УДК 519.688

ТЕХНОЛОГИЯ ЧИСЛЕННО–АНАЛИТИЧЕСКОГО ДИФФЕРЕНЦИРОВАНИЯ

И. С. Григорьев[1], А. И. Проскуряков[2]

Рассматриваются известные подходы вычисления значений производных функций многих переменных, аналитические выражения которых, полученные с использованием символьных вычислений, излишне громоздки. Предлагается работоспособная объектно–ориентированная технология численно-аналитического дифференцирования.

Ключевые слова: Производные сложных функций, численно–аналитическое дифференцирование.

Постановка задачи. Пусть задана функция $f(x_1, \ldots, x_n)$ многих переменных. Требуется вычислить ее значения и значения ее частных производных первого, второго и т.д. порядков. Предполагается, что аналитический вид функции $f(x_1, \ldots, x_n)$ известен, функция обладает требуемой гладкостью и может быть представлена как сложная функция, являющаяся суперпозицией некоторого набора функций φ_i, частные производные которых вычисляются по известным аналитическим формулам. В последующем набор функций φ_i называется базовым. Вычисление должно быть по-возможности точным и быстрым; код программы должен быть "читаемым".

Для решения поставленной задачи предлагается использовать специально разработанную технологию численно-аналитического дифференцирования. Эта технология основана на идее применения объектно-ориентированного программирования к графовому подходу, который был успешно использован при решении технической подзадачи, возникшей в процессе численного решения задачи оптимального управления выведением космического аппарата с опорной орбиты искусственного спутника Земли

[1]Московский государственный университет им. М. В. Ломоносова, Механико–математический факультет, Ленинские Горы, 119899, Москва; доцент, e-mail: iliagri@mail.ru

[2]Московский государственный университет им. М. В. Ломоносова, Механико–математический факультет, Ленинские Горы, 119899, Москва; аспирант и Филиал Московского государственного университета им. М. В. Ломоносова в городе Баку, AZ1144, Баку; старший лаборант, e-mail: ap_91@mail.ru

на орбиту искусственного спутника Луны с ограничениями на радиусы переселения и апоселения [3]. Решение задачи оптимального управления на основе принципа максимума Л.С. Понтрягина сводилось к краевой задаче для системы обыкновенных дифференциальных уравнений 14-го порядка. Краевая задача решалась численно методом стрельбы. Громоздкие производные возникали на этапе получения условий трансверсальности на правом конце траектории при вычислении частных производных терминанта. Упомянутая техническая подзадача была далеко не самой главной трудностью решения [3], а одной из многих; из-за ограниченности размера публикации многие технические детали реализации эффективного алгоритма в [3] отражения не нашли.

Технология численно-аналитического дифференцирования является одним из вариантов реализации автоматического дифференцирования (Automatic Differentiation) [18] с использованием объектно-ориентированного программирования. Автоматическому дифференцированию и различным вариантам его реализации посвящено большое число работ, однако каждая из таких реализаций обладает своими недостатками и широкого распространиния при решении прикладных задач не нашла. Предлагаемая в работе технология численно-аналитического дифференцирования успешно применялась при решении задач [14, 15, 16, 17].

Математические пакеты и символьное дифференцирование. Стандартным способом решения задачи вычисления производных является использование какого-либо математического пакета. С использованием символьных вычислений (во всех математических пакетах блок символьных вычислений по-сути один и тот же) получаются аналитические выражения производных, тем или иным способом они переводятся в программный код. Интерес представляет ситуация, когда эти производные излишне громоздки. В этом случае, если математический пакет не предоставляет возможности автоматической конвертации формул в программный код, написание программы оказывется затруднительным, а если предоставляет, то программа оказывается технически не исполнимой.

При решении упомянутой технической подзадачи попытка использовать символьное дифференцирование в математическом пакете оказалась неуспешной. Формулы частных производных оказались громоздкими. Пакет позволял получить эти производные в виде программного кода (по нескольку десятков строк каждая), но на этапе

компиляции программы возникала ошибка переполнения стека оптимизации и исполняемый код не создавался, то есть компиляция программы оказывалась невозможной.

Отметим, что ограничений применения технологии численно-аналитического дифференцирования по сложности формул определить не удалось. В одной прикладной задаче с производными до нескольких тысяч строк программного кода на формулу (более 60 плотно заполненных печатных страниц для одной производной) применение технологии численно-аналитического дифференцирования оказалось успешным.

