К. В. Михайлов

Программная реализация процедур многомерной аппроксимации

Научный руководитель: д.т.н. профессор В. И. Гурман

Аннотация. В данной статье предложена программная реализация процедуры аппроксимации функций многих переменных. Описан алгоритм работы программы и структуры данных, которые используются. В статье приведены результаты тестирования программы апроксимации функций зависящих от 1 до 4 переменных, а также обозначены проблемы, возникающие при реализации описанного алгоритма.

Ключевые слова и фразы: программная реализация, многомерная аппроксимация, функции многих переменных.

1. Введение

Процедуры многомерной аппроксимации имеют обширную область применения, в частности, требуются для представления имитационных моделей.

Часто требуется выполнить аппроксимацию той или иной многомерной функции, заданной таблицей значений на своей области определения [1]. Обычно эти значения известны не точно, а приближенно, так как определяются экспериментально и могут содержать ошибки измерения. Программа, решающая задачу аппроксимации такой функции, может использоваться как модуль, встраиваемый в сложный программный комплекс, а также может работать самостоятельно.

В связи с тем, что непосредственная аналитическая реализация как правило невозможна, современные исследователи вынуждены использовать аналитическую аппроксимацию функций многих переменных для применения теоретических методов. Конструктивных методов для реализации задачи многомерной аппроксимации предложено мало, и данная работа направлена на то, чтобы восполнить этот пробел.

Работа выполнена при финансовой поддержке РФФИ, (проект № 08-01-00274-а).

2. Постановка задачи

Целью данной работы является реализация на языке программирования Си процедуры аппроксимации функции многих переменных, заданной своими значениями на некоторой сетке узлов, методом наименьших квадратов (МНК) [2], [3].

По этому методу должна быть минимизирована сумма квадратов разностей между значениями в узловых точках x_i приближаемой функции $f(x)$, $x \in R^n$ и приближающей

$$\varphi(x, \{\psi_j\}), \tag{1}$$

где $\{\psi_j\}$ –набор коэффициентов, которые подлежат определению, $j = 1, \ldots, \alpha$.

Иными словами, требуется решить следующую задачу минимизации:

$$S(\{\psi_j\}) = \sum_{i=1}^{\beta} (\varphi(x_i, \psi_j) - f(x_i))^2 \to \min_{\{\psi_j\}}, \tag{2}$$

где β - количество узлов.

Для практической реализации во многих случаях удобно использовать регулярную (прямоугольную) сетку узлов и следующую конструкцию, представляющую собой композицию одномерных полиномов [4]:

(3)

$$\varphi(t, x(t)) = \sum_{j_1=1}^{m_1} (x_1(t))^{j_1} \times \times \left(\sum_{j_2=1}^{m_2} (x_2(t))^{j_2} \left(\cdots \sum_{j_n=1}^{m_n} \psi_{j_1, j_2 \ldots j_n}(t) (x_n(t))^{j_n} \right) \right),$$

где $m_1, m_2, \ldots m_n$ — количество узловых точек по каждой из координат. Если, в частности, число $m_i - 1$ совпадает с порядком i-го одномерного полинома, то значения обеих функций в узлах совпадут, и будет решена задача интерполяции.

3. Алгоритм работы программы

Программа на языке Си решает задачу минимизации 2.

Программа запрашивает у пользователя размерность приближаемой функции $f(x)$, границы интервала, в котором задана каждая

переменная, и степень аппроксимирующего полинома по каждой переменной m_i.

На первом шаге алгоритма, реализованного в программе, выполняется построение регулярной сетки узлов аппроксимации и, далее, вычисляются значения функции $f(x)$ в узлах.

На втором шаге алгоритма строятся аппроксимирующие полиномы по каждой переменной. Эти полиномы используются для построения аппроксимирующей конструкции 3.

На третьем шаге алгоритма выполняется построение функции $S(\{\psi_j\})$ и нахождение её частных производных по ψ_j.

