УДК 517.955.8

ЛИНЕЙНО-БЫСТРЫЕ АЛГОРИТМЫ ДЛЯ ВЫЧИСЛЕНИЯ ДИСКРЕТНОГО ПРЕОБРАЗОВАНИЯ ЛЕЖАНДРА И РЕШЕНИЯ УРАВНЕНИЯ БЮРГЕРСА

А.В. Трусов

Международный институт теории прогноза землетрясений и математической геофизики Российской академии наук

Приводится новый алгоритм для вычисления дискретного преобразования Лежандра. Классические алгоритмы работают со скоростью, пропорциональной N^{2d} (N – количество точек в решетке, d – размерность пространства). Быстрое преобразование Лежандра имеет время $O((N \log_2 N)^d)$, новый алгоритм имеет время $O(N^d)$. Решение уравнения Бюргерса может быть сведено к выполнению преобразования Лежандра. В статье рассматривается другой подход к решению уравнения Бюргерса; приводятся новые алгоритмы, в одномерном случае имеющие время работы $O(N)$ и позволяющие получить решение одномерного уравнения Бюргерса путем вычисления траекторий материальных точек.

LINEAR-FAST ALGORITHMS FOR CALCULATING DISCRETE LEGENDRE TRANSFORMS AND SOLVING THE BURGERS EQUATION

A.V. Troussov

International Institute of Earthquake Prediction Theory and Mathematical Geophysics, Russian Academy of Sciences

A new algorithm for the computation of discrete Legendre transforms is discussed. Classical solutions have a running time proportional to N^{2d} (N is the size of the spatial grid and d is the space dimension). Fast Legendre transform has running time $O((N \log_2 N)^d)$. The new algorithm has a running time $O(N^d)$. The solution of Burgers' equation may be obtained by applying the Legendre transform. Here we consider also another approach to the solution of Burgers' equation. We introduce some linear-time algorithms, which make it possible to solve a one-dimensional version of the Burgers' equation by calculating paths of fluid particles.

ПРЕОБРАЗОВАНИЕ ЛЕЖАНДРА И ЕГО ПРИМЕНЕНИЕ К РЕШЕНИЮ УРАВНЕНИЯ БЮРГЕРСА

Преобразование Лежандра играет важную роль в выпуклом анализе и часто необходимость в его применении возникает в задачах математической физики. В случае если U – конечномерная функция, являющаяся гладкой, строго выпуклой и растущей на бесконечности быстрее линейной функции, то преобразование Лежандра можно определить следующим образом:

$$\tilde{U}(x^*) = \max_{x \in R^n}(\langle x^*, x\rangle U(x)).$$

Преобразование Лежандра применяется для решения гиперболических консервативных систем [1]. В работах [2–5] преобразование Лежандра применено для решения уравнения Бюргерса, используемого при изучении формирования галактик. Численное моделирование в этих работах потребовало высокого разрешения (более 10^6 точек) и усреднения по тысячам реализаций случайных начальных данных.

Прогресс в численном решении уравнения Бюргерса в одномерном и многомерном случаях связан с так называемым *быстрым преобразованием Лежандра* (см. [4–6]). Этот алгоритм позволяет найти дискретное преобразование Лежандра для функции на d-мерной решетке, содержащей N^d узлов за $O((N\log_2 N)^d)$ операций. Альтернативный численный метод для многомерного уравнения Бюргерса [5] использует только $O(N^d \log_2 N)$ операций.

Конструкция быстрого преобразования Лежандра (Fast Legendre Transform) была выполнена по аналогии с быстрым преобразование Фурье (Fast Fourier Transform). По-видимому, именно эта аналогия с одним из самых известных вычислительных алгоритмов, которая вначале позволила придумать алгоритм, явилась и главным психологическим тормозом, который помешал открыть то обстоятельство, что задача может быть решена более эффективно. В данной работе приводится новый алгоритм, который позволяет вычислять дискретное преобразование Лежандра,используя только $O(N^d)$ операций. Другими словами, время работы этого алгоритма линейно зависит от объема исходных данных (по-английски данный алгоритм естественно назвать Linear-time Legendre Transform, а по-русски мы рискнем использовать неологизм *линейно-быстрое преобразование Лежандра*). Рассматривается только одномерный случай, многомерный получается факторизацией, которая проводится аналогично факторизации при выполнении многомерного преобразования Фурье.