Численное дифференцирование. Другим стандартным способом решения задачи является численное дифференцирование. Отметим, что во многих задачах (в том числе и в упомянутой технической подзадаче [3]) требования к точности вычисления производных значительно превосходят достижимую точность таких формул. То есть подход численного дифференцирования оказывается неприменим.

Комплексификация (Complex Step Differentiation). При решении задачи вычисления производных может использоваться теория функций комплексного переменного. Предполагается, что функция имеет гладкое расширение на комплексную область. Вычислению производных с использованием комплексных чисел посвящён ряд работ [8, 9, 10].

Суть одного из возможных вариантов этого подхода рассмотривается для случая вычисления первой производной функции одной переменной. Её разложение в ряд Тейлора в окрестности требуемой точки x имеет вид:

$$f(x + i \cdot h) = f(x) + f'(x)h \cdot i - f''(x)\frac{h^2}{2} - f^{(3)}(x)\frac{h^3}{6}i + \dots \tag{1}$$

В (1) i обозначает мнимую единицу. Из (1) следует

$$f'(x) = \frac{\operatorname{Im} f(x + i \cdot h)}{h} + O(h^2).$$

Выбор величины h осуществляется так, что $f^{(3)}(x)\frac{h^2}{6}$ является величиной, имеющей порядок шага сетки в окрестности величины $f'(x)$ для используемого при вычислениях типа с плавающей точкой. Например, при близких по величине значениях первой и третьей производных и использовании комплексных чисел на основе типа двойной точности машинный эпсилон (шаг сетки в окрестности единицы) составляет величину порядка 10^{-16} и потому выбирается величина $h \approx 10^{-8}$.

В случае необходимости вычисления первых производных по нескольким переменным требуется провести соответствующее число вычислений функции.

В большинстве работ, посвященных вычислению производных с использованием комплексификации, отмечается, что рассмотренный выше подход с использованием формулы Тейлора наиболее эффективен для вычисления производной первого порядка. Данный подход можно распространить на вычисление производных более высокого порядка (в том числе частных производных функции многих переменных), однако из-за влияния эффекта потери точности, который возникает, поскольку в выражениях для производных появляются разности двух близких значений функции, эффективность этого подхода может оказаться недостаточной. Модификация этого подхода, призванная уменьшить погрешность, за счет увеличения порядка метода представлена в [9].

Еще одним подходом вычисления производных при помощи комплексификации является использовании интегральной формулы Коши [8, 10]:

$$f^{(n)}(z) = \frac{n!}{2\pi i} \int\limits_{\Gamma} \frac{f(\xi)}{(\xi - z)^{n+1}}\, d\xi.$$

Интеграл вычисляется численно с использованием той или иной квадратурной формулы. В качестве недостатков данного подхода можно указать наличие погрешности численного интегрирования и необходимость проведения большого числа вычислений функций для достижения высокой точности.

Наконец, отметим работу [6], в которой описан метод вычисления производных с использованием мультикомплексных чисел.

Графовый подход. Теоретической основой многих эффективных подходов является представление процесса вычисления исходной функции в виде ориентированного графа. В каждой вершине такого графа хранятся значения функций и их производных вплоть до требуемого порядка.

Продемонстрируем этот процесс на примере вычисления значения и производных функции

$$f(x_1, x_2, x_3) = x_1 \sin(x_1 + x_2) + x_3 \cos(x_1 x_2). \tag{2}$$

Базовым набором функций в рассматриваемом примере являются $\varphi_1(a, b) = a + b$ (сложение), $\varphi_2(a, b) = a \cdot b$ (умножение), $\varphi_3(a) = \cos(a)$ (вычисление функции cos), $\varphi_4(a) = \sin(a)$ (вычисление функции sin).

Функцию f_k^m, сопоставленную вершине графа, будем называть соответствующей данной вершине. Для вершин и соответствующих им функций введена двойная нумерация; верхний индекс обозначает номер слоя, нижний индекс - номер функции в слое; двойная нумерация не является обязательной, вершины графа могут быть пронумерованы произвольным образом. Две вершины соединены ребром, если функция, соответствующая вершине из слоя с меньшим номером, является аргументом функции, соответствующей вершине из слоя с большим номером (ребро направлено от вершины из слоя с меньшим номером в вершину из слоя с большим номером).

Вершинам нулевого слоя графа сопоставляются переменные $x_1, \ldots, x_n$.

Вершинам первого слоя сопоставляются функции f_j^1, аргументами которых являются вершины нулевого слоя. В рассматриваемом примере это операции сложения и умножения $f_1^1 = \varphi_1(x_1, x_2) = x_1 + x_2$, $f_2^1 = \varphi_2(x_1, x_2) = x_1 x_2$, а, например, функция $\cos(x_1 x_2)$ не может быть сопоставлена вершинам первого слоя, поскольку ее аргументом является не переменная.

