УДК 519.62+681.3.06

А. Н. Рогалев

Институт вычислительного моделирования СО РАН
Академгородок, ИВМ СО РАН, Красноярск, 660036, Россия
E-mail: ran@krsk.info

СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ В ГАРАНТИРОВАННЫХ МЕТОДАХ, ВЫПОЛНЕННЫЕ НА НЕСКОЛЬКИХ ПРОЦЕССОРАХ

Описание метода

Методы, строящие гарантированные границы множеств решений систем дифференциальных уравнений с интервальными данными, основаны на символьном представлении формул, аппроксимирующих оператор сдвига вдоль траектории. По найденным символьным формулам вычисляются множества включения (множества приближенных решений), содержащие каждое приближенное решение при варьировании параметров значений, а также области включения глобальных ошибок для всех приближенных решений, соответствующих этим символьным формулам. Завершает алгоритм операция объединения этих множеств включений, реализуемая, например, как сложение множеств. Такой подход позволяет определять границы множеств решений, точно отслеживающие поведение множества всех точных решений, а также устранить влияние так называемого «wrapping» эффекта, проявляющееся практически во всех двусторонних и интервальных методах.

Систематизируем действия, при исполнении которых строятся гарантированные включения следующей задачи:

$$\frac{dy}{dt} = f(t,y),\ \ y(t^0) = y^0 \in Y^0. \tag{1}$$

Здесь величина $Y^0 = \left[\underline{Y^0}, \overline{Y^0}\right]$ представляет интервальный вектор (прямоугольный параллелепипед), хранящий все точные значения начальных значений.

Шаг 1. Начало работы алгоритма – присвоение значений переменным, идентифицирующим систему: размерность, правая часть, начальные данные.

Шаг 2. Реализуется запись символьных формул S-решений как векторных функций с символьными компонентами $S(t^k, Y^0)$, зависящими от символьных начальных данных $Y_1^0, Y_2^0, \ldots, Y_n^0$. Каждая компонента символьного вектора $S_i(t^k, Y^0), i = 1, \ldots, n$ определяется заново в каждой точке t^k как функция, зависящая от символьных стартовых значений $Y_1^0, \ldots, Y_n^0$. При записи символьных формул происходит перемещение от шага к шагу вдоль кривой, заданной символьной формулой приближенного решения, аппроксимирующей траекторию системы (1). Построенная символьная формула является основой для вычисления области значений функций $Y_S(t, Y^0)$ по всем $y^0 \in Y^0$.

Шаг 3. Последовательно исполняется метод хранения и переработки символьной информации при продвижении вдоль траектории решений, производится на основе статичного хранения этой информации, работы с адресацией памяти с помощью функций поточной обработки.

Шаг 4. Символьная формула приближенного решения преобразуется к виду, который позволяет эффективно и быстро вычислять оценки областей значений приближенных решений (S-решения), соответствующие изменениям параметров задачи (1). Для этого используются кусочно-полиномиальное представление символьных формул и опорные функции для многозначных функций, описывающих области значений.

ISSN 1818-7900. Вестник НГУ. Серия: Информационные технологии. 2006. Том 4, выпуск 1

Шаг 5. Символьная формула (вектор с символьными компонентами) Y, аппроксимирующая оператор сдвига вдоль траектории, позволяет определить прообразы экстремальных значений (верхних и нижних границ для интервальных расширений). Эти прообразы принадлежат вектору начальных значений Y^0 для всех узлов сетки t^k. Для нахождения прообразов применяется одна из форм решения экстремальной задачи. Обозначим найденные точки $y_d^0, d = 1, \ldots, 2n$. Они служат основой для определения множеств («брусов») точных решений, включающих все точные решения из окрестности точек y_d^0.