Предлагаемое линейно-быстрое преобразование Лежандра позволяет решать одномерное уравнение Бюргерса, используя только $O(N)$ операций. Однако мы приведем и другие алгоритмы для решения уравнения Бюргерса. В работе [7] описана модель движения материальных частиц и их столкновений (слипаний) с сохранением импульса; предлагается метод для полного вычисления всей хронологии столкновений, и тем самым для решения одномерного уравнения Бюргерса "в целом" за $O(N\log N)$ действий. Мы даем другой алгоритм, тоже основанный на использовании модели частиц, который асимптотически работает так же, как и решение уравнения Бюргерса при использовании линейно-быстрого преобразования Лежандра, т.е. за $O(N)$ операций. Этот алгоритм позволяет получить решение для фиксированной временно́й координаты, модификация этого алгоритма предназначена для решения уравнения Бюргерса при фиксированной пространственной координате.

Приведенные алгоритмы, кроме скорости работы, обладают рядом других достоинств: они физичны, промежуточные вычисления обладают физическим смыслом, хорошую точность при работе алгоритмов можно получить и при использовании обычных вещественных чисел, а не только чисел с "двойной точностью", и наконец самое важное их достоинство – они обладают так называемым

свойством открытости, т.е. при их работе не требуется одновременно держать в оперативной памяти компьютера все исходные данные.

АЛГОРИТМ ДЛЯ ВЫЧИСЛЕНИЯ ДИСКРЕТНОГО ОДНОМЕРНОГО ПРЕОБРАЗОВАНИЯ ЛЕЖАНДРА

Дискретное преобразование Лежандра для N точек может быть задано в форме

$$\tilde{U}(i) = \max_{i \le j \le N} (ij - U(i)).$$

Рассмотрим связанную с этим преобразованием матрицу с элементами $a_{ji} = ij - U(j)$:

$j = N$	$1 \cdot N - U(N)$	$2 \cdot N - U(N)$	$3 \cdot N - U(N)$	$\cdots$	$N \cdot N - U(N)$	$\cdots$
$\cdots$	$\cdots$			$i \cdot j - U(j)$	$\cdots$	
$j = 3$	$1 \cdot 3 - U(3)$	$2 \cdot 3 - U(3)$	$3 \cdot 3 - U(3)$	$\cdots$	$N \cdot 3 - U(3)$	
$j = 2$	$1 \cdot 2 - U(2)$	$2 \cdot 2 - U(2)$	$3 \cdot 2 - U(2)$	$\cdots$	$N \cdot 2 - U(2)$	
$j = 1$	$1 \cdot 1 - U(1)$	$2 \cdot 1 - U(1)$	$3 \cdot 1 - U(1)$	$\cdots$	$N\ 1 - U(1)$	
	$i = 1$	$i = 2$	$i = 3$	$\cdots$	$i = N$	$\cdots$

Вычисление преобразования Лежандра сводится к вычислению максимума в каждом из столбцов. Вначале обратим внимание на строки. Очевидно, что для каждой пары строк $j_2 > j_1$ существует позиция (т.е. номер столбца), которую мы назовем *доминантной*, обладающая следующим свойством: начиная с этой доминантной позиции элементы строки j_2 будут больше (или равны), чем элементы строки j_1. Доминантная позиция $D(j_2, j_1)$ находится следующим образом:

$$D(j_2, j_1) = \min\{i | ij_2 - U(j_2) >= ij_1 - U(j_1)\}. \qquad (1)$$

Это наблюдение относительно свойств строк можно развить в алгоритм для нахождения максимумов в столбцах. В самом деле, нетрудно видеть, что нахождение максимумов в столбцах фактически сводится к нахождению последовательности доминантных позиций. Мы не будем приводить формально алгоритм, а только проиллюстрируем его численным примером:

