РАСПРЕДЕЛЕННЫЕ ИНФОРМ.-ВЫЧИСЛ. РЕСУРСЫ И МАТ. МОДЕЛИРОВАНИЕ
МКВМ-2004 РАБОЧИЕ СОВЕЩАНИЯ С. 119–125

Параллельное вычисление гарантированных границ решений дифференциальных уравнений

А.Н. Рогалев*

Аннотация. Доклад посвящен исследованию свойств параллельных алгоритмов решения систем обыкновенных дифференциальных уравнений (ОДУ), коэффициенты и начальные данные которых заданы неточно, известны лишь границы (интервалы), содержащие эти величины. Параллельные формы алгоритмы основаны на символьном представлении формул, аппроксимирующих оператор сдвига вдоль траектории дифференциальных уравнений, с последующим вычислением областей, содержащих все возможные значения. Выделены базовые алгоритмические структуры, которые ориентированы на параллелизм по данным и учитывают последовательное исполнение символьных преобразований формул.

1. Введение

Методы построения гарантированных оценок множеств решений систем ОДУ основаны на построении (хранении и преобразовании) символьных формул, аппроксимирующих оператор сдвига вдоль траектории системы ОДУ. Поскольку при этом естественно исполнять операции символьных преобразований так, чтобы обеспечить эффективную загруженность памяти и уменьшить время исполнения, встает задача оптимизации символьных форм алгоритмов. Для этих целей в статье предлагается параллельное исполнение символьных преобразованиях. Пусть в задаче удалось выделить однотипные операции над данными, которые эффективно реализуются на компьютере с параллельной архитектурой. Тогда анализ структуры алгоритма и его оптимизацию достаточно провести на уровне этих базовых операций. Решение этой задачи для символьных формул гарантированного метода во многом похоже на анализ алгоритмов вычисления арифметических выражений [1–3]. В самом деле, пусть базовые операции бинарны. Для того, чтобы можно было надеяться на распараллеливание вычислений, требуется наличие свойств ассоциативности хотя бы у одной базовой операции. если же она обладает дополнительно свойством коммутативности или, более того, определяет групповую структуру на множестве данных, есть основания рассчитывать на ощутимый эффект от оптимизации алгоритма. Учет связей между разнотипными операциями приводит к дальнейшему улучшению исходного алгоритма. Все сказанное относится как к обычным алгоритмам численного анализа, так и к гарантированным оценкам множеств решений, основанным на преобразовании символьных формул.

*Институт вычислительного моделирования СО РАН, Красноярск.

2. Параллельная форма символьных формул и ее свойства

Постановки многих задач решения дифференциальных уравнений содержат данные (например, коэффициенты системы, или начальные данные, или краевые условия), значения которых неизвестны. При этом заданы лишь границы, в пределах которых могут изменяться эти данные, учитывать необходимо каждое из значений данных. В основе построения гарантированных символьных методов лежит символьная формула аппроксимации оператора сдвига вдоль траектории.

Напомним некоторые понятия, на которых основано описание символьных этапов гарантированного оценивания множеств решений дифференциальных уравнений [4–9].

Определим символьную формулу (аналитическое выражение) как последовательность имен переменных и знаков операций, которые нужно проделать в определенном порядке над значениями этих переменных, чтобы получить значение выражения. В силу этого символьный метод (аналитический метод) – запись численного метода как метода преобразования символьной информации (символьных формул) на языке математического анализа. В дальнейшем при записи символьных формул, аппроксимирующих оператор сдвига вдоль траектории, допускается включение в них числовых констант, с отложенным выполнением арифметических действий над ними.

В этом символьный метод отличается от численного алгоритма, основанного на исполнении конечной последовательности действий над конечным множеством чисел. Чтобы строить символьные формулы, аппроксимирующие оператор сдвига вдоль траектории ОДУ и позволяющие получать достаточно точные включения множеств решений (например, интервальные расширения), необходимо получить формулу, хорошо приближающую точное решение, и выполнить алгоритм преобразования этой формулы. Следует также учесть, что требование устойчивости разностных методов как непрерывной зависимости их решений от возмущений входных данных трансформируется в случае реализации интервальных символьных методов.