Для выполнения четвёртого шага алгоритма аппроксимации работает модуль программы, реализующий метод Гаусса решения системы линейных уравнений относительно ψ_j.

На пятом шаге вычисляются значения функций $\varphi(x_i, \psi_j)$ в узловых точках.

4. Реализация структур данных

Аппроксимирующие полиномы представлены в программе односвязными списками. Количество элементов списка определяется степенью полинома, его элементами являются структуры, описывающие мономы. Каждая такая структура состоит из следующих полей [5]:

- коэффициент $\{\psi_j\}$, вещественное число;
- степень x_i, целое число;
- x_i – значение узловой точки, вещественное число;
- указатель на следующий моном полинома.

```
struct monom_phi {
       float psi;
       int ind;
       float x;
       monom_phi * next;
};
```

Функция $S(\{\psi_j\})$ представлена в программе с помощью списка, элементами которого являются структуры, описывающие квадрат разности приближающей и приближаемой функции.

```
struct monom_S {
       monom_phi* phi_i;
       float f_xi;
       int pow;
       monom_S* next;
```

```
};
```

Операции построения и дифференцирования функции $S(\{\psi_j\})$ реализованы как процедуры, выполняющиеся над линейными односвязными списками, которые представляют собой функции $\varphi(x_i, \psi_j)$ и $S(\{\psi_j\})$.

В качестве примера в программе рассматривается аппроксимация по МНК функции

$$f(x) = \cos(\sum_{i=1}^{n} x^i) + \sin(\prod_{i=1}^{n} x^i)^2 + (\sum_{i=1}^{n} ix^i)/5$$

на промежутке $0 \leq x \leq 1$.

5. Тестирование и проблемы, которые предстоит решить

В качестве аппроксимирующей конструкции выбран композиционный полином. Число узлов сетки равно количеству коэффициентов аппроксимирующего полинома, то есть решается задача интерполяции. Интерполяционный полином был выбран в качестве аппроксимирующей конструкции, так как он обеспечивает надёжные результаты аппроксимации: значения приближаемой функции и аппроксимирующего полинома совпадают в узлах сетки.

На сегодняшний день программа отлажена и протестирована для функции четырёх переменных с различным количеством узловых точек по каждой из переменных. Тестирование программы позволило выявить ограничения, связанные с вычислительными ресурсами, на

- количество переменных, от которых зависит приближаемая функция;
- количество узловых точек по каждой из переменных.

Увеличение количества узлов и размерности приближаемой функции влечёт за собой рост объёма оперативной памяти, используемой системой программирования, и времени счёта. Так, для функции четырёх переменных количество узловых точек по каждой из переменных не может превышать 6 для выполненных тестов. Тестирование производилось под управлением операционной системы Linux Debian 2.6.26-2-686, на компьютере со следующей конфигурацией:

- MB ASUS "Maximus Formula";
- Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz;
- 4x2Gb Kingston DDR2;
- HDD WD-WCAUK1186052 500Gb.

Таблица 1. Результаты тестирования программы

Переменные	**Узловые точки**	**Память, Мб**	**Время, с**
1	500	13,5	3
1	1250	83,7	55
1	1615	140	122
2	30x30	62	21
2	35x35	115	58
2	40x40	195,6	134
3	7x7x7	6,5	1
3	10x10x10	53,6	25
3	11x11x11	94,8	57
4	4x4x4x4	3,6	1
4	5x5x5x5	21	6
4	6x6x6x6	90	47

Результаты тестирования программы приведены в таблице 1.

Необходимо найти оптимальное соотношение между количеством переменных приближаемой функции и числом узлов по каждой переменной ради повышения эффективности вычислений, не ухудшая качества результата (в точности вычисленных значений, густоте сетки узлов).