$j = 8$	0.08	8.08	16.08	24.08*	32.08*	40.08	48.08	
$j = 7$	2.40	9.40	16.40	23.40	30.40	37.40	44.40*	
$j = 6$	7.67	13.67	19.67	25.67*	31.67	37.67	43.67	
$j = 5$	9.79*	14.79	19.79	24.79	29.79	34.79	39.79	
$j = 4$	7.02*	11.02	15.02	19.02	23.02	27.02	31.02	
$j = 3$	2.57*	5.57	8.57	11.57	14.57	17.57	20.57	
$j = 2$	0.18*	2.18	4.18	6.18	8.18	10.18	12.18	
$j = 1$	0.15*	1.15	2.15	3.15	4.15	5.15	6.15	
	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$	$\cdots$

Все доминантные позиции, которые последовательно находятся в процессе работы алгоритма, отмечены значком $*$. Обратим внимание, что после нахождения $D(8,7)$ ряд $j = 7$ исключается из ответа и вычисляется доминантная позиция $D(8,6)$. В конце алгоритма мы получаем три доминантные позиции: $D(j_5, j_4), D(j_6, j_5), D(j_8, j_6)$; части строк номер 5, 6 и 8, расположенные между доминантными позициями (они выделены), содержат искомые максимумы столбцов.

Хотя это и не очевидно на первый взгляд, количество операций, выполняемых при работе нашего алгоритма, линейно зависит от количества строк. В самом деле, каждая строка один раз включается в рассмотрение и не больше одного раза исключается из рассмотрения. Каждая из этих операций сводится к вычислению по формуле (1).

Мы реализовали этот алгоритм на языке C и тестировали его на рабочей станции с производительностью около 5 MFlops. Для исходного множества из $N = 2^{20}$ точек (около 1 млн) вычисления занимают менее 10 с.

ПОСТАНОВКА ЗАДАЧИ О ВЫЧИСЛЕНИИ ТРАЕКТОРИЙ

Как было отмечено выше, линейно-быстрое преобразование Лежандра может быть применено к решению уравнения Бюргерса. Однако более физичный алгоритм может быть получен при рассмотрении траекторий материальных точек.

Сама задача может быть поставлена в элементарной форме. В момент времени T_0 на отрезке имеется система из N частиц с различными массами и скоростями. При столкновении двух (или более) частиц вместо исходных участников столкновения образуется одна новая в соответствии с некоторым законом. Требуется определить положение частиц и их скорости при фиксированной временно́й координате. Такая постановка задачи актуальна, например, для космологии. Для других приложений более интересна другая задача: фиксировать пространственную координату, т.е. выбрать точку X и проследить частицы, которые будут проходить через эту точку.

Для приведенных ниже алгоритмов сам закон, по которому происходит столкновение частиц, не существен. Фактически в алгоритмах используется только то свойство, что плоскость пространство–время траектории частиц разбивают на выпуклые фигуры. Как указано в работах [7, 8], для решения одномерного уравнения Бюргерса следует использовать следующий закон столкновения двух частиц: при столкновении образуется новая частица, имеющая массу, равную сумме масс участников, и импульс, равный сумме импульсов участников.

АЛГОРИТМ ДЛЯ РЕШЕНИЯ УРАВНЕНИЯ БЮРГЕРСА ПРИ ФИКСИРОВАННОЙ ВРЕМЕННО́Й КООРДИНАТЕ

В работе [7] используется вся совокупность исходных точек. Их траектории очевидны до момента первого столкновения какой-то пары точек. Задача ставится так – найти время ближайшего столкновения и его участников. Для определения этой последовательности столкновений фактически требуется сортировка, на что и тратится $\log_2 N$ операций. Кроме того, все траектории желательно держать в оперативной памяти для обеспечения быстрого поиска.