Шаг 6. Для каждого начального значения, являющегося прообразом граничных значений S-решений (Set-решений), всего таких точек $2n$ штук, производится выбор произвольных интервалов, содержащих их и обозначенных $Y_{d,i}^0$, где индекс d принимает значения 1 или 2 в зависимости от того, верхняя или нижняя граница S-решения была выбрана, а индекс i принимает значения 1, 2, …, n, т. е. числа, равные размерности по пространству. Выбор интервала $Y_{Brus,d,i}^0$, в котором начинается «брус» точных решений, производится на основе применения принципа сжимающих отображений в форме алгоритмов типа гомотопии. Конструктивно этот алгоритм заключается в поиске «пестрых» симплексов с изменением известных алгоритмов Ивза и Сайгала [Eaves, Saigal, 1972] и настройке их на случай интервальных или многозначных функций.

Шаг 7. Начиная с полученных областей $Y_{Brus,d,i}^0, d = 1, 2; i = 1, 2, \ldots, n$ производится построение S-решений (расширенных за счет выбора начальных «брусов» и привязанных к экстремальным значениям S-решений). Как упоминалось ранее, эти S-решения впоследствии называются «брусами».

Шаг 8. Определение границы глобальной ошибки начинается с применений алгоритма типа гомотопии. Цель применения алгоритма гомотопии – вычислить области значений (многозначные векторные функции) включающие множества точных решений в областях, содержащих каждую из $2n$-граничных точек. Вычисленные многозначные функции образуют граничные гиперплоскости для множества решений системы ОДУ, являющиеся гарантированными границами этого множества решений. Эти границы областей всех решений могут быть завышенными, но они позволяют оценивать глобальную ошибку решений, являясь включениями остаточных членов разложений.

Шаг 9. Гарантированные границы глобальной ошибки множества приближенных решений устанавливаются двумя способами.

1. Производится суммирование оценок локальной ошибки вдоль траектории, заданной символьным решением, причем эти ошибки привязаны к каждому экстремальному значению S-решения. Формула локальной ошибки имеет следующий вид:

$$\rho_i^{n+k} = \frac{h^{p+1}}{(p+1)!}\left(\sum_{i=1}^{k} i^{p+1}\alpha_i - (s+1)\sum_{i=1}^{k} i^p \beta_i\right) y_i^{(p+1)}(t^n + \xi),$$

где α_i, β_i – коэффициенты метода, используемого при аппроксимации точного решения [Рогалев, 2003], $i = 1, \ldots, n$ – индексы компонент точного решения.

Выражение $Y_i^{(p+1)}(t^n + \xi)$ – это символьная формула производной i-компоненты точного решения, $t^n + \xi$ – некоторая промежуточная точка интервала $\left[t^n, t^{n+1}\right]$ На основе этой формулы определяется область значений (многозначное расширение) производной. При этом вычисления для той величины интервала, которая найдена для каждого из прообразов Y_{low}^n, Y_{up}^n на основе принципа неподвижной точки.

Верхняя оценка глобальной ошибки (векторная величина, состоящая из $2n$-компонентов) – это результат накопления величин локальных ошибок вдоль траектории решения. Подобная верхняя оценка насчитывается для всех подынтервалов на всем интервале интегрирования и для каждого экстремального значения, ограничивающего S-решение. Именно так происходит независимое определение символьных формул решений и процесс суммирования локальных ошибок вдоль траекторий решения.

2. Вычисление оценки погрешностей приближенных решений основано на сравнении решений двух систем дифференциальных уравнений: исходной системы ОДУ и системы ОДУ, точным решением которой будет приближенное решение, с символьной формулой, полученной ранее.

Шаг 10. К границам *S*-решений добавляется оценка глобальной ошибки (таких операций объединения будет 2*n*)

$$Y(t, Y^0) = val\,(S(t, Y^0) \cup R(t, Y^0)),$$

где обозначение *val* соответствует нахождению числовых значений областей решений по символьным формулам

В итоге, величина $Y(t, Y^0)$ – это гарантированная оценка множества точных решений с учетом всех видов ошибок.

Преобразования символьных формул

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 (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, используются формулы, хорошо приближающие точное решение, и выполняется алгоритм преобразования этих формул. Следует также учесть, что требование устойчивости разностных методов как непрерывной зависимости их решений от возмущений входных данных трансформируется в случае реализации гарантированных методов.

