Вычисление значений многомерных полиномов

1. Вступление

Данная статья посвящена вычислению значений многомерных полиномов. Она предлагает подход, основанный на одномерной схеме Горнера, которая позволяет быстро и с заданной точностью вычислять указанные значения. Вычисления производятся с помощью сервисов, предоставляемых системой MathCloud.

Вычисление значения функции играет одну из важнейших ролей в высокопроизводительных вычислениях. Одними из самых важных классов функций, с которыми работает прикладная математика, являются полиномы. Ими описываются многочисленные математические объекты, такие, как правые части обыкновенных дифференциальных уравнений, ограничения задач нелинейного программирования, задачи математической физики и другие.

Хорошо известно, что вычисление значений одномерных полиномов осуществляется с помощью схемы Горнера. В данной статье обсуждается проблема обобщения данной схемы на случай многомерных полиномов. Особое внимание уделяется разработке схемы, которая может быть успешно декомпозирована и распараллелена. Таким образом, вычисление значений многомерных полиномов становится доступным для экстренных вычислений.

Схема Горнера для одномерных полиномов изучается во многих работах, например, в [1]. Попытки обобщить эту схему для многомерного случая осуществлялись в [2]. В данный момент существуют несколько многомерных схем. Одна из них рассмотрена в работе [3]. Другая схема, представленная и изученная в статье [4], была разработана для одного специального класса полиномов. Данная работа обобщает указанные схемы, а также предлагает новые алгоритмы, также оказавшиеся успешными.

2. Обобщение схемы Горнера на многомерный случай

Рассмотрим следующее выражение:

$$P[x_1,...,x_n] = \sum_{\alpha} a_\alpha x^\alpha ,$$

где $\alpha = (\alpha_1,...,\alpha_n)$ – n-мерный вектор, а $x^\alpha = x_1^\alpha \cdot ... \cdot x_n^\alpha$. Назовем $P[x_1,...,x_n]$ многомерным полиномом.

Рассмотрим некоторые методы, позволяющие вычислять значение такого полинома в некоторой точке $x_0 = (x_1^0,...,x_n^0)$.

2.1 Примитивный метод

Примитивный метод вычисляет значение полинома путем последовательного вычисления мономов $a_\alpha\ x^\alpha$ для всех α. Суммируя полученные результаты, получаем исходное значение полинома. Существенный недостаток этого метода – необходимость каждый раз заново вычислять повторяющиеся степени α_i. Данный недостаток устраняется следующим методом.

2.2 Улучшенный примитивный метод (метод таблиц)

Данный метод тоже последовательно вычисляет все мономы $a_\alpha x^\alpha$, но избавляет нас от необходимости заново вычислять повторяющиеся степени. Вычисляя некоторую степень α_i, мы ‘запоминаем’ это значение, чтобы использовать его в последующих вычислениях. Фактически мы заполняем следующую таблицу:

Таб 1. Степени переменных

	2	3	…	…	…	*K*	…	…	…
x_1						…			
x_2						…			
…						…			
x_i	x_i^2	x_i^3	…	…	…	x_i^k	…	…	…
…						…			
x_n						…			

Встречая впервые в процессе вычислений некоторую степень k переменной x_i, мы вычисляем соответствующую степень, внося в процессе вычисления в таблицу значения этой переменной для степеней 2, 3,…, k-1, k. Если в дальнейшем мы встречаем любую из степеней 2,…, k данной переменной x_i, мы не вычисляем заново это значение (как в примитивном методе), а берем его из таблицы. Если мы встречаем степень $p>k$ переменной x_i, то не вычисляем степень p заново, а используем уже вычисленное ранее значение. Данная таблица существенно упрощает вычисления.

