О РЕАЛИЗАЦИИ НА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ АЛГЕБРАИЧЕСКО-ДИФФЕРЕНЦИАЛЬНЫХ АЛГОРИФМОВ*

ВВЕДЕНИЕ

Основной областью применения быстродействующих электронных вычислительных машин является осуществление арифметических алгорифмов, т.е. проведение определенных последовательностей арифметических операций над числами с необходимыми логическими разветвлениями. Появление электронных вычислительных машин как раз и связано с большим объемом вычислительной работы во многих арифметических алгорифмах.

Однако в математике большую роль играют и неарифметические алгорифмы, в частности алгорифмы, связанные с выкладками в буквах. Научная математическая работа во многих случаях требует проведения вычислений по таким алгорифмам, но проведение этих вычислений самим математиком иногда бывает очень утомительным и зачастую представляется вообще невозможным из-за большого объема вычислительной работы. Естественно поэтому поставить вопрос об осуществлении и таких алгорифмов на электронных вычислительных машинах, тем более, что многие из этих алгорифмов требуют проведения (вообще говоря многократного) небольшого количества различных типов операций.

Примером такого алгорифма является алгорифм Картана (см. [1]). Он распадается на следующие сравнительно простые математические операции, проводимые в буквах:

1) алгебраические операции над функциями от нескольких переменных: сложение, вычитание, умножение;

2) подстановка функции в функцию вместо переменной (суперпозиция функций)[1];

3) дифференцирование функций от нескольких переменных по ее аргументам.

Алгорифм Картана первоначально возник как аналитический аппарат исследования поверхностей, обладающих различными геометрическими свойствами. В настоящее время он находит применение в другой области математики — в теории аналитических (точных) решений уравнений в частных производных. Если рассматривается система уравнений

$$\sum_{j=1}^{m} \sum_{k=1}^{n} a_{ijk} \frac{\partial u_j}{\partial x_k} = F, \quad i = 1, 2, \ldots m, \tag{1}$$

где a_{iik} . F_i есть функции от u_1, ... u_m , x_1, ..., x_n, то наиболее общий метод получения точных решений системы (1) заключается в присоединении к (1) дополнительного Уравнения

$$f = 0, \tag{2}$$

(*f* - функция от $x_1, \ldots, x_n, u_1, \ldots, u_m$ и от частных производных функций u_j по x_k порядка от первого до *p*-го включительно) и рассмотрении условий совместности получающейся при этом системы.

Накладывая различные требования на произвол решения системы (1) и (2), можно выделить различные классы решений системы (1).

Ряд работ, в которых находятся аналитические решения уравнений газовой динамики, ограничивается случаем $p = 0$, т.е. в качестве (2) берется уравнение

$$f(x_1, \ldots, x_n, u_1, \ldots, u_m = 0$$

хотя и при этом часто получаются громоздкие выкладки (см. [2 - 4]). Случаи же, когда $p > O$, зачастую требуют от математики непосильной вычислительной работы. Поэтому применение электронных вычислительных машин для реализации алгебраически-диф-ференциального алгоритма Картана является весьма желательным.

Кроме того, машинизация математических алгорифмов должна найти применение во многих других областях математики (тензорный анализ, линейная алгебра, алгебра форм, алгорифмы аналитического интегрирования, разложение в ряды и т.п.).

Предлагаемая алгебраически-дифференциальная программа АДП позволяет решать довольно широкий класс задач по определению совместности систем дифференциальных уравнений в частных производных, а также многие другие задачи на электронной вычислительной машине "Стрела" (описание и систему команд этой машины можно найти, например, в [5]).

---

*Проблемы кибернетики. 1961. Вып. 6. (Соавтор В.А. Шурыгин.)

[1]Операции, представленные в п. 1, можно осуществлять также подстановкой заданных функций в функции вида $x + y, x - y, xy$. Поэтому эти операции можно считать частными случаями подстановок.

1. ОПЕРАЦИИ, ОСУЩЕСТВЛЯЕМЫЕ АДП

Описываемая алгебраическо-дифференциальная программа является одной из первых в СССР программ, осуществляющих буквенные выкладки. Поэтому при ее создании естественно было ограничиться (как для первых опытов) буквенными выражениями как можно более простого вида, но в то же время такими, чтобы круг задач, решаемых с помощью АДП, не был бы слишком узок и программа имела бы хоть какую-то практическую ценность.

