f \in F)(\forall u \in \mathbb{M})(\mathrm{lm}(f) \succ v)[\mathrm{NF}_L(fu, F) = 0],$$

где NF_L означает инволютивную нормальную форму, т.е. полином, ни один из мономов термов которого не делится лидирующими мономами множества F.

Таким образом, используя конструктивное и нётерово деление (например деление Жане), пополняя частичный базис ненулевыми инволютивными нормальными формами немультипликативных продолжений элементов частично инволютивного множества F, мы в конце концов получим его инволютивный базис.

Приведем упрощенный алгоритм 1 для вычисления инволютивного базиса, не использующий критерии для пропуска редукций p к нулю. Полный алгоритм приведен в [3].

Алгоритм 1. Вычисление инволютивного базиса

Input: $F, L, \prec$

Output: T – инволютивный базис F

```
1: T := ∅
2: Q := F
3: while Q ≠ ∅ do
4:     P := {p ∈ Q | lm(p) = min(lm(Q)),
5:         NF_L(p, T) ≠ 0}
6:     T := T ∪ P
7:     Q := Q \ P
8:     for all {p ∈ P} do
9:         Q := Q ∪ {p · NM_L(p, T)}
10:    od
11:    for all {r ∈ T} do
12:        if lm(r) ⊐ lm(p) then
13:            Q := Q ∪ {r};
14:            T := T \ {r}
15:        fi
16:    od
17: od
```

Связь между инволютивным базисом и базисом Грёбнера определяется следующей теоремой:

Теорема 1. [3] *Инволютивный базис, авторедуцированный в смысле обычного деления, является редуцированным базисом Грёбнера.*

2. МОТИВАЦИЯ

Получение базисов Грёбнера (и инволютивных базисов) является тяжелой вычислительной задачей как по расходу оперативной памяти, так и по времени, необходимому для завершения алгоритма. Наибольшую трудность представляют вычисления нормальной формы полинома, которые занимают более 90% времени работы. Также известно, что сложность можно снизить сначала вычислив базисы с коэффициентами из кольца модулярных чисел (или с числами Фарея [5]) и потом восстановив базис с коэффициентами из $\mathbb{Q}$. Предлагаемое представление полиномов делает вычисления более подходящими современной архитектуре процессоров и адаптированными к способам работы с оперативной памятью. Также данное представление позволяет перенести часть операций с полиномами на GPU.

3. ПРЕДСТАВЛЕНИЕ ПОЛИНОМОВ

При вычислениях с модулярными коэффициентами в известных представлениях данных для полиномов от многих переменных (списках термов, кучах [6], геобакетах [7] и т.п.) большую часть места в оперативной памяти занимают именно мономы. В задачах из области криптографии [8, 9] и задачах SAT-выполнимости могут присутствовать тысячи переменных, что ведет к “разбуханию” этих структур данных до таких масштабов, что 16Гб RAM не хватает даже на несколько секунд вычислений. В работе [10] автор уже предлагал один из методов снижения потребления памяти, развитие идей этой работы и привело к разработке нового представления полиномов, основанного на следующих идеях:

• нам не надо представлять *все возможные* комбинации мультииндексов, в задачах вычисления базисов Грёбнера множество возможных мономов ограничено широко известной фигурой *staircase* [11];

• в процессе вычислений в множестве полиномов Q, которые необходимо рассмотреть в ходе

алгоритма, образуется большое количество полиномов, содержащих много одинаковых мономов.

Подсчитав количество различных мономов, встречающихся в ходе вычислений базисов на примерах из [12] (см. таблицу 1), можно утверждать, что оно очень мало по сравнению с теоретической оценкой, которую можно найти в работе [13]. Наибольшее количество мономов, встреченных автором в примере *noon9*: 131860, а теоретическая оценка составляет $\binom{9+17}{9} = 3124550$.

Представим все полиномы, которые находятся в частичном базисе и в множестве Q, в виде единой *таблицы* T_K, столбцам которой сопоставим мономы, в данный момент входящие в базис, а строкам – сами полиномы. В ячейках таблицы находятся коэффициенты при мономах.