Вершинам второго слоя сопоставляются функции, аргументами которых являются вершины нулевого и первого слоев. В рассматриваемом примере это функции $f_1^2 = \varphi_4(f_1^1) = \sin(x_1 + x_2)$, $f_2^2 = \varphi_3(f_2^1) = \cos(x_1 x_2)$.

Вершинам i-ого слоя графа сопоставим функции аргументами которых, могут быть вершины всех предшествующих слоев — с нулевого до $i-1$-го слоя.

Последний слой графа состоит из одной вершины, которой сопоставлена сама исходная функция.

Выпишем функции, сопоставленные оставшимся вершинам графа из рассматриваемого примера:

$$
\begin{aligned}
&f_1^3 = \varphi_2(x_1, f_1^2) = x_1 \sin(x_1 + x_2),\\
&f_2^3 = \varphi_2(x_3, f_2^2) = x_3 \cos(x_1 x_2),\\
&f(x_1, x_2, x_3) = f_1^4 = \varphi_1(f_1^3, f_2^3) = x_1 \sin(x_1 + x_2) + x_3 \cos(x_1 x_2).
\end{aligned}
$$

Граф, используемый для вычисления значения и производных этой функции, представлен на рисунке 1.

Вычисление значений и производных в графе производится по слоям.

На нулевом слое известны значения переменных и все требуемые значения производных: одна первая производная по соответствующей переменой равна единице, все

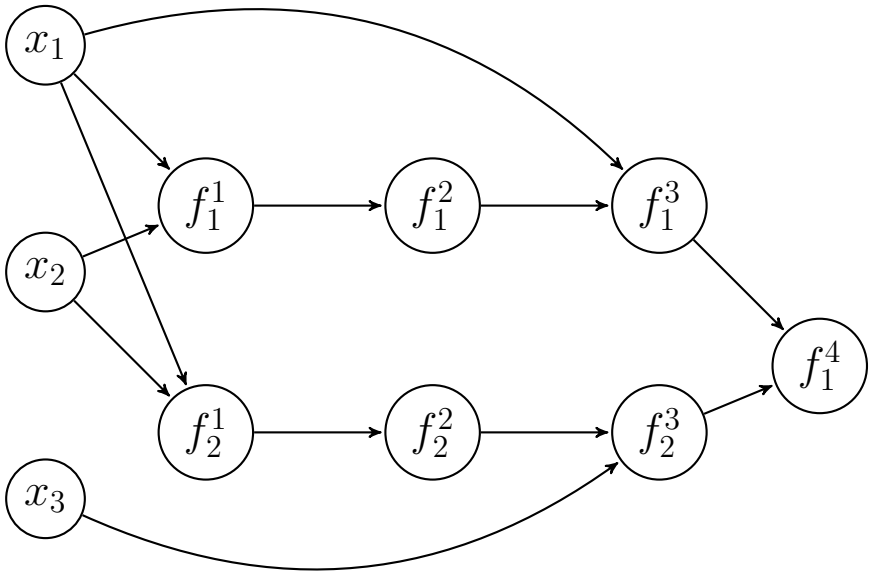

Рис. 1: Пример графа, для вычисления частных производных

остальные производные, в том числе и все производные высоких порядков, равны нулю.

Так как правила вычисления производных для функций из базового набора известны и известны все значения и все необходимые производные на предыдущих слоях, то значения и производные в вершинах на текущем слое могут быть вычислены.

Идеи использования графового подхода для вычисления градиента функций были описаны в работах [1, 2, 4] и др. Идея возможности вычисления функции многих переменных и ее производных с использованием базового набора операций и рекурсивная реализация этой идеи рассматривалась в [7] (под названием автоматическое дифференцирование).

Упомянутая в начале статьи техническая подзадача из [3] была решена с использованием графового подхода. Представленные формулы были развернуты в виде графа, для каждой из вершин этого графа введена переменная–значение и соответствующий ей массив переменых для хранения градиента. Основной недостаток такой реализации графового подхода состоял в громоздкости получаемой программы и в проблемах тестирования полученного кода для исключения ошибок программирования. Отметим, что технология численно-аналитического дифференцирования этот недостаток устраняет.