Естественный способ ускорить этот алгоритм состоит в том, чтобы попытаться провести как можно больше вычислений в окрестности каждой точки, не привлекая к рассмотрению далеко расположенные точки.

Введем формальные операции над траекториями, которые соответствуют слипанию материальных точек. Определим траектории следующим образом:

$$\mathrm{tra}j ::= (x, v, m), \tag{2}$$

где x – позиция в начальный момент времени T_0, v – скорость, m – масса.

Очевидно, что в процессе слипания будут образовываться частицы, которые физически не существовали в начальный момент времени T_0, так что определение компоненты x в формуле (2) требует небольших формальных уточнений. Тем не менее и такие траектории мы будем задавать в виде (2); в этом случае компоненту x можно получить, рассмотрев пересечение продолжения физически существующего отрезка траектории с осью абсцисс.

Результат столкновения двух траекторий, очевидно, задается следующей формулой:

$$(x_1, v_1, m_1) * (x_2, v_2, m_2) = \left(\frac{m_1 x_1 + m_2 x_2}{m_1 + m_2}, \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}, m_1 + m_2 \right).$$

Отметим, что выражение для первой компоненты задает положение центра масс образованной вновь частицы (если бы она существовала в начальный момент времени); формула для второй компоненты следует из закона сохранения момента.

Операция столкновения является ассоциативной операцией:

$$(\mathrm{tra}j_1 * \mathrm{tra}j_2) * \mathrm{tra}j_3 = \mathrm{tra}j_1 * (\mathrm{tra}j_2 * \mathrm{tra}j_3).$$

Это свойство дает возможность вычислить результат столкновения некоторого числа соседних частиц без знания точного порядка их столкновений.

Перейдем к изложению линейно-быстрого алгоритма для решения уравнения Бюргерса для фиксированной временно́й координаты T, который основан на ассоциативности операции столкновения траекторий. В начальный момент времени мы имеем множество траекторий

$$\{p_1, p_2, \ldots, p_N\}.$$

Будем продвигаться справа налево и заменять соседние траектории их пересечением тогда и только тогда, когда это пересечение происходит в момент времени до фиксированного порога T. На каждом шаге алгоритма будем пытаться сформировывать новую траекторию с как можно большей массой и поддерживать общее количество рассматриваемых траекторий как можно меньшим. Получаемые таким образом траектории мы будем называть *кластерами*. В значительной степени слова: траектория, частица, кластер, являются синонимами; термин *кластер* мы будем употреблять в тех ситуациях, когда хотим подчеркнуть способ его образования – то, что он не пересекается с соседями.

Алгоритм работает в несколько шагов; общее количество этих шагов будет зависеть от начального распределения частиц (от 1 до N). Каждый шаг делится на две части.

Первая часть шага. После выполнения предыдущего шага алгоритма мы создали множество из $k-1$ траекторий $\{P_1, P_2, \ldots, P_{k-1}\}$; $\{p_i, p_{i+1}, \ldots, p_N\}$ – остаток тех траекторий из начального множества траекторий, которые еще не рассмотрены.

Получаем следующую схему:

Мы находимся у p_{i+1}, т.е. только что сформировали еще одну траекторию P_k

$$\{P_1, P_2, \ldots, P_{k-1}, P_k, \quad p_i, p_{i+1}, \ldots, p_N\}.$$

Мы только что обнаружили, что траектория P_k, которую мы сейчас формируем, не пересекается до момента времени T с очередной траекторией p_i из начального множества.

Вторая часть шага. Во время второй части шага будем совершать действия в соответствии со следующей схемой:

$$\{P_1, P_2, \ldots, P_{k-1} * P_k, \quad p_i, p_{i+1}, \ldots, p_N\}$$

Продвигаемся влево как можно дальше, сокращая количество траекторий.

Сколько операций тратит наш алгоритм? На первом шаге мы добавляем траектории из начального множества к уже существующему кластеру траекторий из начального множества траекторий; таким образом, общее количество операций составит N.