АДП преобразует выражение вида

$$P = \sum \pm a_i x_1^{\alpha_{1i}} x_2^{\alpha_{2i}} \ldots x_n^{\alpha_{ni}} \qquad (3)$$

где a_i, a_{ki} - любые числа, каждое из которых может быть записано в одной ячейке памяти,
x_k - буквы, причем если они обозначают переменные величины, то одни из них могут быть функциями других.

Выражения вида (3) мы будем называть полиномиалами.

Полиномиалы имеют очень простой вид, однако путем введения соответствующих обозначений любое буквенное выражение, встречающееся в математической практике, можно представить в виде полиномиалов. Например, выражение

$$a \sin x + 1\text{n} (\cos y)$$

превращается в полиномиал, если обозначить *sin x* через *u,* а 1n (cos *y*) – через *u*. Выражение

$$R = \frac{ax + \sqrt{cy^2 + 1}}{dx^2 + ey^2}$$

можно записать как полиномиал

$$R = axu^{-1} + v^{1/2} u^{-1},$$

где *u, v* - функции от *x* и *y,* имеющие вид полиномиалов

$$u = dx^2 + ey^2, \quad v = cy^2 + 1.$$

Для ввода в машину полиномиалы кодируются определенным образом числами (см. разд. 2), кроме того, каждый полиномиал имеет свой номер - какое-нибудь число в восьмеричной системе от 4000 до 6777, которое можно рассматривать как числовой код буквы *P* в (3).

Последовательность операций, которую нужно выполнить над полиномиалами, задается в виде так называемой псевдопрограммы, по виду напоминающей обычную программу из машинных команд, причем в ней указываются только номера поли-номиалов, но не сами полиномиалы. Таким образом, одна и та же псевдопрограмма может быть иногда использована для решения фактически разных задач.

АДП выполняет две основные операции над полиномиалами.

1. *Подстановка полиномиала в полиномиал вместо буквы*. Пусть полиномиал *P* содержит букву *q,* причем только в целых неотрицательных степенях, а полиномиал *Q* букву *q* не содержит. Программа может подставить *Q* в *P* вместо буквы *q.*

Для более удобного задания операции подстановки сделано следующее. Номер подставляемого полиномиала *Q* должен совпадать с кодом буквы *q* (о кодировании полиномиалов речь пойдет в разд. 2), если нужно вместо буквы q_1 подставить поли-номиал Q_1, вместо q_2 - полиномиал Q_2 и т.д. и вместо q_n - полиномиал Q_n, и коды букв q_i, - или, что то же самое, номера полиномиалов Q_i идут подряд, т.е. код q_{i+1} равен коду q_t плюс единица, то достаточно задать номер полиномиала *P,* коды Q_1 и Q_n (см. таблицу).

При этом не обязательно, чтобы полиномиал *P* содержал все буквы от q_1 до q_n (в частности, он может их совсем не содержать), и не обязательно, чтобы для каждой буквы q_i от q_1 до q_n, входящей в полиномиал *P,* имелся бы соответствующий поли-номиал Q_i (в частности, таких полиномиалов может не быть ни для одной буквы из $q_1, \ldots, q_n$.

Подстановкой можно осуществлять алгебраические операции сложения, вычитания и умножения (см. сноску[1]).

Операция подстановки сделана двух типов: с приведением подобных членов у результата и без приведения. Последняя введена для экономии времени, так как приведение подобных членов в длинных полиномиалах проходит долго и обычно в промежуточных выкладках не требуется.

Исходный полиномиал *P*, в который производилась подстановка, в памяти машины по окончании подстановки не остается. Результат подстановки - новый полиномиал -имеет тот же номер *p,*

2. *Дифференцирование полиномиалов.* Пусть полиномиал *R* зависит от величин $x_1, \ldots, x_n, t_1, \ldots, t_m$, причем x_i есть дифференцируемые функции от переменных t_k. Задается таблица производных

$$\partial x/\partial t_k = y_{ik}, \quad i = 1, \ldots, n; \quad k = 1, \ldots, m, \qquad (4)$$

с учетом которой АДП может вычислить полиномиал

$$\frac{\partial R}{\partial t_k} = \sum_{i=1}^{n} \frac{\partial R}{\partial x_i} y_{ik} + \frac{\partial * R}{\partial t_k}, \qquad (5)$$

расписанный, конечно, в виде (3).