Технология численно-аналитического дифференцирования. Основная идея представленной работы состоит в использовании для реализации графового подхода на основе объектно-ориентированного программирования — создания специально разработанного класса **“расширенное число”** (`ext_value` — новый тип данных). Допустимо считать, что технология численно-аналитического дифференцирования является расширением технологии дуальных чисел [11, 13], а дуальные числа — сужение и упрощение технологии численно-аналитического дифференцирования на случай вычисления только первой производной функции одной переменной.

В объекте указанного типа **“расширенное число”** выделяется память для хранения значения и требуемых производных, описываются функции базового набора (операции, которые с этими данными можно проводить). Написание класса “расширенное число” необходимо один единственный раз. После того как класс “расширенное число” написан, достаточно подключить заголовочный файл и написать программу самого вычисления функции, вычисление значений производных при этом дополнительного программирования не потребует.

Оптимизация объёма памяти. Для уменьшения объёма вычисляемой и хранимой информации воспользуемся свойством симметричности старших производных:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i},$$
$$\frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k} = \frac{\partial^3 f}{\partial x_i \partial x_k \partial x_j} = \frac{\partial^3 f}{\partial x_j \partial x_i \partial x_k} =$$
$$= \frac{\partial^3 f}{\partial x_j \partial x_k \partial x_i} = \frac{\partial^3 f}{\partial x_k \partial x_i \partial x_j} = \frac{\partial^3 f}{\partial x_k \partial x_j \partial x_i}.$$

Таким образом при хранении старших производных объём необходимой памяти может быть сокращен. Предлагается хранить значимые значения симметричной матрицы вторых производных по столбцам в верхней правой части матрицы до главной диагонали включительно: $f_{x_0x_0}$, $f_{x_0x_1}$, $f_{x_1x_1}$, $f_{x_0x_2}$, $f_{x_1x_2}$, $f_{x_2x_2}$, $f_{x_0x_3}$, $f_{x_1x_3}$, $f_{x_2x_3}$, … При дифференцировании по n независимым переменным всего $n(n+1)/2$ элементов.

При хранении в линейном массиве формула вычисления индекса в линейном массиве по индексам переменных, по которым производится дифференцирование:

$$m = i + j(j+1)/2, \quad \forall\ 0 \le i \le j < n\_var.$$

Разумеется, так как при реализации используется язык C++, индексация в массивах

начинается с 0.

Для третьих производных также предполагается хранить элементы в упорядоченном виде согласно упорядоченным индексам переменных, по которым производилось дифференцирование: $0 \le i \le j \le k$. Как и для вторых производных значения производных в линейном массиве треьих производных упорядочены. Сначала производится сравнение наибольшего индекса, производная, у которой этот индекс меньше, находится в массиве левее. В случае равенства этих индексов, сравниваются следующие по величине индексы. Таким образом начальный участок линейного массива третьих производных имеет вид: $f_{x_0x_0x_0}$, $f_{x_0x_0x_1}$, $f_{x_0x_1x_1}$, $f_{x_1x_1x_1}$, $f_{x_0x_0x_2}$, $f_{x_0x_1x_2}$, $f_{x_1x_1x_2}$, $f_{x_0x_2x_2}$, $f_{x_1x_2x_2}$, $f_{x_2x_2x_2}$, ...

Формула для вычисления индекса в линейном массиве третьих производных по номерам переменных, по которым производится дифференцирование, в соответствии с введённым порядком имеет вид:

$$m = i + j(j+1)/2 + k(k^2+k+2)/4, \quad \forall\ 0 \le i \le j \le k < n\_var.$$

Аналогичные формулы можно получить и для производных большего порядка.

Замечания по реализации. При реализации технологии можно выделить частные случаи:

1) требуется вычисление только первых производных многих переменных,

2) требуется вычисление и первых, и вторых производных многих переменных,

3) требуется вычисление до третьих производных включительно многих переменных,

4) требуется вычисление производных по одной независимой переменной (необходимость хранения массивов пропадает),

5) требуется вычисление старших производных по двум независимым переменным (для производных любого порядка формула размещения значительно упрощается — индекс в линейном массиве (начиная с нуля) может быть получен суммированием индексов переменных, по которым производится дифференцирование (как и ранее индексы переменных в этом случае 0 для первой переменной и 1 для второй)).

В случае написания кода “нижнего уровня” следует избегать любых лишних действий. Так, если требуется умножить некоторое число на константу, не следует константу приводить к виду расширенного числа с нулевыми производными (в классе явно прописаны соответствующие операторы сложения, вычитания, умножения, деле-

ния с сокращенным числом действий). Не желательно использовать в соответствующей программе операторы с выделяемой динамически памятью, создающие (и потом удаляющие) копий — главный бич быстродействия. Для этого после отладки программы можно использовать "экономные" операции `+=`, `-=`, `*=`.