Пусть H_i – последовательность пространств; $\mathcal{F}^i$, $i = 1, \ldots, n-1$, – последовательность символьных формул непрерывных отображений F^i, таких, что F^i определены на прямом произведении $H_1 \times H_2 \times \ldots \times H_i$, отображают это произведение в пространство H_{i+1} и задают зависимость между значениями решения в каждой точке области определения и начальными значениями. Тогда результат последовательного исполнения преобразований формул

$$
\begin{aligned}
&\mathcal{Y}^1 = \mathcal{F}^1(t^0, t^1, \mathcal{Y}', \mathcal{Y}^1) = \mathcal{S}^1(\mathcal{Y}^0),\\
&\mathcal{Y}^2 = \mathcal{F}^2(t^0, t^1, t^2, \mathcal{Y}^0, \mathcal{Y}^1, \mathcal{Y}^2) = \mathcal{S}^2(\mathcal{Y}^0) \circ \mathcal{S}^1(\mathcal{Y}^0),\\
&\ldots\\
&\mathcal{Y}^i = \mathcal{F}^i(t^0, \ldots, t^i, \mathcal{Y}^0, \mathcal{Y}^1, \ldots, \mathcal{Y}^i) = \mathcal{S}^i(\mathcal{Y}^0) \circ \mathcal{S}^{i-1}(\mathcal{Y}^0) \circ \mathcal{S}^1(\mathcal{Y}^0), \qquad (1)\\
&\ldots\\
&\mathcal{Y}^m = \mathcal{F}^m(t^0, \ldots, t^m, \mathcal{Y}^0, \mathcal{Y}^1, \ldots, \mathcal{Y}^m) = \mathcal{S}^m(\mathcal{Y}^0) \circ \mathcal{S}^{m-1}(\mathcal{Y}^0) \circ \cdots \circ \mathcal{S}^1(\mathcal{Y}^0)
\end{aligned}
$$

будет называться символьной формулой метода сдвига вдоль траектории решения системы ОДУ [4].

Очевидно, что для большинства методов, $H_i = R^i, i = 1, 2, 3, \ldots, m-1$, и $H_m = R^m$, где R^m есть m-мерное евклидово пространство, и $\mathcal{F}_i$ – формула отображения, основанного на элементарных арифметических операциях.

Алгоритм исполнения этого метода особенно просто выглядит в случае применения явных разностных схем. Применяя последовательные подстановки и приведение подобных членов в формуле (1) можно перейти к выражению, зависящему только от y^0. В общем случае в методе предлагается модель вычислений (преобразований и вычислений) символьных формул, основанная на поэтапном статичном хранении информации и преобразовании ее в завершающей стадии метода. Таким образом, формула будет представлять рекурсивную структуру, размер которой изменяется. Для записи такой формулы в компьютере используются линейные динамические структуры [10].

В силу этого модель вычислений (преобразований и вычислений) символьных формул осуществляется без явного выписывания суперпозиций компонент формулы, определяемых на каждом шаге. Связь между этими компонентами определяется посредством задания механизма адресации. Ссылки на адреса различных уровней хранятся в стековой памяти в виде дерева. Генерация кода вычислений по формуле (1) осуществляется в процессе обхода этого дерева, начиная с вершин.

В рассматриваемом классе символьных выражений операнды обозначены буквами каллиграфического шрифта, присутствуют также знаки арифметических операций и числовые константы. Однотипные операции сложения и вычитания будем называть аддитивными, а умножения и деления – мультипликативными. По записи символьного выражения $\mathcal{A}$ однозначно строится бинарное дерево $\mathcal{T}_\mathcal{A}$, определяющее алгоритм его вычисления. Для построения $\mathcal{T}_\mathcal{A}$ нужно одинаковые операнды задавать различными входными узлами графа алгоритма. По бинарному дереву $\mathcal{T}_\mathcal{A}$ однозначно восстанавливается символьное выражение $\mathcal{A}$, которое естественно назвать записью алгоритма $\mathcal{T}_\mathcal{A}$ или программой.

Обработка символьных формул

$$\mathcal{Y}^n = \mathcal{F}^n(t_0, t_1, \ldots, t^n, \mathcal{Y}^0, \mathcal{Y}^1, \ldots, \mathcal{Y}^n) = \mathcal{S}^n(\mathcal{Y}^0) \circ \mathcal{S}^{n-1}(\mathcal{Y}^0) \circ \cdots \circ \mathcal{S}^1(\mathcal{Y}^0)$$

производится по следующей методике. Пусть $\phi(y^0)$ – есть однозначное отображение единичного интервала из R^1 на гиперкуб из R^n, которое каждой точке $t \in R^1$ сопоставляет некоторую точку $y = \phi(t)$. При помощи такого отображения можно построить алгоритм исполнения, который для каждой точки $t \in R^1$ позволяет определить формулу отображения $\mathcal{F}(\mathcal{Y}_1^0, \mathcal{Y}_2^0, \ldots, \mathcal{Y}_n^0)$, и процесс ее сборки по адресам. Для этого предлагается использовать в качестве отображения $\phi(y^0)$ – непрерывное, однозначное отображение единичного интервала на n-мерный куб, известное также под названием кривой Пеано, заполняющей пространство. Фактически кривая Пеано представляет собой непрерывную, нигде не дифференцируемую кривую, которая проходит через все точки единичного гиперкуба в пространстве R^n. Изобразить кривую Пеано нельзя, возможно лишь дать последовательность кривых [10], которая в пределе сходится к ней. Каждая такая кривая называется приближением кривой Пеано и имеет номер, определяющий ее номер в последовательности кривых.