Дальнейшие методы являются, по сути, обобщением схемы Горнера для одномерного случая (многочлена одной переменной). Напомним эту схему. Рассмотрим одномерный полином

$$P[x] = \sum_{i=0}^{n} a_i x^i \ .$$

Вычисляя его значение в точке x_0 с помощью схемы Горнера, мы записываем его в следующем виде:

$$P[x] = (...((a_n x + a_{n-1})x + a_{n-2})x + ...)x + a_0 .$$

Приведем теперь несколько возможных способов факторизации, т.е. определенной группировки слагаемых в целях упрощения вычислений. Отметим следующий момент: в одномерной схеме Горнера последовательно группируются пары слагаемых, в которых за скобку выносится слагаемое наибольшей степени. Обобщив эту идею на многомерный случай, получим следующую схему.

2.3 Схема Горнера 1

Для каждой пары мономов (i, j) многомерного полинома $P[x_1, ..., x_n]$ определим их сумму

$$\xi_{ij} = a_{\alpha_i} \cdot x^{\alpha_i} + a_{\alpha_j} \cdot x^{\alpha_j} = x^{\alpha_{ij}} \cdot (a_{\alpha_i} \cdot x^{\alpha_i - \alpha_{ij}} + a_{\alpha_j} \cdot x^{\alpha_j - \alpha_{ij}}),$$

где

$$\alpha_{ij} = (\min(\alpha_i^1, \alpha_j^1), ..., \min(\alpha_i^n, \alpha_j^n))$$

В результате получим слагаемое

$$\xi_{ij} = a_{\alpha_{ij}} \cdot x^{\alpha_{ij}},$$

где

$$a_{\alpha_{ij}} = a_{\alpha_i} \cdot x^{\alpha_i - \alpha_{ij}} + a_{\alpha_j} \cdot x^{\alpha_j - \alpha_{ij}},$$

т.е. слагаемое того же вида, что и в исходном многочлене, где роль коэффициента играет указанное выше выражение. Для всех i и j вычислим их сумму. Из этих слагаемых, а их будет

$$M - 1 + M - 2 + ... + = \frac{M \cdot (M-1)}{2},$$

выберем то, для которого следующая сумма минимальна:

$$\xi^{\max} = \max_{i,j} \left\{ \sum_{k=1}^{n} \min(\lambda_k^i, \lambda_k^j) \right\}$$

Тогда исходный многочлен пример вид

$$P[x_1, ...x_n] = \xi^{\max} + \sum_{\alpha \neq \alpha_i, \alpha_j} a_\alpha x^\alpha \cdot$$

Таким образом, число слагаемых исходного многочлена уменьшилось на единицу. Продолжаю данную факторизацию, мы в конечном итоге сведем исходный полином либо к единственному члену (тогда мы вычислим его и получим результат), либо к полиному, где никакие два члена не имеют общих слагаемых, которые можно вынести за скобку. В последнем случае мы вычисляем значение получившегося полинома, например, табличным методом.

Нетрудно убедиться, что в одномерном случае такая факторизация дает нам обычную одномерную схему Горнера.

2.4 Схема Горнера 2

Рассмотрим исходный полином $P[x_1, ...x_n]$ как полином относительно одной переменной, например x_1 (т.е. зафиксируем все остальные переменные). Тогда, собирая подобные слагаемые при одинаковых

степенях, перепишем полином в следующем виде:

$$P[x_1,...x_n] = \sum_{\alpha} a_\alpha x^\alpha = \sum_{k=0}^{N_1} A_k^1(x_2,...,x_n)x_1^k .$$

Коэффициенты $A_k^1(x_2,...,x_n)$ зависят от остальных переменных $x_2,...,x_n$, а N_1 – максимальная степень переменной x_1 в исходном полиноме. Если какой-то степени x_1^k в полиноме не было, соответствующий коэффициент $A_k^1(x_2,...,x_n)$ положим равным 0. Теперь применим к $P[x_1,...,x_n]$ одномерную схему Горнера:

$$P[x] = (...((A_{N_1}^1 x + A_{N_1-1}^1)x + A_{N_1-2}^1)x + ...)x + A_0^1,$$

где $A_k^1 = A_k^1(x_2,...,x_n)$. Снова зафиксирует переменную, например, x_2 s:

