ФИЗИКА ЭЛЕМЕНТАРНЫХ ЧАСТИЦ И АТОМНОГО ЯДРА
2020. Т. 51. ВЫП. 4. С. 697–706

ОПЫТ ПОСТРОЕНИЯ НЕВЫРОЖДЕННЫХ 2-ПЕТЛЕВЫХ ОБОБЩЕННЫХ IBP-ТОЖДЕСТВ

А. А. Радионов, Ф. В. Ткачев *

Институт ядерных исследований РАН, Москва

Обсуждается проблема систематического построения дифференциальных операторов для обобщенной IBP-редукции на 2-петлевом уровне. Глубоко оптимизированное программное обеспечение позволяет эффективно строить такие операторы для первых невырожденных 2-петлевых случаев. Обнаружено, что наиболее эффективный подход состоит в построении так называемых частичных операторов, заметно более простых, чем полные, и меняющих степень только одного из полиномов.

We discuss the problem of constructing differential operators for the generalized IBP reduction algorithms at the 2-loop level. A deeply optimized software allows one to efficiently construct such operators for the first non-degenerate 2-loop cases. The most efficient approach is found to be via the so-called partial operators that are much simpler than the complete ones, and that affect the power of only one of the polynomials in the product.

PACS: 02.70.-c

ВВЕДЕНИЕ

IBP-алгоритмы, основанные на так называемых тождествах интегрирования по частям [1], являются стандартным инструментом многопетлевых вычислений. В [2] был предложен их весьма общий вариант, основанный на теореме Бернштейна [3]. Этот вариант привлекателен своей универсальностью и чисто алгебраическим характером. Однако до сих пор не удавалось систематически находить соответствующие дифференциальные операторы (см. ниже). В вычислениях использовалось только найденное в [2] явное решение для однопетлевого случая (так называемый минимальный BT-подход для 2-петлевых интегралов [4]), а также были опубликованы некоторые частичные результаты, связанные с 2-петлевыми интегралами [5]. Нам удалось преодолеть 2-петлевой барьер для построения полных тождеств такого типа. Это не

*E-mail: fyodor.tkachov@gmail.com

значит, что уже сейчас можно свободно строить такие операторы для любых 2-петлевых диаграмм, но по крайней мере можно изучать специфику задачи на примере настоящих невырожденных 2-петлевых интегралов.

О ПРЕДРАССУДКАХ-ОБРЕМЕНЕНИЯХ

Поскольку нахождение требуемых тождеств является чрезвычайно громоздким в вычислительном плане, в силу вступает «правило вертикального трансцендирования междисциплинарных границ»: для наилучших результатов должны быть прозрачны (для того, кто решает задачу) границы между разными уровнями решения задачи, потому что проблемы, возникающие на одних уровнях, могут иметь оптимальное решение совсем на других; это правило по сути является обобщением известного опыта разработки сложного программного обеспечения [6]. В нашем случае уровни такие:

1) уровень прикладной задачи, условно говоря, физика: понимание проблемы, ее формулировка, постановка математической задачи для разработки методов;

2) на уровне математики есть теоретическая часть (теоремы существования и пр.) и часть реализационная (алгоритмы, численные методы и т. п.);

3) на уровне программирования есть подуровень архитектуры (представление данных, структурирование программной системы) и подуровень кодирования.

В обсуждаемой очень громоздкой задаче приходится иметь в виду все эти уровни: делать выбор на уровне, скажем, математики, понимая особенности уровня программирования, и наоборот, поскольку ни на одном из уровней нетерпимы обременения, порождаемые неявными предрассудками.

Например, на уровне приложения можно услышать, что «только аналитические ответы выживут в вечности». Но значащие цифры — конечная цель расчета — не менее «живучи». При этом аналитические ответы могут занимать многие страницы и выражаться в терминах специальных функций, которые еще надо довести до значащих цифр. А в некоторых важных случаях нужны не «аналитические ответы», а кусок кода, который можно встроить в генератор Монте-Карло.