Для обеспечения произвольного доступа к коэффициентам полинома по мономам и для упрощения работы с данными к таблице добавим еще несколько структур:

• связанный список мономов $L_{\prec}$, отсортированный по убыванию по упорядочению $\prec$, каждому элементу сопоставим номер его столбца в таблице T_K;

• хэш-таблицу H_{col}, по ключу-моному выдающему номер его столбца в таблице T_K;

• хэш-таблицу H_{mon}, по номеру столбца таблицы T_K выдающую моном, ему соответствующий.

4. ДЕТАЛИ РЕАЛИЗАЦИИ

Для задач вычисления базисов Грёбнера и инволютивных базисов представление полиномов должно обеспечивать высокую эффективность следующих операций, перечисленных в порядке снижения важности:

• редукция полиномов (как по лидирующим, так и по хвостовым термам), включающая поиск подобных термов;

• умножение полинома на переменную (продолжение);

• нахождении старшего монома в полиноме

Умножение полинома на переменную фактически является переносом всех его коэффициентов в другие столбцы таблицы, соответствующие мономам, которые получаются при умножении каждого монома полинома на переменную. Это является достаточно сложной операцией (см. алгоритм 2), которая потенциально приводит к необходимости расширения таблицы при добавлении новых столбцов, соответствующих мономам, которых еще не было в полиномах частичного базиса. С течением времени все необходимые для вычислений мономы будут найдены и изменения таблицы прекратятся. Очевидно, что этот же алгоритм применяется для умножения полинома на моном.

Алгоритм 2. Умножение полинома на переменную

Input: i – строка полинома в T_K,
x_j – переменная, на которую умножаем,
i' – строка произведения

1: **for all** k – номер столбца T_K, $T_K[i][k] \neq 0$ **do**
2: $u = H_{mon}[k] \cdot x_j$
3: **if** $u \notin H_{col}$ **then**
4: **if** T_K – заполнена **then**
5: расширяем T_K с запасом
6: **fi**
7: l – свободный столбец T_K
8: $H_{col}[l] = u$
9: $H_{mon}[u] = l$
10: в $L_{\prec}$ вставляем (u, l)
11: **fi**
12: $T_K[i'][H_{col}[u]] = T_K[i][k]$
13: **od**

Редукция полинома f по g выполняется по алгоритму 3.

Алгоритм 3. Редукция полиномов

Input: f – полином, i – его строка в T_K,
t_f – терм, который надо редуцировать,
g – полином, j – его строка в T_K,
t_g – терм, по которому надо редуцировать

1: находим $u \in \mathbb{M}, a, b \in \mathbb{K} : b \cdot t_f = a \cdot t_g \cdot u$
2: составляем временный полином в строке j', умножая g на u
3: **for all** k – номер столбца T_K **do**
4: $T_K[i][k] = b \cdot T_K[i][k] - a \cdot T_K[j'][k]$
5: **od**

После некоторого количества редукций обязательно надо проверить плотность таблицы и осуществить компрессию убрав нулевые столбцы.

Одним из преимуществ предлагаемого представления является ориентированность на архитектуру современных процессоров и учет кэширования данных памяти. Процессоры намного быстрее обрабатывают данные из кэша, чем из оперативной памяти, разница может достигать нескольких десятков раз. Поэтому очень важно, чтобы процессор мог предсказать, какие данные и в каком порядке будут обработаны, чтобы успеть доставить их в кэш. Если выполнять редукцию по алгоритму 3, то процессор уверен, ка-

Таблица 1. Количество мономов в вычислениях

Пример	#	Пример	#
assur44	544	eco11	6215
butcher8	586	eco12	19077
chemequs	77	extcyc5	907
chemkin	1124	extcyc6	4365
cohn3	651	f744	1611
cpdm5	515	f855	4404
cyclic5	168	fabrice24	478
cyclic6	583	filter9	2673
cyclic7	4738	hairer2	10111
cyclic8	32122	hairer3	11690
d1	658	hcyclic7	6208
des18_3	367	hcyclic8	42504
des22_24	506	hf744	8069
discret3	1101	hf855	59707
dl	3520	hietarinta1	870
eco8	482	i1	1043
eco9	1222	ilias13	3609
eco10	2548	ilias_k_2	1579

кие данные следует получить, что выражается в значительном снижении кэш-промахов. Данное утверждение можно проверить, используя инструмент `valgrind-tool=chegrind` [14]. Для примера *reimer6* списочное представление имеет 9.6% кэш-промахов, а табличное – всего 5.4%, аналогичная картина наблюдается и для остальных примеров.