$$A_k^1(x_2,...,x_n) = \sum_{l=0}^{N_2(k)} A_l^2(x_3,...,x_n)x_2^l,$$

где $N_2(k)$ - максимальная степень переменной x_2 в коэффициенте $A_k^1(x_2,...,x_n)$. К $A_k^1(x_2,...,x_n)$ снова применим одномерную схему Горнера, и т.д. Продолжая данный алгоритм рекурсивно, на конечном шаге i_0 получим, что $A_m^{i_0} = A_m^{i_0}(x_n)$. Применяя к этим коэффициентам одномерную схему Горнера и последовательно возвращаясь к первому шагу, мы вычислим искомое значение полинома $P[x_1,...,x_n]$.

В этой схеме важно отметить следующее. На произвольном шаге i коэффициенты $A_l^i(x_{i+1},...,x_n)$ можно считать независимо друг от друга. Поэтому, в отличие от предыдущего способа факторизации, где каждый последующий шаг зависел от предыдущего, этот алгоритм отлично распараллеливается.

2.5 Схема для специального класса полиномов

По сути, эта схема основана на той же идее фиксирования всех переменных, кроме одной, но представляет собой последовательную (а не рекурсивную) реализацию. Для большей ясности и простоты мы опишем этот алгоритм для некоторого особого класса полиномов (т.н. ‘полных полиномов’). Эта схема интересна тем, что каждой итерации ее цикла выполняется только одна операция умножения! Прежде чем описать алгоритм, введем некоторые определения.

Рассмотрим полином

$$P[x_1,...,x_n] = \sum_{\alpha \in I} a_\alpha x^\alpha ,$$

где $I \subset \mathrm{N}_0^n$ множество мультииндексов $\alpha = (\alpha_1,...,\alpha_n)$ данного полинома. Назовем множество I *нижним множеством* при условии, что, если некоторый мультииндекс $\beta \in I$, то все мультииндексы $\alpha \le \beta$ также принадлежат I. Напомним, что $\alpha \le \beta \Leftrightarrow \alpha_i \le \beta_i \ \forall i = 1...n$. Определим также понятие

обратного лексикографического порядка на множестве мультииндексов I, а именно $\alpha \prec \beta \Leftrightarrow \exists k \in \{1,...,n\}, \alpha_k < \beta_k, \alpha_{k+1} = \beta_{k+1},..., \alpha_n = \beta_n$.

Используя данный порядок, упорядочим мультииндексы α исходного полинома в порядке убывания: $\alpha(0),...,\alpha(M)$, где $M = |I| - 1$. Качественно такое упорядочивание означает следующее. Введем в рассмотрение множества I_k (которые также будут использоваться в дальнейшем):

$$I_k := \left\{\alpha \in \mathrm{N}_0^{n-1} : (\alpha, k) \in I\right\}, \quad k = 0...k_I,$$

где $k_I = \max\{\alpha_n : \alpha \in I\}$, т.е. k_I – наибольшая степень переменной x_n в исходном полиноме. Очевидно, что $I = I_0 \cup ... \cup I_k$. Заметим также, что

$$\alpha \prec \beta \; \forall \alpha \in I_i, \beta \in I_j, \quad i < j, \; i, j = 0...k_I$$

Иными словами, множество I разбивается на подмножества I_k путем упорядочивания слагаемых исходного полинома по убыванию степени последней переменной x_n. Проделывая то же самое рекурсивно с множествами I_k , но проводя их упорядочивание по предпоследней переменной x_{n-1}, мы получим обратный лексикографический порядок на множестве I.

Опишем теперь алгоритм вычисления значения полинома $P[x]$ в некоторой точке $x = (x_1,..., x_n)$.

Переменные

1. n – количество переменных в полиноме
2. $r_0,..., r_n$ - счетчики.
3. $i_1,..., i_n$ - индексы.

Инициализация

1. $r_0 = a_{\alpha(0)}, r_1 = ... = r_n = 0$.
2. $i_k = \alpha(0)_k, k = 1...n$.