Особенности реализации программы заключаются в том, что аппроксимирующая конструкция в каждой узловой точке представлена линейным односвязным списком, а весь набор таких конструкций представляет собой массив указателей на начала этих списков. Размер этого массива равен количеству неизвестных коэффициентов аппроксимирующей конструкции, количество коэфициентов равно степени аппроксимирующего полинома. Количество числовых значений, стоящих при искомых коэффициентах аппроксимирующей конструкции (в системе линейных уравнений), равно количеству узловых точек по каждой из переменных в степени, равной количеству переменных функции. Так, для функции десяти переменных с пятьюдесятью узловыми точками по каждой необходимо создать 50^{10} массивов для хранения 50^{10} вещественных значений в каждом массиве. Хранить такой объём информации на стеке системы программирования не представляется возможным. Увеличение стека shell, в котором проходит

вычисление (запуск программы), даёт возможность увеличить число узлов по каждой переменной, но незначительно, хотя существенно увеличивает время счёта. Одно из решений, которое позволяет увеличить количество узлов по каждой переменной приближаемой функции, заключается в том, чтобы все промежуточные результаты вычислений коэффициентов аппроксимирующей конструкции, а также значений приближаемой функции и аппроксимирующего полинома, записывать в файлы, а не хранить в оперативной памяти, которой располагает система программирования.

Такой подход позволяет увеличить число узлов по каждой переменной аппроксимируемой функции (а именно этот параметр оказывает большее влияние на увеличение объёма используемой памяти), но не является достаточно эффективным для решения задач аппроксимации, возникающих на практике. Такая версия программы позволяет выполнить аппроксимацию функции, например, пяти переменных с двадцатью узловыми точками по каждой из переменных, и ей потребуется около 130,3851 терабайт дискового пространства $(20^5 * 14/2^{20}) * 20^5$ – таков объём файлов для хранения информации, где в каждом файле 20^5 строк 14 символов для записи одного значения в файл). Запуск программы для аппроксимации функции пяти переменных с 10 узловыми точками по каждой из них потребует около 130 гигабайт дискового пространства, и это является приемлемым для проведения расчётов на персональном компьютере.

Другим решением проблемы недостатка оперативной или внешней памяти для хранения промежуточных результатов вычислений может стать разбиение всей области, на которой выполняется аппроксимация, на подобласти, в каждой из которых по каждой переменной выбирается две узловые точки, и в качестве аппроксимирующей конструкции рассматривается композиция линейных сплайнов.

В дальнейшем планируется параллельная реализация программы и добавление возможности использовать переменный (нерегулярный) шаг по каждой из переменных приближаемой функции.

Список литературы

[1] Пармёнова Л. В. *Программная реализация процедур многомерной аппроксимации* // Молодежный симпозиум с международным участием «Теория управления: новые методы и приложения». — Переславль-Залесский: Изд-во «Университет города Переславля», 2009. — ISBN 978–5–901795–22–4, с. 51–54.

[2] Гурман В. И. Принцип расширения в задачах управления. — Москва: Наука. Физматлит, 1997. — 288 с.

[3] Хемминг Р. В. Численные методы. — 2-е изд., перераб. и доп. — М.: Наука. Физико-математическая библиотека инженера, 1972. — 400 с.

[4] Ухин М. Ю. Приближенный синтез дискретного оптимального управления. — М.: Физматлит, 2006.

[5] Михайлов К. В. *Реализация процедур многомерной аппроксимации на примере функции двух переменных* // Молодежный симпозиум с международным участием «Теория управления: новые методы и приложения». — Переславль-Залесский: Изд-во «Университет города Переславля», 2009. — ISBN 978–5–901795–22–4, с. 44–46.

УГП, 5И52

K. V. Mikhailov. *Program implementation of approximation algorithm for functions of n-variables* // Proceedings of Junior research and development conference of Ailamazyan Pereslavl university. — Pereslavl, 2010. — p. 48–54. (*in Russian*).

Abstract. The article is dedicated to program implementation of approximation algorithm for functions of n-variables. The algorithm and data structures are described. Test results of the program that approximates functions of 1–5 variables are presented. Also problems of implementation of approximation algorithm for functions of n-variables are indicated.

Key Words and Phrases: program implementation, approximation algorithm, functions of n-variables.