Заключение. Тестирование классов, реализующих технологию численно-аналитического дифференцирования проводилось на примерах, в которых аналитические формулы производных не слишком громоздки и могут быть откомпилированы. В тестирующей программе считались разности значений, полученных с использованием реализации класса `ext_value` и по аналитическим формулам. Вычисленные разности имели порядок погрешности представления чисел.

Итак, технология численно-аналитического дифференцирования позволяет упростить программирование в задачах, где требуется вычисление громоздких производных. Она реализована на языке программирования C++ и доступна для свободного использования. Страница проекта:
`http://mech/math.msu.su/~iliagri/ext_value.html`

Список литературы

[1] Воеводин В.В., Воеводин Вл.В. "Параллельные вычисления", Санкт-Петербург, БХВ-Петербург, 2002.

[2] Горбань А.Н., Сенашова М.Ю. "Быстрое дифференцирование, двойственность и обратное распространение ошибки" // Вычислительные технологии, том 4, специальный выпуск, 1999.

[3] Григорьев К.Г., Григорьев И.С. "Исследование оптимальных пространственных траекторий перелетов космического аппарата с реактивным двигателем большой ограниченной тяги между орбитами искусственных спутников Земли и Луны" // Космические исследования. 1997. Т. 35. № 1. С. 52–75.

[4] Сенашова М.Ю. "Оценки погрешностей вычисления сложной функции многих переменных и ее градиента" // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2007. — Т. 10, № 1. — С. 77–88.

[5] Справочное руководство по небесной механике и астродинамике. Под ред. Дубошина Г.Н.: Наука, 1976.

[6] Adriaen Verheyleweghen. “Computation of higher-order derivatives using the multi-complex step method”. December 6, 2014.

[7] Dan Kalman. Doubly recursive multivariate automatic differentiation. Mathematics Magazine, 73(3): pp. 187–202, 2002

[8] James N Lyness and Cleve B Moler. “Numerical differentiation of analytic functions”. SIAM Journal on Numerical Analysis, 4(2):202–210, 1967

[9] Kok-Lam Lai, L. Crassidis. “Generalizations of the Complex-Step Derivative Approximation”. University at Buffalo, State University of New York, Amherst, NY 14260-4400.

[10] Martins, J. R. R. A., Kroo, I. M., and Alonso, J. J., “An Automated Method for Sensitivity Analysis Using Complex Variables” American Institute of Aeronautics and Astronautics, 2000, AIAA 2000-0689.

[11] Corliss G., Faure C., Griewank A., Hascолt L., Naumann U. Automatic Differentiation Bibliography // Automatic Differentiation of Algorithms: From Simulation to Optimization. Springer, 2002. PP. 383-425.

[12] Neidinger Richard D. Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming // SIAM REVIEW, Vol. 52, No. 3, pp. 545-563, 2010

[13] Семёнов К.К. Автоматическое дифференцирование функций, выраженных программным кодом // Изв. Вузов. Приборостроение. Т. 54, № 12. 2011.

[14] Григорьев И.С., Проскуряков А.И. Оптимизация целевой орбиты и траектории апсидального импульсного выведения космического аппарата на нее с учетом сброса отработавших ступеней в атмосферу // Инженерный журнал: наука и инновации, 2019, вып. 4.
http://dx.doi.org/10.18698/2308-6033-2019-4-1869

[15] Григорьев И.С., Проскуряков А.И. Импульсные перелеты космического аппарата со сбросом ступеней в атмосферу и фазовым ограничением (часть I) // Инженерный журнал: наука и инновации, 2019, вып. 9.
http://dx.doi.org/10.18698/2308-6033-2019-9-1917

[16] Григорьев И.С., Проскуряков А.И. Импульсные перелеты космического аппарата со сбросом ступеней в атмосферу и фазовым ограничением (часть II) // Инженерный журнал: наука и инновации, 2019, вып. 10.
http://dx.doi.org/10.18698/2308-6033-2019-9-1925

[17] Proskuryakov A.I. Comparison of apsidal and non-apsidal impulse trajectories of the spacecraft launching to the target orbit, taking into account the reset of stages into the atmosphere // AIP Conference Proceedings 2171, 060008 (2019)
http://doi.org/10.1063/1.5133206

[18] Bartholomew-Biggs M., Brown S., Christianson B., Dixon L. Automatic differentiation of algorithms // Journal of Computational and Applied Mathematics. 124 (2000). PP. 171–190