Тело цикла

1. For $l = 1...M$ do: $k = \max\{1 \le t \le n : \alpha(l)_t \ne \alpha(l-1)_t\}$. k находится из определения обратного лексикографического порядка.
2. $i_k = i_k - 1$.
3. $r_k = x_k(r_0 + ... + r_k)$.
4. $r_0 = a_{\alpha(l)}, \; r_1 = ... = r_{k-1} = 0$.
5. $i_1 = \alpha(l)_1,..., i_{k-1} = \alpha(l)_{k-1}$.

Докажем следующую теорему.

Теорема. $P[x] = r_0 + r_1 + ... + r_n$

Доказательство. Доказательство проведет индукцией по числу переменных n. Пусть $n = 1$. В силу

того, что множество I нижнее, оно имеет вид:

$$I = ((0,0,...,0),(1,0,...0),...,(k_I,0,...0))$$

Значение k_I уже было введено выше. Наш полином принимает вид:

$$P[x_1] = \sum_{j=0}^{k_I} a_{(j,0,...0)} x_1^j$$

Если $n = 1$, то у нас есть 2 счетчика: r_0 и r_1, где

$$r_0 = a_{(k_I,0,...,0)},\ r_1 = 0$$

После первой итерации алгоритма они принимают значения:

$$r_0 = a_{(k_I-1,0,...,0)},\quad r_1 = x_1 a_{(k_I,0,...,0)},$$

а после k итераций:

$$r_0 = a_{(k_I-k,0,...,0)},$$

$$r_1 = (...(x_1 a_{(k_I,0,...,0)} + a_{(k_I-1,0,...,0)})x_1 + ... + a_{(k_I-k+1,0,...,0)})x_1 = \sum_{j=k_I-k+1}^{k_I} a_{(j,0,...,0)} x_1^j$$

Полагая $k = k_I$, получаем после всех итераций:

$$r_0 + r_1 = a_{(0,0,...,0)} + \sum_{j=1}^{k_I} a_{(j,0,...,0)} x_1^j = \sum_{j=0}^{k_I} a_{(j,0,...,0)} x_1^j = P[x_1]$$

Итак, основание индукции доказано. Пусть теперь $n>1$. Чтобы проделать шаг индукции, запишем наш полином в немного другом виде и вспомним определение множеств I_k:

$$P[x_1,...,x_n] = \sum_{\alpha \in I} a_\alpha x^\alpha = \sum_{k=0}^{k_I} q_k x_n^k,$$

где

$$q_k = \sum_{\beta \in I_k} a_{(\beta,k)} \prod_{j=1}^{n-1} x_j^{\beta_j}$$

Обратимся теперь к множествам I_k. Заметим, что $I = I_0 \cup ... \cup I_k$ и $\alpha \prec \beta\ \forall\ \alpha \in I_i$, $\beta \in I_j$, где $i<j$, $i,j = 0...k_I$. Следовательно, существуют такие числа

$$-1 = M_{k_I+1} < M_{k_I} < ... < M_1 < M_0 = M,$$

что

$$I_k = \{\alpha(M_{k+1}+1),...,\alpha(M_k)\}, \quad k = 0,...,k_I,$$

то есть эти числа M_k обозначают границы множеств I_k. Теперь мы выполняем первые M_k итераций нашего алгоритма. По предположению индукции:

$$r_0 + ... + r_n = \sum_{\alpha \in I_{k_I}} a_\alpha \prod_{j=1}^{n-1} x_j^{\alpha_j} = q_{k_I},$$

учитывая определение q_k. Заметим, что

$$\alpha(M_{k_I}+1)_n < \alpha(M_{k_I})_n,$$

поэтому следующая итерация алгоритма дает нам следующие значения переменных-счетчиков:

$$r_0 = a_{\alpha(M_{k_I}+1)}, \ r_1 = ... = r_{n-1} = 0, \ r_n = x_n q_{k_I}$$

В итоге мы получим следующие значения счетчиков:

$$r_0 = a_{\alpha(M_{k_I-k}+1)}, \ r_1 = ... = r_{n-1} = 0, \ r_n = \sum_{j=k_I-k}^{k_I} q_j x_n^j$$