Таким образом, m-е приближение можно рассматривать как некоторую аппроксимацию m-й функции в рекурсивной формуле (1). Это соответствие задано отображением элементов конечного множества отрезков из единичного интервала и элементами конечного множества гиперкубов, входящих в R^n.

Ограничившись крупнозернистым параллелизмом и соответствующими ему многопроцессорными архитектурами, предлагается строить параллельную форму символьных формул приближенных решений, используя запись арифметического выражения в виде дерева и обработку поддеревьев с однотипными операциями.

Рассмотрим группу эквивалентных арифметических преобразований, в которую включим расстановку скобок и перестановку местами сомножителей в термах или слагаемых в подвыражениях, а также вынесение знаков вычитания или деления за скобку.

Пусть $\mathcal{S}_\mathcal{C} = \{\mathcal{PC},\ \mathcal{P} \in \pi\}$ – подмножество множества эквивалентных символьных формул арифметических выражений, получаемых из выражения $\mathcal{C}$ с помощью преобразований из π–группы эквивалентных арифметических преобразований. Поставим задачу нахождения оптимального выражения $\mathcal{C}^*$ на классе $\mathcal{S}_\mathcal{C}$:

$$\min_{\mathcal{D}\in\mathcal{S}_\mathcal{C}} h_{\mathcal{T}(d)} = h_{\mathcal{C}^*}. \tag{2}$$

Здесь h – глубина формулы. Далее решение задачи (2) обозначается через $\mathcal{C}^*$, где $\mathcal{C}$ – символьная форма произвольного выражения.

Алгоритм решения задачи (2) основан на обработке поддеревьев с однотипными операциями.

Узлам графа алгоритма припишем числа, называемые уровнями (уровнями подвыражений), равные высотам поддеревьев, для которых эти узлы являются вершинами. Рассмотрим лишь дерево $\mathcal{T}$ с аддитивными операциями. Это соответствует арифметическому выражению $A = A(A_1, A_2, \ldots, A_k)$, где $A_1, A_2, \ldots, A_k$ – подставляемые в A подвыражения, в котором помимо аргументов и скобок имеются лишь знаки сложения и вычитания. Путем обход дерева от корня к листьям нетрудно определить знаки операндов $A_1, A_2, \ldots, A_k$, с которыми их нужно просуммировать в соответствии с алгоритмом $\mathcal{T}$, чтобы получить выражение $A' = A'(A_1, A_2, \ldots, A_k) \in \mathcal{S}_\mathcal{C}$. Здесь A' – результат раскрытия скобок в A.

Оптимальное дерево $\mathcal{T}^*$ строится по схеме сдваивания, применяемой к набору вершин $t_1, \ldots, t_k$ деревьев $\mathcal{T}_{A_1}, \mathcal{T}_{A_2}, \ldots, \mathcal{T}_{A_k}$. Сначала каким-либо алгоритмом сортировки набор $t_1^0 = t_1, \ldots, t_k^0 = t_k$ упорядочивается по возрастанию уровней. Затем этот список просматривается слева направо и модифицируется. На первом проходе списка скобками выделяются пары, состоящие из узлов минимального уровня. Составляется новый список $t_1^1 = (t_1, t_2)$, $\ldots$, $t_l^1 = (t_{2l-1}, t_{2l})$, $t_{l+1}^1 = t_{2l+1}$, $\ldots$, $t_k^1 = t_k$. Парам $(t_1, t_2), \ldots, (t_{2l-1}, t_{2l})$ приписываются уровни, на единицу большие, чем у $t_1, t_2, \ldots, t_{2l}$. Если множество указанных пар пусто, то на первом проходе вершину минимального уровня t_1 дополняем множеством вершин следующего по порядку уровня $t_2, t_3, \ldots, t_{2l}$, и точно так же скобками слева направо выделяем пары в список $t_1^1, \ldots, t_{k'}^1$. На последующих проходах схема сдваивания применяется к спискам $t_1^1, \ldots, t_{k'(l)}^1$, $l = 1, 2, \ldots$ Этот процесс продолжается до нахождения списка из единственного

элемента. После завершения схемы сдваивания становится известным уровень вершины (высота) дерева $\mathcal{T}^*$, который равен сумме $r + \max_{i=1,2,\ldots,k} h_{\mathcal{T}(A_i)}$, где r – число тактов в только что изложенной схеме логарифмического суммирования.