Здесь слева мы написали $\partial R/\partial t_k$, так как это при $m > 1$, вообще говоря, является частной производной с учетом зависимостей (4) при фиксированных всех остальных t_j ($j \neq k$}. Справа $\partial^* R/\partial t_k$ - частная производная R по t_k , входящей в R явно, без учета зависимостей (4).

Описанных операций подстановки и дифференцирования достаточно для решения очень многих задач, в частности во многих случаях для реализации алгорифма Картана.

АДП проводит над полиномиалами также операции вспомогательного характера.

1.Присвоение полиномиалам с номерами P, $P + 1$, ..., $P + n$, других номеров Q, $Q + 1$,..., $Q + n$ соответственно.

2. Дублирование полиномиалов с номерами P, $P + 1$, ..., $P + n$, причем новым полиномиалам присваиваются заданные номера Q , $Q + 1$,..., $Q + n$ соответственно.

3. Ввод полиномиалов с номерами P, $P + 1$,..., $P + n$. Перфокарты с набитыми на них полиномиалами ставятся на читающее устройство машины, причем перед каждым полиномиалом помещается перфокарта, в первой строке которой в разрядах второго адреса команд указывается число строк (ячеек памяти), занимаемых полиномиалом, уменьшенное на единицу, в восьмеричной системе. Эта перфокарта называется справкой данного полиномиала.

Таблица

Операции	Псевдокоманда				
	I A	II A	III A	Контрольный сигнал	Код операции
Подстановка с приведением подобных членов	P	Q_1	Q_n	1	05
Подстановка без приведения подобных членов	P	Q_1	Q_n	1	04
Дифференцирование	R	t_k	S	1	06
Присвоение других номеров	P	n	Q	1	10
Дублирование	P	n	Q	1	07
Ввод	0000	n	P	1	01
Вывод	P	n	0000	1	02
Стирание	P	n	0000	1	03

4. Вывод полиномиалов с номерами P, $P + 1$, ..., $P + n$. Перед выдачей каждого из этих полиномиалов выдается его справка-.

5. Стирание нолиномиалов с номерами P, $P + 1$, ..., $P + n$. После осуществления этой операции место, занимаемое этими полиномиалами, считается свободными и может быть занято другими полиномиалами.

Во всех этих операциях n может быть равно нулю (когда операция проводится над одним полиномиалом).

Задаются эти операции в виде строк - псевдокоманд (см. таблицу).

Обозначения в этой таблице те же, что в описании операций выше. Вид операции задается в тех же разрядах, что и код операции в обычных машинных командах, причем в 36-м разряде должна стоять единица.

Псевдокоманды располагаются в памяти машины в том порядке, в каком они должны исполняться, причем они могут чередоваться и с обычными машинными командами, исключая только команды групповых операций, задаваемые в две строки. Машинные команды, стоящие в псевдопрограмме, мы также будем называть псевдокомандами. Они в 36-м разряде должны иметь 0.

Команда условного перехода первого рода (код операции 20), стоящая в псевдопрограмме, передает управление только на псевдокоманды, при этом нуль по третьему адресу не заносится. Команда условного перехода второго рода (код операции 27) передает управление только на подпрограммы из машинных команд, причем эти подпрограммы могут содержать команды групповых операций.

2. КОДИРОВАНИЕ ПОЛИНОМИАЛОВ ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ ПРОГРАММЫ

Ячейка памяти машины "Стрела" состоит из 43 разрядов, имеющих номера с 0 по 42. Назовем группу разрядов с 0 по 11-й первой клеткой данной ячейки, группу разрядов с 12 по 23-й - второй клеткой, с 24 по 35-й - третьей клеткой данной ячейки.

Рассмотрим, как кодируются полиномиалы (все указываемые при этом числа -в восьмеричной системе).