Также код, более подходящий для современных процессоров, должен содержать как можно меньше условий, которые могут приводить к сбросу конвейера выполнения инструкций и вследствие этого приводящих к уменьшению скорости работы суперскалярных блоков выполнения CPU. Поэтому редукции с нулевыми коэффициентами не обходятся, а выполняются так же, как и значащие редукции, поскольку условный их обход приводил бы к понижению производительности.

Нахождение старшего монома в полиноме. После проведения операции редукции может понадобиться обновить указатель на старший моном полинома. Нахождение его производится простой итерацией по списку $L_{\prec}$ до первого ненулевого коэффициента в соответствующем столбце строки полинома в таблице T_K.

5. ВОЗМОЖНОСТЬ ВЫЧИСЛЕНИЯ БАЗИСОВ НА GPU

Современные GPU являются мощнейшими процессорами для вычислений общего типа, об-

Таблица 2. Типичное профилирование GPU-вычислений

Время(%)	Вызов	Процедура
15.48%	470890	gpuReduceEle(..)
84.83%	941780	cudaMemcpy
14.01%	470890	cudaLaunchKernel

ладающими некоторыми особенностями, которые надо учитывать при адаптации алгоритмов для них. Во-первых, нужно учитывать архитектуру SIMD: на GPU запускается так называемое ядро вычислений (kernel), которое во множество потоков обрабатывает одинаковым образом входные данные, организованные в batch и warp разделы (используется терминология CUDA от фирмы NVidia). Во-вторых, наиболее эффективно происходят вычисления с большими объемами данных, поскольку GPU имеют собственную локальную память, в которую копируются исходные данные и из которой они потом должны быть ско-

Таблица 3. Сравнение списочного и табличного представлений

Пример	t_{list}	t_{table}
cyclic7	2.39	5.16
cyclic8	273.54	559.85
dl	14.6	56.91
eco10	2.86	6.84
eco11	42.68	59.64
eco12	1538.1	**631.47**
extcyc6	3.8	9.56
f855	2.82	14.8
hairer2	6.65	10.17
hcyclic7	2.56	10.99
hcyclic8	279.67	1426.53
hf744	2.38	16.72
hf855	259.93	1462.96
ilias13	10.88	35.01
katsura9	17.29	21.56
katsura10	225.78	240.00
katsura11	2513.21	2619.63
noon7	3.94	22.69
noon8	139.72	791.98
noon9	5797.72	*
redcyc7	2.17	5.03
redeco11	20.91	34.56
redeco12	244.33	376.01
reimer6	1.63	1.98
reimer7	228.27	**146.60**

пированы обратно в память, принадлежащую CPU. Из этого возникает третье условие эффективной адаптации: передач CPU ↔ GPU должно быть как можно меньше.

На данный момент автором реализован простейший вариант адаптации: перенос части алгоритма 3 на GPU. В процессе вычислений на CPU конструируются два полинома в табличном представлении, которые передаются на GPU, редуцируются ядром

```
__global__ void gpuReduceEle
        (mtype* aWhat, mtype* aByWhat,
        int aN,
        const mtype aA, const mtype aB,
        const mtype aM,
        const int aK, const mtype aMod) {
    const int i = blockIdx.x * blockDim.x
            + threadIdx.x;
    if (i < aN) {
       mtype c = aWhat[i] * aA
                    + aByWhat[i] * aB;
        gpuBarretRed(c, aM, aK, aMod);
        aWhat[i] = c;
        aByWhat[i] = 0;
    }
}
```