При $k = k_I$ - 1:

$$r_n = \sum_{j=1}^{k_I} q_j x_n^j$$

Наконец, сделаем последние итерации алгоритма. Счетчик r_n при этом не изменит своего значения, а сумма остальных $r_0, ..., r_{n-1}$ даст q_0. В итоге, после всех итераций, мы получим:

$$r_0 + ... + r_n = q_0 + \sum_{j=1}^{n} q_j x_n^j = \sum_{j=0}^{n} q_j x_n^j = P[x_1, ..., x_n],$$

что и доказывает корректность нашего алгоритма. Здесь важно отметить, что наличие в цикле алгоритма всего одной операции умножения является следствием того факта, что I - нижнее множество.

3. Вычислительные аспекты

3.1 Уровни декомпозиции алгоритмов

1. Декомпозиция изначальной задачи на вычислительные узлы и установление взаимодействия между ними. Это обеспечивает система MathCloud[5].

2. Возможность распараллелить исходный алгоритм на вычислительных узлах. С этой позиции из всех

описанных методов Схема Горнера 2 является оптимальной.

3.2 Система MathCloud

Целью данной системы является предоставление унифицированного доступа к проблемно-ориентированным вычислительным сервисам и поддержка интеграции этих сервисов в решение прикладных задач. В основе системы MathCloud лежат следующие принципы: удобство разработки сервисов, простой доступ к этим сервисам и применение открытых технологий. Для этих целей активно используются современные веб-технологии. Сервис-ориентированный подход позволяет пользователям MathCloud абстрагироваться от необходимым им конкретных ресурсов и формулировать запрос к системе в терминах предметной области. Это существенно облегчает задачу неподготовленному пользователю. Данный подход также идеально подходит для интеграции программных ресурсов, таких как математические и вычислительные пакеты. В случае, когда для выполнения запроса сервису требуются вычислительные ресурсы, данный запрос может преобразовываться в вычислительные задания, запускаемые на кластере или в грид. Тем самым, MathCloud также опирается на существующие вычислительные ресурсы и инфраструктуры. Однако детали реализации вычислений внутри сервиса скрыты от его пользователей[6].

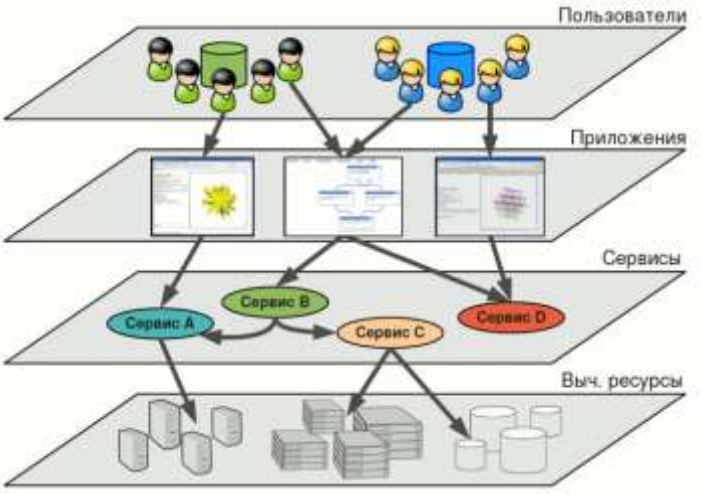

Рис 1. Архитектура MathCloud

Нижний уровень архитектуры MathCloud содержит вычислительные ресурсы, используемые для функционирования сервисов среды. Формально данный уровень не относится к самой среде MathCloud, а формируется из доступных разработчикам и пользователям ресурсов – суперкомпьютеров, кластеров, сегментов грид-сетей с установленным необходимым оборудованием.