Знак плюс кодируется числом 7001, знак минус - числом 7003, буквы - любыми числами от 4000 до 7000 включительно. Численные коэффициенты и показатели степени записываются в нормализованном виде в двоичной системе в специально отведенных для них ячейках, а в строке 3 кодируются номером соответствующей ячейки, отсчиты-вая от первой из ячеек, занимаемых полиномиалом, считая ее за нулевую.

В первую клетку первой из ячеек, занимаемых полиномиалом, записывается номер полиномиала, а в 36-й разряд этой ячейки ставится единица. Со второй клетки этой ячейки начинается запись самого полиномиала. Кодами символов, из которых состоит эта запись, заполняется вторая и третья клетки первой ячейки, затем первая, вторая и третья клетки следующей ячейки и т.д. Коды численных коэффициентов записывают-ся в начале одночленов, коды показателей степеней - сразу после тех букв, к которым они относятся.

Буквы в одночленах записываются в порядке возрастания их кодов. За счет этого ограничения получается большой выигрыш в скорости работы программы.

Знак "+" в начале полиномиала не опускается. Буквы с нулевыми показателями степени и коды числа 1 не пишутся (кроме случая, когда весь одночлен равен 1).

В конце полиномиала ставится знак 7002.

Числа, относящиеся к полиномиалу, обычно записываются сразу же после ячеек с закодированным полиномиалом, хотя возможны и другие случаи расположения.

Пример. Полиномиал

$$P = 2x_1x_2^2 - 1 + x_2^{1/2}x_3$$

полиномиала равным 5000, следующим образом:

Адрес ячейки	I клетка	II клетка	III клетка	36 разряд	37-42 разряды
k	5001	7001	0005	1	00
$k+1$	4001	4002	0005	0	00
$k+2$	7003	0006	7001	0	00
$k+3$	4002	0007	4003	0	00
$k+4$	7002	0000	0000	0	00
$k+5$	2000	0000	0000	0	00
$k+6$	2000	0000	0000	0	00
$k+7$	2000	0000	0000	0	00

Последние три строки таблица - числа 2, 1 и 0,5 в двоичной системе в нормализованном виде. Полиномиал, тождественно равный нулю, кодируется так:

P 7002 0000 1 00

где P — номер полиномиала.

При выборе способа кодирования полиномиалов приходится считаться с имеющимися в наличии устройствами для набивки перфокарт и печати с перфокарт на бумагу, так как эти устройства объединяют двоичные цифры в группы по три или по четыре, образуя восьмеричные или десятиричные цифры, которые, в свою очередь, разбиваются на отдельные группы - мантиссу и порядок в числах, адреса и код операции в командах. Выбранный способ кодирования полиномиалов хотя и не очень экономичен с точки зрения использования разрядов ячеек памяти, однако удобен при набивке перфокарт и печати с перфокарт на бумагу. Как показывает опыт, при некотором навыке кодировать и читать полиномиалы нетрудно, и ошибок при этом не бывает, однако на повестку дня необходимо поставить вопрос о создании устройств, позволяющих набивать на перфокарты информацию, заданную непосредственно для печати буквенных выражений с перфокарт, на которых они набиты в виде численных кодов. Иначе говоря, требуется автоматизировать процесс кодирования буквенных выражений, что имеет большое значение также для автоматизации программирования.

Можно кодировать полиномиалы и другими, более удобными способами, и специальными программами переводить эти коды к описанному здесь виду.

То, что на каждое число отводится целиком ячейка памяти, позволяет использовать числа в очень широких пределах с большим числом значащих цифр.

Таблица обозначений производных (4) кодируется так:

I клетка	II клетка	III клетка	36 разряд	37-42 разряды
7000	0000	0000	1	00
x_1	$t1$	$y_{1\,1}$	0	00
x_2	$t2$	$y_{1\,2}$	0	00
x_3	$t3$	$y_{1\,3}$	0	00
	и так далее		0	00

Порядок строк произвольный, только в конце должна стоять строка из нулей. Эта таблица находится в памяти машины на правах полиномиала, имеет номер 7000, ее можно вводить в машину и выводить с помощью псевдокоманд ввода и вывода как полиномиал с номером 7000, а также стирать, присваивать ей другой номер и дублиро-вать ее как обычный полиномиал, только к моменту исполнения операции дифферен-цирования она должна иметь номер 7000. Это позволяет иметь несколько различных таблиц и использовать каждый раз ту, которую нужно, например, в задачах, где требует-ся вычислять то полные, то частные производные зависимость каких-нибудь перемен-ных от других то учитывается, то не учитывается.