Есть предрассудки-обременения на уровне формулировки. Точкой отсчета здесь является то, что в конечном счете нужно уметь, задав значения масс и импульсов, выдавать численный ответ для вычисляемой амплитуды. Все, что сверх того (аналитические формулы и пр.), — это приятный, но необязательный бонус.

Есть предрассудки-обременения на уровне математики. Широко распространен род веры в магию «красивой» математики (базисы Гребнера и т. п.), реализованной на «мощных» системах компьютерной алгебры (CAS). Однако мощь таких систем оборачивается накладными расходами из-за семан-

тического зазора (semantic gap) между их входными языками и машинным уровнем.

Есть предрассудки-обременения на уровне программирования. Это прежде всего вера в мощь промышленных инструментов (C++, Java, ...), которые на самом деле несут огромный груз избыточной сложности [13]. Кроме того, распространено мнение, что CAS сделает за вас всю грязную работу, хотя чаще при наивном программировании просто случится так называемый промежуточный взрыв (intermediate blow-up). Кстати говоря, в нашем случае такой «взрыв» имеет место в весьма злокачественном варианте.

ОБОБЩЕННЫЕ IBP-АЛГОРИТМЫ

IBP-алгоритмы были открыты давно [1] и хорошо известны. Обратим внимание только на один аспект используемых там тождеств. С точки зрения фейнмановской параметризации (интеграл по стандартному симплексу размерности $L-1$, где L — это число линий) такие тождества связывают интегралы, причем одни являются интегралами по частям границы области интегрирования для других. Этот паттерн возникнет в обобщенных IBP-алгоритмах.

В 1950-х гг. И. М. Гельфанд и др. ввели [8] аналитическое продолжение по μ для выражений x^μ, трактуемых как обобщенные функции, на любые, в том числе отрицательные нецелые μ с помощью интегрирования по частям. В размерной регуляризации, которая родственна такому аналитическому продолжению (см. [9]), интегрирование по частям для аналитического продолжения применяется в работе [10]. В 1954 г. Гельфанд высказал гипотезу, что для любого полинома $P(x_0, x_1, \ldots)$ от любого числа переменных $x_0, x_1, \ldots$ комплексная степень $P^\mu(x_0, x_1, \ldots)$ существует как обобщенная функция в смысле аналитического продолжения по μ. Затем Бернштейн доказал [3], что для любого такого P существует дифференциальный оператор конечного порядка $D(\partial_0, \partial_1, \ldots)$, где ∂_i — это частная производная по x_i, такой что

$$D(\partial_0, \partial_1, \ldots) P^\mu(x_0, x_1, \ldots) = b(\mu) P^{\mu-1}(x_0, x_1, \ldots), \quad b(\mu) \neq 0. \qquad (1)$$

Можно сказать, что оператор D понижает на единицу степень, в которую возводится полином P. При этом коэффициенты D суть полиномы от x_i, μ и от коэффициентов полинома P. Смысл доказательства Бернштейна в том, что если брать полиномы P возрастающего порядка с неизвестными коэффициентами, то после сведения тождества к системе однородных линейных уравнений число неизвестных растет быстрее числа уравнений, при этом из-за дифференцирований происходит перемешивание мономов разного порядка по x, так что при достаточно больших порядках полинома P по x_i, μ, ∂_i система насыщается и у нее появляются ненулевые решения.

Для приложений к вычислению многопетлевых интегралов требуется (тривиальное) обобщение формулы (1) на случай нескольких полиномов P_i:

$$D\left(\partial_0, \partial_1, \ldots\right) \prod_i P_i^{\mu_i}\left(x_0, x_1, \ldots\right) = b\left(\mu\right) \prod_i P_i^{\mu_i - 1}\left(x_0, x_1, \ldots\right), \quad b\left(\mu\right) \neq 0. \tag{2}$$

Для стандартных интегралов с двумя и более петлями при $D \neq 4$ таких полиномов два: один, «топологический», зависит только от топологии диаграммы Фейнмана, другой, «кинематический», является линейной комбинацией размерных физических параметров — квадратов масс и пр. (В нескалярных моделях числители подынтегральных выражений содержат дополнительные полиномы, но присутствие последних не влияет ни на тождества (2), ни на сценарий их использования.)

