В Ы Ч И С Л И Т Е Л Ь Н Ы Е С И С Т Е М Ы

Сборник трудов
1965 г. Института математики СО АН СССР Выпуск 17

О РЕШЕНИИ ЗАДАЧ НА УНИВЕРСАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Э.В. Евреинов, Ю.Г. Косарев

В связи с разработкой универсальных вычислительных систем высокой производительности (УВС) [1],[2] возникает необходимость рассмотрения возможно более широкого круга задач для определения эффективности их решения на УВС.

С этой целью в данной работе рассматривается 16 типов задач, охватывающих основные разделы вычислительной математики. При этом используются наиболее распространенные методы их решения. При анализе эффективности решения задач на УВС применяется методика и терминология, изложенная в [3]. Схемы параллельных алгоритмов описываются с помощью ρ -языка [4].

При разработке схем параллельных алгоритмов преследовалась цель показать принципиальную возможность эффективного решения задач на УВС с большим числом машин. Переход к алгоритмам для меньшего числа машин может производиться по схеме, описанной в работе [3].

§ 1. РЕШЕНИЕ ЗАДАЧ ЛИНЕЙНОЙ АЛГЕБРЫ

Решению задач линейной алгебры посвящено очень много работ, наиболее полное изложение ее методов дается в монографии [5], которой мы будем придерживаться в дальнейшем. Специально разработанных способов решения этих задач на УВС нет, хотя некоторая работа в этой области и была проделана [6] - [8].

Анализ известных методов решения задач линейной алгебры показывает возможность их эффективной реализации на УВС. Рассмотрим а) решение систем линейных уравнений методом последовательных приближений; к таким системам сводится большое число задач физики и техники, как непосредственно, так и при решении дифференциальных уравнений (обыкновенных и в частных производных) разностными методами; б) обращение матриц методом пополнения; в) вычисление коэффициентов характеристического полинома методом А.М. Данилевского.

Будут исследоваться преимущественно задачи с большим числом уравнений или матрицами порядка 10^3 и больше, так как менее трудоемкие задачи, хотя и можно их решать на УВС, достаточно быстро выполняются и на современных ЭВМ.

В дальнейшем будем предполагать, что каждая ЭМ имеет набор операций, определяемый кругом решаемых задач. Вычислительная система имеет структуру, аналогичную описанным в [2].

1°. Решение системы линейных уравнений методом последовательных приближений

Под последовательным приближением понимается следующий итерационный процесс. Система уравнений $AX = G$ записывается в виде:

$$X = BX + G, \qquad (1.1)$$

где $B = E - A$. Здесь A, B - матрицы, X и G - векторы, E - единичная матрица. Последовательные приближения вычисляются по формуле

$$X^{(k)} = BX^{(k-1)} + G, \qquad (1.2)$$

начиная с некоторого исходного приближения $X^{(0)}$, которое может быть выбрано произвольно. Разумеется, условия сходимости

процесса предполагаются выполненными. Он заканчивается, если

$$|X^{(K)} - X^{(K-1)}| < \varepsilon \quad (\varepsilon > 0). \qquad (I.3)$$

Этот метод называется методом простой итерации.

Формулу (I.2) представим в развернутом виде:

$$\left.\begin{aligned} x_1^{(K)} &= g_1 + b_{11}x_1^{(K-1)} + \ldots + b_{1j}x_j^{(K-1)} + \ldots + b_{1n}x_n^{(K-1)}, \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_i^{(K)} &= g_i + b_{i1}x_1^{(K-1)} + \ldots + b_{ij}x_j^{(K-1)} + \ldots + b_{in}x_n^{(K-1)}, \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n^{(K)} &= g_n + b_{n1}x_1^{(K-1)} + \ldots + b_{nj}x_j^{(K-1)} + \ldots + b_{nn}x_n^{(K-1)}. \end{aligned}\right\} \qquad (I.4)$$