Алгоритм решения задачи (2) состоит в обходах исходного дерева сверху вниз для поиска максимальных "концевых" поддеревьев с однотипными операциями. Каждое концевое поддерево оптимизируется на основании схемы сдваивания и затем отсекается. Последующая оптимизация производится с усеченным поддеревом.

Механизм усечения состоит в следующем:

- Рассмотренные и оптимизированные концевые поддеревья помечаются.
- Если поддерево помечено, то говорят, что оно отброшено (отсечено).
- Если пометки убираются, то это соответствует "вклеиванию" одного или цепочки поддеревьев.

Пусть для определенности на первом шаге алгоритма выделяются концевые аддитивные поддеревья. Концевое поддерево – это такое поддерево, входами которого являются либо операнды, либо помеченные поддеревья. Концевые поддеревья с аддитивными операциями оптимизируются по схеме сдваивания и помечаются.

На втором шаге ищутся максимальные концевые мультипликативные поддеревья. С ними проводятся те же преобразования.

Начиная с третьего шага, помимо отсечений требуются вклеивания некоторых поддеревьев. Именно, если имеются входы выделяемых концевых поддеревьев, являющихся помеченными поддеревьями того же типа, то их нужно вклеивать. Затем следует оптимизировать полученные поддеревья. Таким образом, концевые поддеревья определяются по усеченным деревьям, а последние строятся в результате преобразований отсечения и вклеивания. Без операции вклеивания обойтись не удается.

Обозначим через $\mathcal{T}_0 = \mathcal{T}_1$, $\mathcal{T}_2$, $\ldots$, $\mathcal{T}_l = \mathcal{T}^*$ набор деревьев, построенных соответственно на шагах $k = 1, 2, \ldots, l$, где номер l находится из условий того, что $l+1$-шаг алгоритма не приводит к модификации структуры.

Программная реализация предложенного метода преобразования символьных формул основана на рекурсии. Это связано с рекурсивной природой данных. Применение рекурсии позволило создать достаточно эффективную программу построения гарантированных оценок и привело к уменьшению вычислительной сложности алгоритмов.

Для нахождения оптимального выражения символьной формулы требуется $O(n^2)$ операций, причем число это достигается на последовательных алгоритмах.

Анализ устойчивости параллельных форм производится для классов возмущений, составленных из элементов пространства параметров и многозначных отображений, действующих на этом пространстве.

3. Реализация параллельных алгоритмов решения дифференциальных уравнений

Многие параллельные методы решения ОДУ основываются на крупнозернистом параллелизме (параллелизме процессов) и соответствующем ему многопроцессорным архитектурам [3]. Особенности гарантированных алгоритмов, использующих символьные формулы, аппроксимирующие оператор сдвига вдоль траектории, наложили определенные ограничения на выбор языка реализации и создание программ.

Для описываемой задачи был выбран стандарт в области систем, ориентированных на обмен сообщениями, – MPI (Message Passing Interface). Использовалось то свойство, что MPI способен учесть различные представлений данных на различных ЭВМ. Декларирующий механизм MPI позволяет описать структуры данных произвольной сложности.

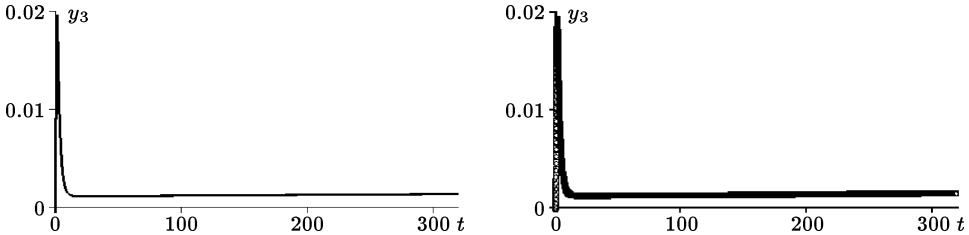
Программные модули реализовывались в среде компилятора Intel C/C++ ver 7.0 noncommercial. По сравнению с реализацией на последовательной ЭВМ время расчетов для мультикомпьютерной среды время расчетов уменьшилось в 6 раз для системы (3), состоящей из восьми нелинейных обыкновенных дифференциальных уравнений.