Обобщенная IBP-редукция состоит в том, чтобы применить эту формулу несколько раз, заменяя произведение двух полиномов в комплексных степенях на левую часть формулы (2) и избавляясь от производных интегрированием по частям. В результате происходит «поднятие» степеней полиномов до значений, безопасных для численного интегрирования, а граничные члены, появляющиеся после интегрирования по частям, будут интегралами по частям границы исходного симплекса, как и в случае стандартных IBP-алгоритмов.

ОСНОВНАЯ ЗАДАЧА

Для интересных приложений в пертурбативной КТП нужно научиться строить оператор D по заданной паре полиномов P. Поскольку для D есть только теорема существования, естественно оттолкнуться от метода неопределенных коэффициентов: сначала делается предположение о порядках D по ∂_i и x_i, затем D выписывается с неопределенными коэффициентами, наконец, уравнение (2) сводится к системе однородных линейных уравнений относительно этих коэффициентов. Результат Бернштейна гарантирует, что, начиная с достаточно больших порядков D, система будет иметь искомые ненулевые решения с $b\left(\mu\right) \neq 0$.

Для однопетлевых интегралов топологический полином равен 1, а кинематический квадратичен по x, и решение для D можно выписать в явном виде, см. [2]. Этот частный случай оказался полезным даже для 2-петлевых вычислений, см. [4].

ПРИМЕР ОТ GRACE

Коллаборация GRACE показала [5] первые примеры, связанные с 2-петлевыми интегралами, однако такие, где фигурирует только один, кинематиче-

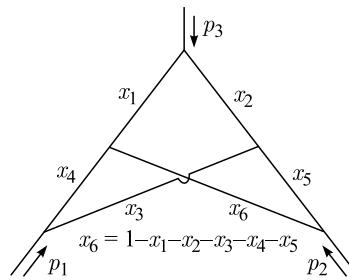
ский, полином (это связано со спецификой задач, изучавшихся в [5]):

$$\int\limits_0^1 dx_1 \cdots dx_6 \delta\left(1 - \sum x\right) \frac{1}{\left(P + i\varepsilon\right)^2}, \qquad (3)$$
$$P = f_1 s_1 + f_2 s_2 + f_3 s_3 - f_0 \sum m_i^2 x_i,$$

где

$$f_0 = (x_1 + x_2)(x_3 + x_4 + x_5 + x_6) + (x_3 + x_4)(x_5 + x_6),$$
$$f_1 = (x_1 + x_2 + x_5 + x_6) x_3 x_4 + x_1 x_3 x_6 + x_2 x_4 x_5,$$
$$f_2 = (x_1 + x_2 + x_3 + x_4) x_5 x_6 + x_2 x_3 x_6 + x_1 x_4 x_5,$$
$$f_3 = (x_3 + x_4 + x_5 + x_6) x_1 x_2 + x_2 x_4 x_6 + x_1 x_3 x_5,$$

$$s_1 = 10, \quad s_2 = 0, \quad s_3 = 0, \quad m_i^2 = i.$$



Если в результатах, приведенных в [5], сократить громоздкие общие числовые факторы, то D примет такой вид (заметим, что проверять такие тождества — но не строить — легко в любой системе компьютерной алгебры общего назначения):

$$D = -\frac{1}{20}\partial_1^2 + \frac{1}{10}\partial_1\partial_2 - \frac{1}{20}\partial_1\partial_3 + \frac{1}{20}\partial_1\partial_4 + \frac{13}{20}\partial_1\partial_5 - \frac{1}{20}\partial_2^2 + \frac{1}{20}\partial_2\partial_3 -$$
$$- \frac{1}{20}\partial_2\partial_4 + \frac{7}{20}\partial_2\partial_5 - \frac{1}{20}\partial_3^2 + \frac{1}{10}\partial_3\partial_4 + \frac{13}{20}\partial_3\partial_5 - \frac{1}{20}\partial_4^2 + \frac{7}{20}\partial_4\partial_5 +$$
$$+ 6\partial_5 - \frac{603}{20}\partial_5^2 + 3\mu\partial_5 - 3x_1\partial_2\partial_5 + 57x_1\partial_5^2 - 3x_2\partial_2\partial_5 + 24x_2\partial_5^2 -$$
$$- 3x_3\partial_3\partial_5 + 18x_3\partial_5^2 - 3x_4\partial_3\partial_5 + 45x_4\partial_5^2 + 3x_5\partial_5^2. \qquad (4)$$