Для обработки полиномиалов служит специальный блок программы - блок просмотра. Чтобы описать его работу, представим себе, что клетки, в которых записан полиномиал, расположены в одну линию или нанесены на узкую ленту в один ряд. Пусть по этой ленте может двигаться считывающая и записывающая головка, которая может делать следующие операции:

1) сдвинуться на одну клетку вправо и выдать символ, записанный в клетке, в которую сдвинулась головка, в 1 клетку какой-нибудь стандартной ячейки памяти машины;

2) та же операция, но с движением влево;

3) записать в клетку, в которой находится головка, новый символ, заданный в 1 клетке стандартной ячейки;

4) в любой момент времени можно запомнить, в какой клетке какой ячейки находится головка, т.е. записать в какую-нибудь ячейку памяти мащины адрес ячейки и номер клетки, где находится головка в данный момент;

5) в любой момент времени можно вернуть головку в клетку, которая была отмечена по предыдущей операции.

Кроме этой головки пусть имеется вторая головка, которая может двигаться по ленте только вправо и только считывать символы, но не записывать новые, и третья головка, которая может только записывать символы и после каждой записи автомати-чески сдвигаться на одну клетку вправо.

Если в клетках ленты записаны полиномиалы, то с помощью этих головок можно над ними проделывать различные операции. При этом удобнее иметь именно три головки указанного типа.

При подстановке полиномиала в полиномиал первая головка двигается по полинсмиалу, в который производится подстановка, вторая - по подставляемому поли-номиалу, а третья головка, получая от первых двух необходимую информацию, записывает на свободном месте в памяти машины результат подстановки — новый полиномиал. При приведении подобных членов первая и вторая головки двигаются по различным одночленам полиномиала. Первая головка при этом стирает ненужные одночлены и записывает коды новых численных коэффициентов.

Таким образом, операции над полиномиалами можно разбить на элементарные операции, осуществляемые головками, и на операции сравнения символов, считываемых головками, между собой и с определенными эталонами. Результаты сравнения определяют последовательность этих элементарных операций.

Блок просмотра имитирует работу описанных головок. Каждая операция, осуществ-ляемая головкой, задается одной неизменной командой. В большей своей части АДП состоит из этих команд, команд сравнения и команд условных и безусловных переходов,

3. РАСПРЕДЕЛЕНИЕ ПАМЯТИ. БЛОК - СХЕМА ПРОГРАММЫ

Память машины распределена следующим образом

АДП

Ячейки с 1 по 714 (включая рабочие ячейки) Псевдопрограмма

Полиномиалы

Свободное место под новые полиномиалы

Полиномиалы располагаются в памяти машины вплотную один к другому, без пропусков. Новые полиномиалы, вырабатываемые АДП, записываются на свободное место вплотную к уже имеющимся.

В начале работы в ячейках 15, 16, 17 должна быть записана следующая информация:
в 1 клетке ячейки 15 - адрес той псевдопрограммы, с которой надо начать работу,
в 1 клетке ячейки 16 - адрес ячейки, с которой начинаются полиномиалы, в 1 клетке ячейки 17 - адрес ячейки, с которой начинается свободное место под полиномиалы.

В процессе работы программы информация в ячейках 15 и 17 автоматически корректируется, причем в ячейке 15 указывается адрес псевдокоманды, которая будет испол-няться следующей.

По функциональным признакам АДП можно разбить на следующие блоки.

1. Блок управления, который выбирает из псевдопрограммы псевдокоманды, определяет, не являются ли они машинными командами, и, если являются, то исполняет их, если же они задают операции над полиномиалами, то подготавливает работу нужного блока и передает на него управление.

2. Блок элементарных подстановок, выполняющий двоякую функцию.

Во-первых, он в заданный полиномиал P подставляет всюду вместо q^m, где q -заданная буква, полиномиэл Q_q^{m-1}, где Q - полиномиал, подставляемый вместо q.