В качестве нулевого приближения принимается

$$x_i^{(0)} = g_i \quad (i = 1, 2, \ldots, n).$$

На каждом шаге приближений вычисляется разность

$$\Delta x_i^{(K)} = |x_i^{(K)} - x_i^{(K-1)}|.$$

Задача считается решенной, когда выполняется условие (I.3) или, что то же,

$$\max_{1 \le i \le n} \Delta x_i^{(K)} < \varepsilon. \qquad (I.5)$$

Очевидно при вычислении каждого приближения выполняется n^2 операций умножения, n^2 операций сложения, n операций вычитания и n операций сравнения, т.е. всего $2n^2+2n$ операций. Покажем, что при числе машин n может быть получен ρ -кортеж с коэффициентом эффективности $\delta \approx 1$. Действительно, построим процесс вычислений так, что каждая ЭМ m_i производит вычисление одного из $x_i^{(K)}$ и хранит в своей памяти коэффициенты $b_{i1}, \ldots, b_{in}, g_i$ и $x_i^{(K-1)}$, а также программу своей работы.

Тогда на первом такте из машины m_1 во все остальные ЭМ будет передано число $x_1^{(K-1)}$; на втором - будут вычислены произведения $e_{i1}^{(K)} = b_{i1} \cdot x_1^{(K-1)}$; на третьем - суммы $S_{i1}^{(K)} = g_i + e_{i1}^{(K)}$.

После этого во все машины засылается число $x_2^{(K-1)}$, хранящееся в машине m_2. Затем вычисляются произведения $e_{i2}^{(K)} = b_{i2}x_2^{(K-1)}$, суммы $S_{i2}^{(K)} = S_{i1}^{(K)} + e_{i2}^{(K)}$ и т.д.

После $3n$ тактов получается $x_i^{(K)} = S_{in}^{(K)}$. Затем производятся операции вычисления $\Delta x_i^{(K)} = |x_i^{(K)} - x_i^{(K-1)}|$ и условного перехода по результату сравнения $\Delta x_i^{(K)}$ с ε.

Если $\max_{(1 \le i \le n)} \Delta x_i^{(K)} < \varepsilon$, то процесс прекращается, в противном случае, выполняется следующее приближение.

Общее число тактов для каждого шага приближения при этом будет равно $h(n) = 3n+2$, а если предусмотреть возможность совмещения пересылки числа с вычислительными операциями (в данном случае умножением), то $h(n) = 2n+2$.

На каждом такте будут работать все n машин. В результате за указанное число тактов выполняется все $2n^2+2n$ операций. Объем памяти каждой ЭМ должен состоять из $n+3$ ячеек для хранения коэффициентов $x_i^{(K-1)}$ и $x_i^{(K)}$, из двух рабочих ячеек для хранения $S_{ij}^{(K)}$ и $e_{ij}^{(K)}$, а также из r ячеек для программы. В данном случае программа будет весьма простой и не превысит IO команд. Таким образом, требуемый объем памяти ЭМ будет равен

$$V = (n+5)m_1 + rm_2, \qquad (I.6)$$

где m_1 и m_2 - количества двоичных разрядов чисел и команд, соответственно.

Процесс решения данной задачи на УВС из n элементарных машин может быть проиллюстрирован следующей граф-схемой (рис. I). На этой схеме операции, выполняемые одновременно, находятся в одной колонке. Стрелками указаны информационные связи, необходимые для выполнения каждой из операций.