Оператор D оказался здесь довольно простым, однако сложность построения D с двумя полиномами примерно соответствует сложности задачи с одним полиномом, равным произведению двух исходных, и сложность расчета резко возрастает при учете топологического полинома, имеющего в 2-петлевом случае второй порядок.

Нам удалось найти вычислительные процедуры, позволяющие довольно эффективно строить операторы D по крайней мере для простых из числа невырожденных 2-петлевых случаев. Кратко опишем эти процедуры, начиная с уровня кодирования.

УРОВЕНЬ КОДИРОВАНИЯ

В качестве языка реализации мы выбрали популярный диалект семейства Oberon [11] — Component Pascal [12]. Однако речь здесь идет не только о выборе языка, но и об идеологии программирования. См. подробнее [13], здесь лишь несколько замечаний.

Oberon — это наследник языка Pascal по прямой линии (через язык Modula-2). Его можно было бы назвать Pascal'88 или, еще лучше, Ultra Pascal, от «nec plus ultra». Потому что это вершина эволюции императивных языков, примерно так же, как позиционная десятичная нотация является вершиной числовых нотаций. Oberon — это предельно минималистичный язык без каких-либо особенных «фич», зато в нем не только реализована строгая статическая типизация, но она и распространена на динамические записи, причем сделано это без накладных расходов для обычных императивных программ. В частности, там исключены целые классы привычных, для тех, кто программирует на C++, ошибок времени выполнения, в первую очередь нарушения целостности памяти (segment violation etc.). Невозможно переоценить эти свойства в задачах компьютерной алгебры.

УРОВЕНЬ КОМПЬЮТЕРНОЙ АЛГЕБРЫ

Мы использовали минималистичный «движок» (engine), являющийся развитием библиотеки BEAR 2, ранее с успехом примененной в весьма громоздких расчетах по распадам b-кварка [14]. В использованной сейчас версии 3 и интерфейсы, и алгоритмы изменились в достаточной мере, чтобы оправдать переименование движка: теперь это Gulo («россомаха»). Это по-прежнему очень небольшая библиотека, спроектированная, однако, так, чтобы поддерживать алгебраические вычисления практически неограниченного объема, не ограничивать программиста в отношении представления данных и позволять ему управлять памятью и файлами с такой степенью детализации, с какой это необходимо в конкретной задаче (в простейшем случае — полностью скрывая такие детали). В задачах рассматриваемого типа проектировщик должен иметь полный контроль над представлением данных, над алгоритмами арифметики, над алгоритмами сортировки и т. д. Библиотека Gulo (как и ранее BEAR) позволяет разделить «алгебру» и представление данных, не закрывая путей к дальнейшей оптимизации.

С точки зрения производительности новая версия, Gulo, существенно превосходит предыдущую на очень больших выражениях. В рамках изначально

заложенных проектных требований наша библиотека выглядит оптимальной, и опыт подтверждает, что принятые архитектурные решения позволяют библиотеке не мешать механизмам кеширования файлов, используемых операционными системами, что важно, поскольку при неудачных решениях взаимодействие с файловой системой может обернуться весьма существенными накладными расходами.

Второй программный компонент — это средства решения систем уравнений, возникающих в IBP-алгоритмах, включая обсуждаемые здесь обобщенные. Zipper2 — это новая версия для программы Zipper, использованной в расчетах [14]. Это в своей основе решатель для больших систем однородных линейных уравнений с дополнительными модификациями. Как уже отмечалось, стартовой точкой выбраны алгоритмы гауссовского типа, которые мы оптимизируем, опираясь на статистику и на изучение алгоритмов «высокого уровня», формулируемых в терминах базиса Гребнера и т. п. Заметим, что комбинация Gulo/Zipper2 существенно эффективнее старой версии BEAR/Zipper на больших задачах, хотя уже старая версия в тестах, проведенных в контексте проекта [14], оказалась в 8 раз быстрее, чем программное обеспечение для того же класса задач, написанное на C++.