Во-вторых, вырабатывает полиномиалы

$$\frac{\partial R}{\partial x_1} y_{ik}, \qquad \frac{\partial * R}{\partial t_k} \tag{6}$$

для производной (5). Какую из этих функций будет выполнять блок зависит от того, на какой из двух его входов передано управление.

3. Блок управления подстановкой осуществляет операцию подстановки полиномиалов $Q_1, \ldots, Q_n$ в P так, как она была описана в разд. 1 путем многократного обращения к блоку элементарной подстановки.

4. Блок управления дифференцированием выбирает из таблицы производных величины y_{ij}, затем, обращаясь к блоку элементарной подстановки, получает компоненты (6) и складывает их.

5. Блок приведения подобных членов.

6. Блок ввода, вывода, стирания, присвоения других номеров и дублирования полиномиалов.

7. Блок просмотра, имитирующий работу описанных в разд. 2 головок.

Все эти блоки сделаны по возможности независимыми друг от друга, т.е. при внесении изменений в отдельный блок остальные блоки останутся без существенных изменений - в них, как правило, изменяется только команды передачи управления на измененный блок, если они имеются.

Ни один блок АДП не использует магнитную ленту, однако при решении больших задач ее все же стоит использовать, главным образом для того, чтобы при случайном сбое машины можно было бы начать счет не с самого начала, а с какого-нибудь из промежуточных этапов. Проще всего для этого, по-видимому, время от времени записывать на ленту весь внутренний накопитель. Команда записи на ленту или считывания с ленты может стоять в псевдопрограмме в качестве псевдокоманды, в том числе команда с кодом операции 60 для обращения к стандартной программе, закоммутиро-ванной в "Стреле", проводящей операции с лентой с контрольным суммированием.

Ввиду того, что заранее обычно неизвестно, в каких ячейках памяти будет находиться данный полиномиал, записывать на ленту отдельные полиномиалы в процессе работы программы очень трудно. Часто бывает целесообразным промежуточные полиномиалы, занимающие много места в памяти, для уменьшения нехватки памяти на то время, пока они не нужны, выводить на перфокарты и в памяти машины стирать, а затем, когда понадобится, снова их вводить с помощью соответствующих псевдокоманд.

Все операции АДП проводит в один просчет. Для организации второго просчета с целью выявления случайных сбоев машины также удобно использовать ленту. При этом также можно записывать на ленту весь внутренний накопитель, кроме одной ячейки, где находится контрольная сумма, чтобы при считывании с ленты она не затиралась.

4. ЗАКЛЮЧИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

С помощью АДП можно решать довольно разнообразные задачи.

Приведем пример одной из таких задач. В работе [6] был указан алгорифм для получения дифференциального уравнения для функции $w = w(u, v)$ в случае, когда u, v и w удовлетворяют системе квазилинейных дифференциальных уравнений

$$a_i \frac{\partial u}{\partial x} + b_i \frac{\partial u}{\partial y} + c_i \frac{\partial v}{\partial x} + d_i \frac{\partial v}{\partial y} + e_i \frac{\partial w}{\partial x} + f_i \frac{\partial w}{\partial y} = 0 (i = 1,2,3),$$

где $a_i, \ldots, f_i$ - функции от u, v и w. В этой же работе этот алгорифм в качестве примера был применен к конкретной системе

$$\frac{\partial u}{\partial x} + w v^{-k} \frac{\partial v}{\partial y} = 0, \qquad k = \text{const};$$

$$\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = 0; \qquad \frac{\partial w}{\partial y} = 0.$$

Программа смогла повторить выкладки, приведенные в работе, за 12 минут. Было получено уравнение третьего порядка (в [6] приведено не это уравнение, а уравнение второго порядка, но АДП уравнения не интегрирует и поэтому сделала несколько меньше, чем сделано в работе).

Не всегда применение АДП дает нужный эффект. Если задача имеет сложную схему вычислений и громоздкие исходные полиномиалы, то на подготовку задачи к счету уходит очень много времени. Однако есть задачи, при решении которых АДП оказывает неоценимую помощь. Сюда следует прежде всего отнести вычисление определителей, элементами которых являются буквенные выражения - полиномиалы.