Пример гарантированного оценивания жестких систем ОДУ с неточно заданными начальными данными. Для проверки результатов выбрана система уравнений, описываюшая химическую реакцию с участием восьми реагентов. Эта система была предложена Шефером [11] для объяснения роста и дифференциации растительной ткани при высоких уровнях светового облучения независимо от фотосинтеза. Готтвальд предложил использовать ее в качестве тестового примера. Соответствующие уравнения имеют вид:

$$\begin{aligned}
\frac{dy_1}{dt} &= -1.71y_1 + 0.43y_2 + 8.32y_3 + 0.0007,\\
\frac{dy_2}{dt} &= 1.71y_1 - 8.75y_2,\\
\frac{dy_3}{dt} &= -10.03y_3 + 0.43y_4 + 0.035y_5,\\
\frac{dy_4}{dt} &= 8.32y_2 + 1.71y_3 - 1.12y_4,\\
\frac{dy_5}{dt} &= -1.745y_5 + 0.43y_6 + 0.45y_7,\\
\frac{dy_6}{dt} &= -280y_6y_8 + 0.69y_4 + 1.71y_5 - 0.43y_6 + 0.69y_7,\\
\frac{dy_7}{dt} &= 280y_6y_8 - 1.81y_7,\\
\frac{dy_8}{dt} &= -y_7
\end{aligned} \tag{3}$$

с начальными данными: $y_1(0) = [1-\delta, 1+\delta]$, $y_2(0) = y_3(0) = y_4(0) = y_5(0) = y_6(0) = y_7(0) = [-\delta, \delta]$, $y_8(0) = [0.0057 - \delta, 0.0057 + \delta]$, $t_{\text{out}} = 321.8122; 421.8122$.

Численные результаты (рисунок) были получены для компьютера с процессором Intel Celeron(R), CPU 1.8 ГГц, 256 Мбайт, а также в мультикомпьютерной



Сравнение экземпляров приближенных решений, начинающихся в граничных точках интервала начальных данных и посчитанных с высокой точностью (а), и проекции гарантированных границ (б) множества точных решений системы (3) на плоскости (t, y_3) в интервале $0 \le t \le 321$: —□— гарантированная верхняя граница y_3; —○— гарантированная нижняя граница y_3

среде. Расчеты проводились для $t \in [0, 321]$, для одного процессора время счета составило 579.5 с, для мультикомпьютерной системы – 98 с.

Список литературы

[1] Воеводин В.В. Математически модели и методы в параллельных процессах. – Москва: Наука, 1986.

[2] Alabdulkarrem M., Lakschmivaran S., Dhall S. Scalability analysis of large code using factorial design // Parallel Computing. – 2001. – Vol. 27. – P. 1937–1947.

[3] Jackson K., Norsett S. The potential for parallelism in Runge–Kutta methods. Part I: RK formulas in standard form // SIAM J. Numer. Anal. – 1995. – Vol. 32. – P. 49–82.

[4] Рогалев А.Н. Исследование практической устойчивости при постоянно действующих возмущениях // Вычислительные технологии (совместный выпуск по материалам Международной конференции ВТММ-2002). – 2002. – Т. 7, № 5. – С. 148–150.

[5] Рогалев А.Н. Задачи практической (интервальной) устойчивости с заданной областью предельных отклонений // Тр. пятой междунар. конф. памяти академика А. П. Ершова. Международное совещание по интервальной математике и методам распространения ограничений. – Новосибирск, 2003. – С. 90–100.

[6] Рогалев А.Н. Гарантированные методы решения систем обыкновенных дифференциальных уравнений на основе преобразования символьных формул // Вычислительные технологии. – 2003. – Т. 8, № 5. – С. 102–116.

[7] Рогалев А.Н. Поведение динамических систем при экстремальных возмущениях // Там же. – С. 68–77.

[8] Рогалев А.Н. Гарантированные оценки безопасного функционирования технических и электроэнергетических систем // Тр. всерос. конф. с междунар. участием "Современные методы математического моделирования природных и антропогенных катастроф". – Красноярск, 2003. – Т. 3. – С. 42–48.

[9] Рогалев А.Н. Включение множеств решений дифференциальных уравнений и гарантированные оценки глобальной ошибки // Вычислительные технологии. – 2003. – Т. 8, № 6. – С. 80–94.

[10] Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985.

[11] Schäfer E. A new approach to explain the "High Irradiance Responses" of photomorphogenesis on the basis of phytochrome // J. of Math. Biology. – 1975. – Vol. 2. – P. 41–56.