В алгоритмах гауссовского типа есть две фазы, прямой и обратный ход (высокоуровневые математические методы по сути вносят дополнительную организацию в эти фазы с использованием информации о специфической алгебраической структуре исходной задачи). Мы применяем некоторые оптимизации, поэтому разделение на две фазы не вполне чистое. Во всяком случае две фазы равно громоздки, и на стыке между ними имеет место серьезный blow-up. Однако для второй фазы нам удалось найти прием (так называемый R-trick), который фактически устраняет фазу обратного хода за счет отказа от поиска общего решения (такая постановка задачи была бы предрассудкомобременением, см. обсуждение в начале доклада). Устранение полной фазы обратного хода приводит к весьма существенному ускорению.

Вторая важная оптимизация касается арифметики числовых коэффициентов. Здесь имеется большое пространство для экспериментирования. (Подобные эксперименты крайне затруднены при работе с C++, поскольку они требуют изменения представления данных в сильно динамических структурах, и отладка превращается в бесконечную агонию в условиях отсутствия защиты со стороны компилятора.)

Наконец, мы пока едва тронули предусмотренные в Gulo и Zipper2 опции по оптимизации хранения выражений. Здесь остается значительный резерв.

ЧАСТИЧНЫЕ D-ОПЕРАТОРЫ

Третья оптимизация заключается в том, чтобы отказаться от поиска оператора D, понижающего степень всех полиномов сразу, как в уравнении (2),

в пользу частичных операторов, понижающих степень только одного из них:

$$\begin{aligned} D_0\,(x,\mu,\partial)\,P_0^{\mu_0}P_1^{\mu_1} &= b_0\,(\mu)\,P_0^{\mu_0-1}P_1^{\mu_1}, \quad b_0\,(\mu)\neq 0,\\ D_1\,(x,\mu,\partial)\,P_0^{\mu_0}P_1^{\mu_1} &= b_1\,(\mu)\,P_0^{\mu_0}P_1^{\mu_1-1}, \quad b_1\,(\mu)\neq 0. \end{aligned} \tag{5}$$

Из двух таких частичных операторов всегда можно построить один полный, удовлетворяющий (2):

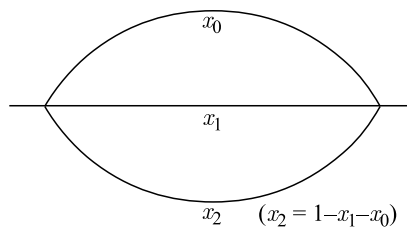
$$\begin{gathered} D\,(x,\mu,\partial) = D_1\,(x,\mu-e_0,\partial)\,D_0\,(x,\mu,\partial)\,,\\ b\,(\mu) = b_0\,(\mu)\,b_1\,(\mu-e_0). \end{gathered} \tag{6}$$

Однако система уравнений, получающаяся для частичного оператора, вовсе не выглядит проще, чем для полного. Кроме того, априори отнюдь не очевидно, что простейшие (т. е. имеющие минимальные порядки по x и ∂) полные операторы допускают представление в виде композиции частичных. Но оказывается, что ситуация именно такова: частичные операторы по сложности оказываются как бы факторами полных и находятся гораздо быстрей. Тем более, что в петлевых интегралах из теорий с нескалярными частицами подынтегральные выражения содержат два полинома в разных степенях, и выгодно «поднимать» степень только одного оператора, тем более, что частичный оператор проще полного.

ПРИМЕРЫ ВЫЧИСЛЕНИЯ ОПЕРАТОРОВ D

Описанные оптимизации привели к ускорению расчетов на несколько порядков величины даже по сравнению с версиями, сделанными на основе программного обеспечения (BEAR 2 + Zipper), которое никак нельзя назвать неэффективным. При этом еще практически не задействованы плановые оптимизации, связанные с хранением данных на диске (предаллокация дефрагментированных файлов и т. п.). Это серьезный резерв, так как речь идет о данных весьма большого объема и дисковый ввод–вывод по интегралу очень затратен.