Пусть H_i – последовательность пространств; F^i, $i = 1, 2, \ldots, n-1$ – 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

$$\begin{aligned}
&Y^1 = F^1(t^0, t^1, Y^0, Y^1) = S^1(Y^0),\\
&Y^2 = F^2(t^0, t^1, t^2, Y^0, Y^1, Y^2) = S^2(Y^0) \circ S^1(Y^0),\\
&\ldots\\
&Y^i = F^i(t^0, \ldots, t^i, Y^0, Y^1, \ldots, Y^i) = S^i(Y^0) \circ \ldots \circ S^2(Y^{i-1}) \circ S^1(Y^i)\\
&\ldots\\
&Y^m = F^m(t^0, \ldots, t^m, Y^0, Y^1, \ldots, Y^m) = S^m(Y^0) \circ \ldots \circ S^2(Y^{i-1}) \circ S^1(Y^i)
\end{aligned} \qquad (2)$$

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

Очевидно, что для большинства методов $H_i = R^i$, $i = 1, 2, \ldots, m-1$ и $H_m = R^m$, где R^m есть *m*-мерное евклидово пространство, и F_i – это формула отображения, основанного на элементарных арифметических операциях.

Выполнение преобразований над символьными формулами является первым этапом построения гарантированных границ решений задачи, после которого нужно провести числовые расчеты. Некоторые системы компьютерной алгебры позволяют (исходя из построенных формул) генерировать программы на подходящем для таких расчетов языке программирования. Однако получаемые формулы, как правило, являются громоздкими, и непосредственные вычисления с их помощью неэкономны, особенно, если речь идет о вычислениях в цикле. Кроме того, предоставляемая системами компьютерной алгебры возможность проводить вычисления с числами большой разрядности делает арифметические операции дорогими, так что проблема экономии вычислений стоит в этом случае еще более остро. Алгоритм исполнения гарантированных методов особенно просто выглядит в случае применения символьных формул явных разностных схем. Используя последовательные подстановки и приведение подобных членов, формулу (2) можно преобразовать в выражение, зависящее только от Y^0.

При этом в большинстве случаев крайне высоки затраты памяти для хранения получающихся символьных формул.

Поэтому в общем случае для описываемого класса методов 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 [Вирт, 1985].

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 (2) осуществляется в процессе обхода этого дерева начиная с вершин.

В обозначенном выше алгоритме получения символьных формул

$$Y^n = F^n(t^0, \ldots, t^n, Y^0, Y^1, \ldots, Y^n) = S^n(Y^{i-1}) \circ \ldots \circ S^2(Y^{i-1}) \circ S^1(Y^i)$$

используется следующая методика обработки последовательности символьных формул. Пусть $\phi(y^0)$ – это однозначное отображение единичного интервала из R^1 на гиперкуб из R^n которое каждой точке $t \in R$ сопоставляет некоторую точку $y = \phi(t)$. При помощи такого отображения можно построить алгоритм исполнения, который для каждой точки $t \in R$ позволяет определить формулу отображения $F(Y_1^0, Y_2^0, \ldots, Y_n^0)$ 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 пространств. 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, можно лишь дать последовательность кривых [Вирт, 1985], 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-приближение можно рассматривать как некоторую аппроксимацию m-функции в рекурсивной формуле (2). Это соответствие задано отображением элементов конечного множества отрезков из единичного интервала и элементами конечного множества гиперкубов, входящих в R^n.

Такое соответствие строится следующим образом. Гиперкуб разбивается на 2^n гиперкубов, называемых квантами первого разбиения. Длина ребра каждого такого кванта равна ½, кванты нумеруются индексом i_1 от $2^n - 1$ так, чтобы номера квантов, имеющих общую грань, отличались на 1. Обозначать кванты первого разбиения будем $r(i_1), i_1 = 2^n - 1$.