Задачи, в которых требуется вычислять такие определители, иногда встречаются в некоторых разделах прикладной математики, но вычислить такой определитель, например четвертого порядка, вручную часто бывает очень утомительно. АДП эту работу проделывает довольно быстро путем подстановки буквенных выражений в определитель, расписанный в виде суммы произведений своих элементов - полиномиал, расписанный раз навсегда. Определители до четвертого порядка включительно расписываются таким образом в виде сравнительно небольших полиномиалов. Оказалось не очень трудным расписать и определитель пятого порядка, хотя он занял неполную
21 перфокарту. Определители более высокого порядка можно свести к определителям пятого порядка, причем если исходный определитель имеет много нулей среди своих элементов, то эта задача сильно упрощается, если же нулей мало, то, как показывает опыт (да и простые подсчеты), получаются, как правило, настолько громоздкие вы-ражения, что памяти машины не хватает и решать эту задачу с помощью АДП на "Стреле" нет возможности.

Применение АДП эффективно при решении большого числа однотипных задач, имеющих одну я ту же (пусть сложную) псевдопрограмму.

До тех пор, пока кодирование полиномиален не будет автоматизировано, АДП будет более эффективна в тех случаях, когда исходные данные и нужные результаты не являются громоздкими, в частности, в задачах, где требуется проверить равенство нулю каких-нибудь выражений, например определителей в буквах.

Ввиду того, что псевдопрограмма может содержать циклы и может сама себя из-менять в процессе работы, возможности АДП очень большие. С ее помощью, в част-ности, можно вести вычисления не только по алгорифмам, всегда приводящим к опре-деленной цели, но и решать - точнее, пытаться решать - задачи, сводящиеся к поиску каких-либо выражений и к перебору большого числа различных возможностей, напри-мер к поиску решений дифференциальных уравнений, и т.п. В такого ряда задачах вопрос о возможности достижения цели обычно перестает быть открытым только тогда, когда цель бывает уже достигнута, но не раньше.

Описанную программу следует рассматривать прежде всего как инструмент, не-посредственно обрабатывающий полиномиалы. При добавлении к ней других блоков, управляющих ходом выкладок, можно изменить способ задания операций над поли-номиалами, т.е. от псевдокоманд описанного здесь вида перейти к какому-нибудь совершенно другому способу. Этим можно упростить подготовку задачи к счету и тем самым расширить круг задач, решение которых с помощью АДП эффективно. Надо сказать, что при создании программы вопрос о наиболее рациональном способе задания операций над полиномиалами не решался, так как все внимание было обращено непо-средственно на техническое осуществление программы. Возможно, этот вопрос имеет много общего с проблемами, стоящими при автоматизации программирования.

Принятый нами способ кодирования полиномиалов хоть и не очень удобен и доволь-но громоздок, все же имеет то достоинство, что программа с большой скоростью рабо-ты. При других способах, предлагавшихся при создании программы, эта скорость полу-чилась в несколько раз меньше. Как мы уже говорили, можно полиномиалы кодиро-вать другими способами и специальной программой переводить коды к принятому нами виду.

При составлении псевдопрограмм можно использовать метод стандартных про-грамм из псевдокоманд.

В заключение пользуемся случаем выразить благодарность Ю.И. Морозову и особенно Б.К. Потапкину за ценные предложения по составлению АДП.

ЛИТЕРАТУРА

1. *Фиников С.П.* Метод внешних форм Картана в дифференциальной геометрии. М.; Л.: Гостехтеоретиздат, 1948. 432 с.
2. *Яненко Н.Н.* Бегущие волны системы квазилинейных уравнений // ДАН СССР. 1956. Т. 109. № 1. С. 44-47.
3. *Погодин Ю.Я., Сучков В.А., Яненко Н.Н.* О бегущих волнах уравнений газовой динамики // ПММ, 1958. Т. 22, вып. 2. С. 188-196.
4. *Сидоров А.Ф., Яненко Н.Н.* К вопросу о нестационарных плоских течениях политропного газа с прямолинейными характеристиками // ДАН СССР. 1958. Т. 123, № 5. С. 832-834.
5. *Китов А.И., Криницкий Н.А.* Электронные цифровые машины и программирование. М.: Физматгиз, 1959.

6 *Яненко Н.Н.* Сведение системы квазилинейных уравнений к одному квазилинейному уравне-нию // УМН. Т. 10,, вып. 3 (6). С 173-178.