УДК 550.388, 520.16

НОВЫЙ АЛГОРИТМ СВЕРТКИ ДЛЯ ОБРАБОТКИ ДАННЫХ РАДАРА НЕКОГЕРЕНТНОГО РАССЕЯНИЯ

С.С. Алсаткин, А.Л. Воронов

NEW CONVOLUTION ALGORITHM FOR PROCESSING INCOHERENT SCATTER RADAR DATA

S.S. Alsatkin, A.L. Voronov

В данной работе представлена нерекурсивная программная реализация алгоритма Карацубы. Необходимость в разработке подобной реализации данного алгоритма возникла в связи с тем, что его прямая рекурсивная реализация приводит к неконтролируемому расширению памяти и, вследствие этого, к резкому увеличению времени исполнения операций. Это не соответствует теоретическим выкладкам, которые учитывают лишь общее число операций сложения и умножения. Нерекурсивная реализация позволяет увеличить производительность по сравнению с рекурсивной реализацией за счет контроля распределения динамической памяти. Алгоритм позволяет значительно ускорить обработку больших объемов данных, получаемых на Иркутском радаре некогерентного рассеяния.

The nonrecursive program realization of Karatsuba algorithm is presented. A necessity of working-out this algorithm is that its direct recursive realization result in uncontrolled storage extension, and as a consequence, the operation time increases sharply. This does not correspond to theoretical computations allowing for only total number of addition and subtraction operations. The nonrecursive realization makes it possible to raise efficiency as compared with the recursive realization due to controlling the dynamic storage distribution. The algorithm allows us to quicken the processing of large data level from Irkutsk incoherent scatter radar.

Введение

Свертка применяется в большинстве задач обработки данных, в том числе получаемых на Иркутском радаре некогерентного рассеяния (ИРНР). Для вычисления свертки обычно применяют алгоритм на основе быстрого преобразования Фурье (БПФ), который превосходит по скорости вычисления стандартный алгоритм. Но, если необходимо вычислить большое количество сверток, нужно иметь алгоритм, более эффективный по скорости по сравнению с БПФ. В качестве такой альтернативы в данной статье рассматривается алгоритм Карацубы [Дональд Кнут, 2007; Карацуба, Офман, 1962].

Алгоритм Карацубы известен давно, но мало распространен из-за своей сложности и вычислительных затрат при увеличении длины входных последовательностей по сравнению с БПФ. Данный алгоритм превосходит БПФ по производительности при определенных длинах входных последовательностей, но для остальных значений, как правило, БПФ является наиболее быстрым алгоритмом вычисления свертки.

Преобразование, на котором основан данный алгоритм имеет вид:

$$p(x)=p_0+p_1x;\ q(x)=q_0+q_1x.$$

Найдем произведение данных выражений:

$$p(x)q(x)=(p_0+p_1x)(q_0+q_1x) \text{ или}$$
$$p(x)q(x)=p_0q_0+p_0q_1x+p_1q_0x+p_1q_1x^2.$$

В полученном выражении имеем четыре умножения и одно суммирование. Для сокращения числа умножений два перекрестных произведения в последнем выражении представим в следующем виде:

$$p_0q_1x+p_1q_0x=$$
$$=p_1q_1x+p_0q_0x-(q_0-q_1)(p_0-p_1)x.$$

Тогда окончательное выражение для произведения примет вид:

$$p(x)q(x)=p_0q_0+$$
$$+\left(p_1q_1+p_0q_0-(q_0-q_1)(p_0-p_1)\right)x+p_1q_1x^2.$$

Таким образом, вместо четырех произведений и одного суммирования, получим три произведения и четыре суммирования. Для свертки между полиномами вида $D(x)=\sum_{i=0}^{N-1}d_ix^i$ и $G(x)=\sum_{i=0}^{N-1}g_ix^i$ алгоритм Карацубы становится рекурсивным.

Из самой сути преобразования Карацубы следует, что для эффективного вычисления произведения двух полиномов должны выполнятся следующие требования.

1. Размерности обоих полиномов должны быть одинаковыми, т. е. число элементов в первом и втором полиномах одинаково.

2. Число элементов должно быть четным и бдолжно ыть степенью 2, так как каждый из полиномов делится на две равные части.

Мультипликативная сложность алгоритма определяется следующим выражением:

$$mul=N^{1.585}.$$

Из приведенной сравнительной таблицы видно, что алгоритм Карацубы при длинах входных последовательностей до 2000 по производительности превосходит алгоритм на основе БПФ. В задачах обработки данных на ИРНР длины входных последовательностей не превосходят 2000 элементов, что делает использование данного алгоритма актуальным.

В таблице приведены сравнительные характеристики производительности алгоритма Карацубы и алгоритма на основе БПФ.