Чистый эффект состоит в том, что уже на данном этапе оказалось возможным находить D-операторы для настоящих 2-петлевых интегралов, причем делать это достаточно эффективно. Простейший такой интеграл соответствует диаграмме «sunset» (2-петлевая собственная энергия в скалярной модели $g\phi^4$):



Примеры явных выражений для двух частичных операторов для этой диаграммы приведены в [16].

Построение частичных операторов для соответствующей пары полиномов делается сейчас за время порядка 1 с на бытовом ноутбуке. С такой производительностью становятся возможными интересные игры. Например, можно восстановить точную аналитическую зависимость по крайней мере для одного из кинематических параметров. На указанной веб-странице [16] дан пример одного из частичных операторов для той же диаграммы с восстановленной таким способом зависимостью от k^2 для случая равных масс. Ответы малопригодны для чтения человеком, но этого и не предполагается. Важно то, что их легко проверять прямой подстановкой в определяющее уравнение (2) в любой CAS, например, в Wolfram Mathematica.

ЗАКЛЮЧЕНИЕ

Итак, в этой весьма громоздкой задаче открылось неожиданно много серьезных оптимизаций. Даже при том, что не исчерпаны возможности плановых опций и не исследованы возможности параллелизации, производительность уже сейчас повышена на несколько порядков величины даже по сравнению с предыдущей версией, основанной на базе программного обеспечения (BEAR 2 + Zipper), существенно более эффективного по сравнению с системами на основе CAS общего назначения. Таким образом открывается дорога к задачам следующего этапа — к построению операторов D для 2-петлевых интегралов более сложных топологий и к разработке программ конкретного интегрирования с использованием обобщенных IBP-алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. *Tkachov F. V.* // Phys. Lett. B. 1981. V. 100. P. 65–68.
2. *Tkachov F. V.* // Nucl. Instr. Meth. A. 1997. V. 389. P. 309–313.
3. *Бернштейн И. Н.* // Функц. анализ и его прил. 1972. Т. 6, вып. 4. С. 26–40.
4. *Passarino G.* // Nucl. Phys. B. 2001. V. 619, No. 1. P. 257–312;
   Ferroglia A. et al. // Nucl. Phys. B. 2003. V. 650. P. 162–228.
5. *Fujimoto J., Kaneko T.* GRACE and Loop Integrals // PoS LL. 2012. V. 2012. P. 047;
   Fujimoto J. https://indico.desy.de/indico/event/4362/session/13/contribution/9/material/slides/0.pdf.
6. *Fox J.* Software and Its Development. Prentice Hall, 1982.
7. *Леньшина Н. Д., Радионов А. А., Ткачев Ф. В.* // ЭЧАЯ. 2020. Т. 51, вып. 4. С. 644.
8. *Гельфанд И. М., Шилов Г. Е.* Обобщенные функции и действия над ними. Обобщенные функции. Вып. 1. 2-е изд. М.: Физматгиз, 1959.
9. *Speer E. R.* // J. Math. Phys. 1968. V. 9. P. 1404.

10. *'t Hooft G., Veltman M.* // Nucl. Phys. B. 1972. V. 44. P. 189–213.
11. *Wirth N., Gutknecht J.* Project Oberon: Design of an Operating System and Compiler. Addison-Wesley, 1992.
12. Component Pascal Language Report. Zurich: Oberon microsystems, 2001.
13. *Tkachov F. V.* // Proc. of the 2nd CPP Symp., Tokyo, Nov. 27–30, 2001; J. Phys. Conf. Ser. 2014. V. 523. P. 012011.
14. *Czarnecki A. et al.* // Phys. Rev. Lett. 2006. V. 96. P. 171803.
15. *Akritas A. G.* Elements of Computer Algebra with Applications. Wiley, 1989.
16. http://www.inr.ac.ru/∼ftkachov/projects/genIBP/