Каждый квант первого разбиения разбивается таким же образом, как гиперкуб в R^n, на кванты второго разбиения с длиной ребра ¼, которые нумеруются по тому же закону, что и кванты первого разбиения. При этом нулевой квант второго разбиения, входящий в r_{i_1}, должен иметь общую грань с $(2^n - 1)$-квантом второго разбиения, входящим в $r(i_1 - 1)$. Кванты второго разбиения обозначим через $r(i_1, i_2)$, где i_2 – номер кванта второго разбиения, входящего в квант $r(i_1)$.

Аналогично получаем кванты любого разбиения с номером $r(i_1, i_2, \ldots, i_m)$, $0 \le i_j \le 2^n$. Способ соединения между собой квантов, номера которых отличаются на 1 (в лексикографическом порядке), лежит в основе алгоритма, связывающего адреса, по которым хранятся компоненты символьной формулы (2).

Совокупность гиперкубов в пространстве R^n, рекурсивно квантованных m-раз, носит название дискретного пространства m-го разбиения и обозначается G_m^n. Если в каждом разбиении число частей, на которые делится квант предыдущего разбиения выбрать равным 2^n, то при любом m количество квантов в G_m^1 и G_m^n совпадает. Кванты m-го разбиения G_m^1, полученные из одного кванта $(m-1)$-го разбиения, нумеруются слева направо индексом i_m, $0 \le i_m \le 2^n$ и обозначаются $q(i_1, i_2, \ldots, i_m)$. Сопоставив кванты G_m^1 и G_m^n с одинаковыми наборами индексов $i_1, i_2, \ldots, i_m$, получаем взаимно-однозначное отображение $\phi_m : R_m^1 \to R_m^n$ или для

отдельных квантов $\phi_m(q(i_1, i_2, \ldots, i_m))$. Отображение ϕ_m можно также интерпретировать как соответствие между равноотстоящими точками из R^1 (центрами квантов m-разбиения) и узлами равномерной прямоугольной решетки в R^n, где каждый узел также является центром кванта m-разбиения. Итогом работы описанного выше алгоритма будет возможность в любой точке t^k построить символьную формулу решения (2) и вычислять на основе этой формулы значения решений.

Оптимизация алгоритма за счет распараллеливания символьных формул

Здесь мы рассмотрим еще одну модель построения символьных формул метода сдвига вдоль траектории. Отметим, что в этой модели используются способы представления древовидных структур данных с помощью функций поточной обработки [Burge, 1975], которая материализует составляющие части структур лишь при их необходимости.

Пусть $Y^0 = (Y_1^0, Y_2^0, \ldots, Y_n^0)$ – набор некоторых символьных переменных, а целочисленная переменная i – параметр цикла, изменяющийся от m до n. Рассмотрим функции, которые задаются арифметическими выражениями, составленными из имен символьных переменных, чисел, знаков арифметических операций $\times, /, +, -$ и операции возведения в степень. Возьмем набор таких функций, $\Phi_0(t, Y_1^0, Y_2^0, \ldots, Y_n^0), \ldots, \Phi_k(t, Y_1^0, Y_2^0, \ldots, Y_n^0)$ и набор знаков операций $\oplus_1, \oplus_2, \ldots, \oplus_k$, в который включены операции сложения, умножения или возведения в степень. С помощью этих двух наборов можно составить системы символьных формул $F_0(t, i), F_1(t, i), \ldots, F_k(t, i)$ вида

$$F_0(t, i) = \begin{cases} \oplus_0(t, Y_1^0, Y_2^0, \ldots, Y_n^0), & \text{если } i = m, \\ F_0(t, i-1) \oplus_i F_1(t, i), & \text{если } i > m, \end{cases}$$

$$F_{k-1}(t, i) = \begin{cases} \oplus_{k-1}(t, Y_1^0, Y_2^0, \ldots, Y_n^0), & \text{если } i = m + k - 1, \\ F_{k-1}(t, i-1) \oplus_i F_k(t, i), & \text{если } i > m + k - 1. \end{cases} \quad (3)$$

Система символьных уравнений (3), определяющая $F_0(t, i)$, может быть изображена линейно:

$$\left\{m, \otimes_1, \oplus_0, \left\{m+1, \otimes_2, \oplus_1, \left\{\ldots, \left\{m + k - 1, \otimes_k, \oplus_{k-1}\right\}\ldots\right\}\right\}\right\}.$$

Интересно отметить, что высота параллельной формы уменьшается при увеличении количества арифметических операций.