Для составления программы работы системы удобнее ввести циклы. Логическая схема соответствующего ρ - алгоритма может быть записана в виде следующей строчки:

$$\dot{H}_1 \dot{F}_2 \dot{O}_3^j \dot{A}_4^j \dot{A}_5^j \dot{P}_6(j) \dot{A}_7 \dot{P}_8 \dot{Я}_9, \qquad (I.7)$$

где $\dot{H}_1$ - ρ - оператор настройки, устанавливающий состояния коммутаторов ЭМ. Как видно из рис. I, в данной задаче на каждом из этапов вычислений осуществляется пересылка кода поочередно из каждой ЭМ во все остальные, начиная с m_1. Поэтому ρ -оператор $\dot{H}_1$ устанавливает все входы и выходы ЭМ в открытом состоянии. ρ - оператор $\dot{F}_2$ устанавливает первоначальное значение индекс-регистров и счетчиков циклов во всех ЭМ.

$\dot{O}_3^j$ - ρ - оператор обмена информацией между ЭМ зависит от параметра j ($j = 1, 2, \ldots, n$). При $j=1$ передается код в линию связи из ЭМ m_1, при $j=2$ из ЭМ m_2 и, наконец, при $j=n$ из ЭМ m_n. Коды, поступающие в линию связи, воспринимаются всеми ЭМ.

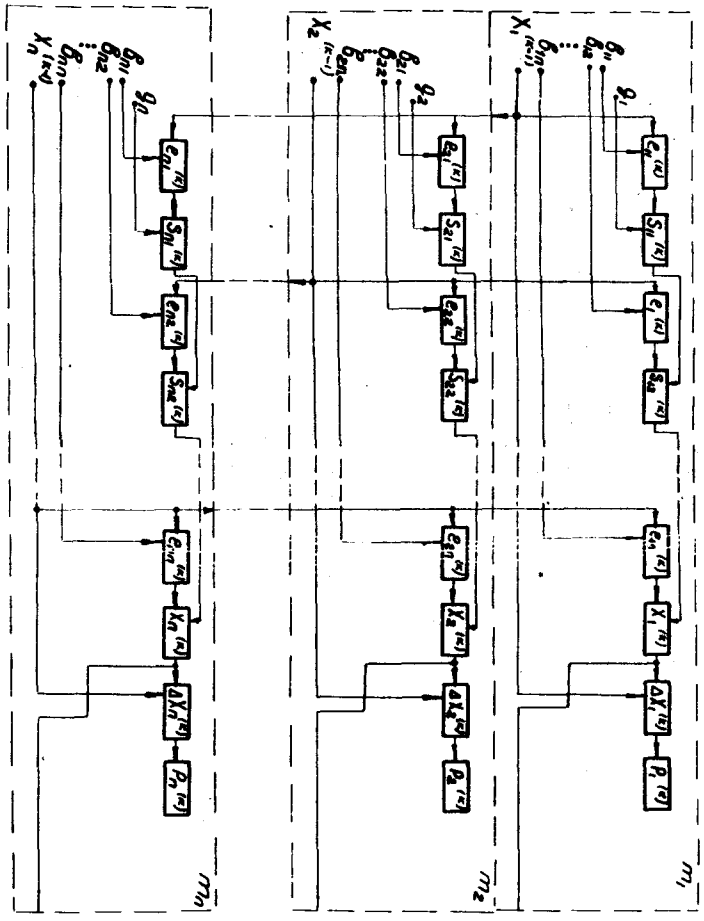


Рис. I.

A_4^j, A_5 и A_7 – арифметические ρ – операторы, производящие вычисление $e_{ij}^{(k)}$, $S_{ij}^{(k)}$ и $\Delta x_i^{(k)}$. Первый из них в зависимости от параметра j (номер цикла) изменяет адрес ячейки ОП, из которых берется соответствующее b_{ij}. $P_6(j)$, P_8 являются ρ -операторами обобщенного условного перехода. Первый из них управляет изменением параметра j и при $j=n$ переводит систему на выполнение ρ -оператора A_7. Второй проверяет, удовлетворено ли условие (I.5), если удовлетворено, то указывает на окончание решения задачи, если не удовлетворено, то передает управление ρ -оператору F_2.

Как видно из рис. I, ρ -операторы состоят из одинаковых составляющих. Благодаря этому, программы работы всех ЭМ совпадут с точностью до адресов.