и передаются обратно. Такая реализация является скорее концептом, поторый позволяет удостовериться, что вычисления инволютивных базисов и базисов Грёбнера с использованием GPU возможно. В таблице 2 показаны результаты профилирования запуска вычислений примера *reimer6* на видеокарте начального уровня NVidia GT710, способной запускать одно ядро вычислений использующее до 1024 потоков обработки данных с использованием CUDA 10.0 (то, что сумма процентов больше 100 является особенностью работы профилировщика от NVidia). Видно, что собственно сами вычисления в ядре занимают лишь малую часть, основные же затраты идут на копирование данных и накладные расходы на запуск вычислений. Времена вычислений на GPU приведены в данной работе не будут, поскольку целью было показать именно принципиальную возможность вычислений используя предложенное представление данных, так как довольно очевидно, что ни одно из классических представлений полиномов не позволит получить даже такого результата.

6. РЕЗУЛЬТАТЫ ТЕСТОВЫХ ПРОГОНОВ

В таблице 3 представлены результаты сравнения реализаций алгоритма вычисления инволютивных базисов с использованием табличного и классического списочного представлений полиномов. В программах различались только представления данных, остальная реализация (ввод-вывод, работа с инволютивными делениями и т.д.) одинакова. Вычисления велись на компьютере с CPU Intel Core i5-3470@3.20GHz, RAM 8Gb под управлением Ubuntu 16.04LTS. В качестве хэш-таблицы использовалась *spp::sparse_hash_map* от Google [15], примеры взяты из [12]. Видно, что табличное представление сравнимо со списочным, в большинстве случаев медленнее, но в двух из представленных примеров существенно быстрее.

7. ЗАКЛЮЧЕНИЕ

Несмотря на то, что в большинстве примеров при вычислениях на CPU текущая реализация табличного представления оказалось медленнее списочного, автор считает данное представление полиномов весьма перспективным как с точки зрения ускорения расчетов на CPU, так и с точки зрения переноса вычислений на GPU.

СПИСОК ЛИТЕРАТУРЫ

1. *Бухбергер Б.* Базисы Грёбнера. Алгоритмический метод в теории полиномиальных идеалов // Компьютерная алгебра. Символьные и алгебраические вычисления. Москва: Мир, 1986. С. 331–372.
2. *Кокс Д., Литтл Дж., О'ши Д.* Идеалы, многообразия и алгоритмы. Введение в вычислительные аспекты алгебраической и коммутативной алгебры: Пер. с англ. // М.: Мир, 2000. 687 с., ил.

3. *Gerdt V.P., Blinkov Yu.A.* Involutive Bases of Polynomial Ideals // Math. Comp. Sim. 1998. V. 45. P. 519–542.
4. *Гердт В.П., Блинков Ю.А.* Инволютивное деление мономов // Программирование. 1998. Т. 6. С. 22–24.
5. *Янович Д.А.* Параллельное модулярное вычисление базисов Грёбнера и инволютивных базисов // Программирование. 2013. Т. 39. № 2. С. 75–80.
6. *Johnson S.C.* Sparse polynomial arithmetic // ACM SIGSAM Bulletin. 1974. V. 8 (3). P. 63–71.
7. *Yan T.* The Geobucket Data Structure for Polynomials // Journal of Symbolic Computation. 1998. V. 25. P. 285–293.
8. *Cid C., Murphy S., Robshaw M.J.B.* Small Scale Variants of the AES // 12th International Workshop, FSE 2005, Paris, France, February 21–23, Revised Selected Papers. 2005. P. 145–162.
9. *Jean-Charles Faugère, Antoine Joux* Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases // 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17321. Proceedings. 2003. P. 44–60.
10. *Янович Д.А.* Компактное представление полиномов для алгоритмов вычисления базисов Грёбнера и инволютивных базисов // Программирование. 2015. Т. 41. № 2. С. 63–68.
11. *Zimmermann P., Casamayou A., Cohen N., Connan G. et al.* Computational Mathematics with SageMath // SIAM. 2018. P. 203–205.
12. *Verschelde J.* The Database with Test Examples. http://www.math.uic.edu/ jan/demo.html
13. *Monagan M.* Sparse polynomial division using a heap // Journal of Symbolic Computation. 2011. V. 46. P. 807–822
14. http://www.valgrind.org
15. https://github.com/sparsehash/sparsehash