Справедлива лемма. Рассмотрим $A = BE + BG$, где $B, E, G,$ – подвыражения, алгоритмы вычисления которых имеют соответственно высоты h_B, h_E, h_G. Вынесение общего множителя за скобку $A' = B(E + G)$ приводит к уменьшению высоты параллельной формы на единицу, т. е. $h_{A'} = h_A - 1$ тогда и только тогда, когда $\max\{h_E, h_G\} < h_B$.

Доказательство этой леммы практически очевидно, аналогичные результаты приводятся в [Brent et al., 1973; Воеводин, 1991]. Но она не отвечает на вопрос, когда именно нужно раскрывать скобки. Кроме того, ее нужно применять к предварительно оптимизированным подвыражениям сомножителям, т. е. рассматривать пары $A = BE + BG$ и $A = B(E + G)$.

Тогда, вообще говоря, не исключена возможность того, что, раскрывая скобки, мы сначала «потеряем» в смысле высоты параллельной формы, зато потом «выиграем», если полученные выражения вновь улучшим. При этом уменьшить высоту параллельной формы алгоритма за счет вынесения множителя можно только путем уменьшения или сохранения уровней подвыражений.

Предложенное описание является способом представления древовидной структуры с помощью формулы с вложенными скобками [Knut, 1998]. Известно, что любая символьная формула алгебраического выражения имеет древовидную структуру. Непосредственный перевод этих выражений в программы для вычисления этих выражений может быть осуществлен с помощью обратной польской записи и компиляции комбинаций. Для применяемого класса гарантированных алгоритмов решения систем ОДУ этот подход требует больших затрат машинной памяти и приводит к сильному увеличению времени исполнения реализаций алгоритмов. Описанный в [Burge, 1975] способ представления древовидных структур с помощью функций поточной обработки в комбинации с другими методами позволяет актуализировать

составляющие части структур лишь при их необходимости, в нашем случае в заключительной части алгоритма.

Назовем целое число k-глубиной системы символьных уравнений (3), а саму систему символьных уравнений (3) сгенерированной формулой символьного метода, основанного на аппроксимации сдвига вдоль траектории. При этом в символьной формуле можно выделить k-подсистем системы (3), соответствующие набору индексов $0 \le r \le k$. Системы символьных уравнений вида (3), в которых все $\oplus_j = +$ (все $\oplus_j = \times$ или $\oplus_j = -$) при $j = 1, 2, \ldots, k$ назовем простейшими символьными системами, в противном случае, когда в наборе операций встречаются разные знаки из $\{+, \times, -\}$, назовем смешанными системами. Для простейших систем можно употреблять сокращенную линейную запись $\{m, \otimes, \oplus_1, \oplus_2, \ldots, \oplus_k\}$. Такое сокращение линейной записи можно распространить и на простейшие системы смешанного типа. Отметим, что если в системе (3) будет $\otimes_p = \otimes_{p+1} = \ldots = \otimes_q$, $0 \le p \le q \le k$, то фрагмент записи

$$\ldots, \{m + p - 1, \otimes_p, \oplus_{p-1}, \{\ldots, \{m + q - 1, \otimes_q, \oplus_{q-1}, \{\ldots, \{\ldots,$$

может быть заменен на $\ldots, \{m + p - 1, \otimes_p, \otimes_{p-1}, \otimes_{p-2}, \ldots, \otimes_{q-1}, \{\ldots,$ без потери информации о виде системы.

Считая неизменной структуру символьной формулы, будем при реализации вычисления области значений этого выражения придерживаться следующего.

Значение формулы зависит только от значений составляющих его подформул и не зависит от других его свойств: приводимые формулы имеют одно и то же значение; соответствующие процедуры вычисления значений формул строятся из простейших элементов.