N	$n(3\log_2 n+1)$	$n^{1.585}$
64	2816	729
128	6400	2187
256	14336	6562
512	31744	19687

1024	69632	59064
2048	151552	177197
4096	327680	531606
8192	704512	1594861

Существенный недостаток существующей реализации преобразования Карацубы, как и у алгоритма БПФ, заключается в том, что число элементов в полиномах должно быть равно степени 2. Если нужно найти свертку двух полиномов степени m, где m не является степенью 2, нужно предварительно расширить их до ближайшего числа N, соответствующего степени 2. Программная реализация алгоритма Карацубы является рекурсивной, что приводит к увеличению времени вычисления свертки.

Далее рассмотрим модифицированную нерекурсивную реализацию алгоритма, не требующего расширения массивов до числа элементов, кратных степени 2, и по производительности превосходящего рекурсивную реализацию.

Модифицированный алгоритм Карацубы

Устранение необходимости расширения числа элементов в массиве до ближайшей степени 2

Рассмотрим два полинома

$$D(x)=\sum_{i=0}^{N-1} d_i x^i \text{ и } G(x)=\sum_{i=0}^{N-1} g_i x^i$$

одинаковой размерности с числом элементов $N>0$. Запишем выражение для свертки между данными полиномами

$$S(x)=\left(\sum_{i=0}^{N-1} d_i x^i\right)\left(\sum_{i=0}^{N-1} g_i x^i\right).$$

Следуя принципу преобразования Карацубы, разобьем каждый полином на два полинома:

$$\begin{cases} D(x)=\sum_{i=0}^{N-1} d_i x^i=\sum_{i=0}^{n_1-1} d_i x^i+\sum_{i=n_1}^{N-1} d_i x^i \\ G(x)=\sum_{i=0}^{N-1} g_i x^i=\sum_{i=0}^{n_1-1} g_i x^i+\sum_{i=n_1}^{N-1} g_i x^i. \end{cases}$$

Сделаем во второй сумме каждого полинома преобразования вида $i\to i-n_1$:

$$\begin{cases} D(x)=\sum_{i=0}^{N-1} d_i x^i=\sum_{i=0}^{n_1-1} d_i x^i+x^{n_1}\sum_{i=0}^{n_2} d_i x^i \\ G(x)=\sum_{i=0}^{N-1} g_i x^i=\sum_{i=0}^{n_1-1} g_i x^i+x^{n_1}\sum_{i=0}^{n_2} g_i x^i. \end{cases}$$

Тогда свертка примет вид:

$$S(x)=S_0(x)+x^n[-S_0(x)+S_1(x)+S_2(x)]+x^{2n}S_1(x), \qquad (1)$$

где введены следующие обозначения:

$$\begin{cases} S_0(x)=\left(\sum_{l=0}^{n_1-1} p_l x^l\right)\left(\sum_{l=0}^{n_1-1} q_l x^l\right) \\ S_1(x)=\left(\sum_{l=0}^{n_2-1} p_{l+n_1} x^l\right)\left(\sum_{l=0}^{n_2-1} q_{l+n_1} x^l\right) \\ S_2(x)=\left(\sum_{l=0}^{n_1-1}\left(p_l+p_{l+n_1}\right)x^l\right)\left(\sum_{l=0}^{n_1-1}\left(q_l+q_{l+n_1}\right)x^l\right). \end{cases}$$

Блок-схема нерекурсивной реализации метода Карацубы.

Как видно из последнего выражения, $n_1+n_2=N$, при этом n_1 и n_2 могут быть как равными между собой, если N четное, так и отличаться на 1, т. е. $n_1=n_2+1$, если N нечетное. Таким образом, в программной реализации алгоритма достаточно отслеживать, является ли число N четным или нечетным, чтобы правильно разбить полиномы и вычислить получившиеся свертки.

Блок-схема нерекурсивной реализации алгоритма представлена на рисунке.

Основные этапы, осуществляемые в алгоритме Карацубы: разбиение входных массивов, вычисление непосредственно свертки и сборка.

Программная реализация данного алгоритма включает два дополнительных этапа.

1. Корректировка работы алгоритма для проверки необходимости осуществления сборки, т. е. определения того, вычислены ли все свертки в выражении (1): $S_0(x)$, $S_1(x)$, $S_2(x)$.

2. Корректировка адресов и числа элементов в полиномах. Выбор свертки $S_0(x)$ или $S_1(x)$ либо вычисление $S_2(x)$. Основной цикл выполняется до тех пор, пока не будет вычислена основная свертка $S(x)$.

Заключение

Модифицированный алгоритм Карацубы показал выигрыш более чем в два раза по сравнению с его стандартной программной реализацией.

СПИСОК ЛИТЕРАТУРЫ

Кнут Д. Искусство программирования. Т. 2. Получисленные алгоритмы. 3-е изд. М.: Вильямс, 2007. 832 с.

Карацуба А.А., Офман Ю. Умножение многозначных чисел на автоматах // Доклады Академии Наук СССР. 1962. Т. 145. № 2.

Институт солнечно-земной физики СО РАН, Иркутск, Россия