На втором шаге исключаем некоторые кластеры; каждый кластер может быть представлен его последней (или первой) компонентой, так что общее количество операций не превышает N. Итак, общее количество операций равно N в лучшем случае и $2N$ в худшем случае.

АЛГОРИТМ ДЛЯ РЕШЕНИЯ УРАВНЕНИЯ БЮРГЕРСА ПРИ ФИКСИРОВАННОЙ ПРОСТРАНСТВЕННОЙ КООРДИНАТЕ

Как указано в работе [7], траектории частиц разбивают полуплоскость $X * T$ на выпуклые многоугольники. Это обстоятельство указывает на некоторую возможную симметрию алгоритмов решения уравнения Бюргерса при фиксированной временно́й координате и при фиксированной пространственной координате.

Действительно, для порога по пространственной координате, который находится правее (или левее) всех исходных позиций траекторий, алгоритм содержательно не меняется. Меняется только условие формирования кластеров из частиц. А именно, условие пересечения в момент времени до фиксированного порога T заменяется условием: пересечение происходит в точке, находящейся левее (правее) порога.

В случае произвольного порога T алгоритм обрастает некоторыми подробностями. Ограничимся только общими указаниями о работе алгоритма. Вначале мы

рассматриваем отдельно множество частиц, находящихся левее порога, и множество частиц, находящихся правее порога. Для каждого из этих множеств в отдельности формируем кластеры из частиц, которые будем называть левым и правым множествами. Рассматривая соседние с порогом кластеры из левого и из правого множеств, определяем, какая из этих двух частиц первой пересекает порог. Если, например, первой пересекает частица из левого множества, то:

исключаем ее из левого множества;

объединяем ее с самой левой частицей из правого множества;

пересчитываем кластеры из правого множества.

Мы реализовали этот алгоритм на языке С и тестировали его на IBM-совместимом персональном компьютере. При дискретизации, использующей $N = 2^{17}$ точек (приблизительно 100 тыс.), вычисления производятся в оперативной памяти компьютера (640 кБ) и занимают менее 1 мин.

Благодарности. Работа была выполнена в Обсерватории г.Ниццы и поддержана Министерством высшего образования Франции и фондом ИНТАС (грант INTAS-93-457), а также Российским фондом фундаментальных исследований (грант 94-05-16444). Автор приносит благодарность А.Я. Гордону и сотрудникам Обсерватории г.Ниццы И. Вирту, А. Нуллезу, У. Фришу, М. Хенону за полезные обсуждения.

ЛИТЕРАТУРА

1. *Lax P.D.* Hyperbolic systems of conservation laws and the mathematical theory of shocks waves. S.I.A.M., 1973. 150 p.
2. *Noullez A., Vergassola M.* A fast legendre transform algorithm and applications to the adhesion model // J. Sci. Comp. 1994. Vol.9. P.259-281.
3. *She Z.-S., Aurell E., Frisch U.* The inviscid Burgers' equation with initial conditions of Brownian type // Comm. Math. Phys. 1992. Vol.148. P.623-641.
4. *Vergassola M., Dubrulle B., Frisch U., Noullez A.* Burgers' equation, Devil's staircases and the mass distribution for large-scale structures // Astron. and Astrophys. 1994. Vol.289. P.325-356.
5. *Weinberg D.H., Gunn J.E.* Large-scale structure and the adhesion approximation // Mon. Not. Roy. Astron. Soc. 1990. Vol.247. P.260-286.
6. *Brenier Y.* Un algorithme rapide pour le calcul de transformes de Legendre-Fenchel descrete // C. r. Acad. Sci. Serie I. 1989. Vol.308. P.587-589.
7. *Гордон А.Я.* Быстрый алгоритм пространственно-временного решения уравнения Бюргерса с исчезающей вязкостью // Наст. сб.
8. *Гурбатов С.Н., Малахов А.Н., Саичев А.И.* Нелинейные случайные волны в средах без дисперсии // Современные проблемы физики. Вып.91. М.: Наука, 1990. 215 с.