На уровне сервисов реализуется удаленный программный доступ к некоторой востребованной пользователям MathCloud функциональности. Прикладные сервисы, являющиеся главными компонентами среды, ориентированы на решение определенного класса математических задач и служат основой для создания. Большинство математических ресурсов могут быть представлены в виде одной или нескольких функций, каждая из которых имеет свой набор входных и выходных параметров.

На уровне приложений реализуется доступ пользователей к сервисам MathCloud через проблемно-ориентированные интерфейсы.

Копии нашей программы были размещены на следующих ресурсах: суперкомпьютеры МГУ (‘Чебышев’ и ‘Ломоносов’), ЮУГУ, МСЦ РАН, кластеры ВЦ РАН и ИППИ, сегмент европейской грид-

сети. Доступ к этим ресурсам и их взаимодействие осуществлялось с помощью системы MathCloud.

3.3 Эксперименты

Вычисления проводились на полиномах со случайным образом сгенерированными коэффициентами. Максимальная степень полиномов была принята равной 100. Опишем результаты экспериментов в виде трех таблиц со степенью полиномов, равной 25, 50 и 100.

Таб 2. Экспериментальные данные, степень полинома 25

	время вычисления, мсек
Примитивный метод	5.1
Улучшенный примитивный метод	2.1
Схема Горнера 1	7
Схема Горнера 2	1.4
Схема для специального класса полиномов	1.7

Таб 3. Экспериментальные данные, степень полинома 50

	время вычисления, мсек
Примитивный метод	10.3
Улучшенный примитивный метод	4.2
Схема Горнера 1	12.6
Схема Горнера 2	2.8
Схема для специального класса полиномов	3.4

Таб 3. Экспериментальные данные, степень полинома 100

	время вычисления, мсек
Примитивный метод	20.3
Улучшенный примитивный метод	8.2
Схема Горнера 1	28
Схема Горнера 2	5.5
Схема для специального класса полиномов	6.8

Остальные эксперименты показали качественно аналогичную картину. Самой быстрой оказалась схема Горнера 2. Несколько медленнее проявила себя схема для полиномов с нижним множеством мультииндексов. За ней следует улучшенная примитивная схема, работающая более чем в 2 раза быстрее, чем примитивный аналог. Необходимость выбирать необходимую пару мономов на каждом шаге схемы Горнера 1 привела к тому, что данным алгоритм оказался самым медленным.

4. Заключение

Итак, в данной работе рассмотрены 5 схем вычисления значений многомерных полиномов в точке. Вычислительные эксперименты показали, что схема Горнера 2 является самой многообещающей их них. Это является следствием структуры алгоритма, позволяющей декомпозировать его на различные вычислительные узлы и успешно распараллеливать.

Дальнейшее исследование многомерных полиномов включает в себя разработку других методов, позволяющих вычислять их значение в точке. Схема Горнера не является единственной схемой вычисления значений одномерных полиномов. Таким образом, обобщение других алгоритмов (например, таких, как метод Эстрина) также может оказать существенное влияние на исследование данной задачи.

Литература

[1] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Введение в алгоритмы (3^{e} издание). MIT Press and McGraw-Hill, 2009.

[2] Волков С.Ю. Схема Горнера и численное решение систем обыкновенных дифференциальных уравнений с полиномиальной правой частью. Теория и практика системного анализа: труды второй всероссийской научной конференции молодых ученых с международным участием, Рыбинск, 2012.

[3] Афанасьев А.П. Продолжение траекторий в оптимальном управлении. КомКнига, 2005

[4] Pena J.M., Sauer T. On the multivariate Horner scheme, Computing (Vienna/New York), v 65, 2000, стр. 313-322.

[5] Math Cloud. URL http://www.mathcloud.net

[6] Сухорослов О.В. Архитектура и реализация сервис-ориентированной научной среды MathCloud // XIII Российская конференция с участием иностранных ученых "Распределенные информационные и вычислительные ресурсы" (DICR'2010), Новосибирск, 2010.