Если одна функция обрабатывает символьную формулу в естественном порядке, а другая обрабатывает элементы формулы в том же порядке, то в таких случаях нет необходимости получать весь список перед применением к нему второй функции. Две функции могут быть скомбинированы таким образом, что на любой стадии вторая функция будет выдавать требования для следующего элемента, которая затем создается с помощью первой функции. Тем самым формирование следующего элемента символьной системы откладывается до тех пор, пока он действительно не будет необходим.

Более экономично по памяти использовать комбинацию двух функций, в которой только один элемент символьной системы появляется как промежуточный результат. Функция, которая вызывается для получения следующего элемента символьной системы, должна создавать его и вновь устанавливать себя таким образом, чтобы подготовиться к получению остатка, или хвоста, списка при очередном обращении. Вычислительные процедуры для нахождения областей значений символьных формул строятся на основе преобразования формулы в последовательность инструкций, значение выражения будет найдено после того, как процедура будет исполнена.

Гарантированные границы решений ОДУ с неточно заданными начальными данными

Программы нахождения гарантированных границ решений ОДУ реализовывались в среде компилятора Intel C/C++ ver 7.0 noncommercial. По сравнению с реализацией на последовательной ЭВМ время расчетов для мультикомпьютерной среды уменьшилось в 6–8 раз для систем (4), (5).

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

Границы областей точных решений систем обыкновенных дифференциальных уравнений строились для систем, у которых параметры исходной задачи были заданы как интервалы. Числовые значения параметров этих задач приведены в надписях к графикам множеств решений и границ этих множеств.

1. Обыкновенное дифференциальное уравнение второго порядка Ван-дер-Поля, записанное в виде системы двух уравнений

$$\begin{aligned} \frac{dy_1}{dt} &= y_2, \\ \frac{dy_2}{dt} &= y_1 - \mu(1 - y_1^2)y_2, \end{aligned} \tag{4}$$

где $\mu = 2$ – параметр. Хорошо известно, что начало координат асимптотически устойчиво при $\mu > 0$ и область притяжения ограничена неустойчивым предельным циклом.

2. Система нелинейных обыкновенных дифференциальных уравнений

$$\begin{aligned} \frac{dy_1}{dt} &= -2y_1 + 4y_2y_3 + 4y_4y_5, \\ \frac{dy_2}{dt} &= -9y_2 + 3y_1y_3, \\ \frac{dy_3}{dt} &= -5y_3 - 7y_1y_2 + 25, \\ \frac{dy_4}{dt} &= -5y_4 - y_1y_4, \\ \frac{dy_5}{dt} &= -y_5 - 3y_1y_4 \end{aligned} \tag{5}$$

получена при разложении системы уравнений Навье-Стокса в ряд по ортогональной системе базисных функций и удержании 5 членов этого разложения.

На плоскостях y_1, y_2, y_3, y_4 и других фазовых плоскостях графики представляют из себя изображения взаимозависимостей между переменными. В некотором смысле графики являются аналогами графиков в полярных системах координат, строят значения относительно начальной точки.

Список литературы

Вирт Н. Алгоритмы + структуры данных = программы. М.: Мир, 1985.

Воеводин В. В. Математические основы параллельных вычислений. М.: Изд-во Моск. гос. ун-та, 1991.

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

Burge W. Recursive programming techniques. Massachusets; L.: Addison-Wesley publishing company, 1975.

Brent R. P., Kuck D. J., Maruyama K. The parallel evaluation of arithmetic expressions without division. IEEE Transactions on Computers, 1973. C. 22.

Eaves R. C., Saigal R. Homotopies for computation of fixed points on unbounded regions // Mathematical Programming. 1972. Vol. 3, № 2. P. 225–237.

Knut D. The art of computer programming. Vol. 1: Fundamental algorithms. Massachusets; Berkley; Sydney: Addison-Wesley Longman Inc., 1998.

Материал поступил в редколлегию 18.09.2006