Расхождение в программах, которое может возникнуть из-за отличия составляющих ρ -оператора обмена O_3^j может быть устранено. Действительно, каждую составляющую ρ -оператора O_3^j можно представить в виде логической схемы из трех операторов $P_{31}^{(ij)} Q_{32} O_{33} (A_4^j)$. Оператор $P_{31}(ij)$ проверяет совпадение номера цикла (j) с номером машины (i). При $j \neq i$ выполняется оператор O_{33}, принимающий код из канала связи и направляющий его в арифметическое устройство. При $j=i$ выполняется оператор Q_{32}, который отправляет код $(x_i^{(k-1)})$, хранящийся в ОП, во внешний канал связи, принимает его из канала связи и отправляет в арифметическое устройство, а также передает управление оператору A_4. Номер машины i у каждой ЭМ может храниться в ячейке с одним и тем же адресом. Таким образом, отличие в номерах ЭМ будет проявляться только в содержимом этой ячейки, а не программах.

Для составления программы можно воспользоваться логической схемой (I.7), заменяя соответствующие ρ -операторы простыми операторами. Нетрудно видеть, что все операции, которые использовались при решении данной задачи, являются простыми, т.е. они либо одно-, либо двуместные и по сложности не превосходят операции умножения. Таким образом, мы не допустили противоречия с условиями, которые были положены в основу данного анализа (см. [3]).

2°. Обращение матриц методом пополнения

Пусть даны квадратные матрицы n -го порядка:
$A(a_{ij})$, обратную которой $A^{-1}(\alpha_{ij})$ надо найти;

$B(b_{ij})$, отличающаяся от матрицы A только элементами строки с номером k ;

$V(v_{ij})$, у которой элементы всех строк, кроме k -ой, равны нулю, а элементы k -ой строки таковы, что $A=B+V$, и $B^{-1}(\beta_{ij})$, обратная B . Тогда, как показано в [5], любой j -й столбец матрицы A^{-1} может быть найден по формуле:

$$\alpha_j=\beta_j-\frac{(v_k,\beta_j)}{1+(v_k,\beta_k)}\beta_k\,, \qquad (I.8)$$

где α_j - столбец матрицы A^{-1} с номером j , β_j и β_k - столбцы матрицы B^{-1} с номерами j и k , соответственно; v_k - строка матрицы V с номером k , а (v_k,β_j) - скалярное произведение векторов.

При обращении матриц методом пополнения матрица A рассматривается как последний член последовательности $A_0=E, A_1, \ldots, A_n=A$, причем переход от матрицы A_{k-1} к матрице A_k осуществляется посредством замены k -ой строки матрицы A_{k-1} на k -ую строку матрицы A . В качестве начальной берется единичная матрица E . Нетрудно видеть, что у матриц A_k^{-1} строки с номерами $i>k$ останутся те же, что и у матрицы E . Матрица A^{-1} получается в результате повторения указанного процесса n раз. Применительно к этому случаю формула (I.8) для перехода на k -ом шаге примет вид:

$$\alpha_j^{(k)}=\alpha_j^{(k-1)}-\frac{(w_k,\alpha_j^{(k-1)})}{1+(w_k,\alpha_k^{(k-1)})}\alpha_k^{(k-1)}\,, \qquad (I.9)$$

где $\alpha_j^{(k)}$ и $\alpha_j^{(k-1)}$ - j -ые столбцы матриц A_k^{-1} и A_{k-1}^{-1}, соответственно. $\alpha_k^{(k-1)}$ - k -ый столбец матрицы A_{k-1}^{-1} , $w_k=(a_{k1},\ldots,a_{k,k-1},\ldots,a_{kn})$ - k -ая строка матрицы $W=A-E$. Подробнее (I.9) может быть записано в виде:

$$\alpha_{ij}^{(k)}=\alpha_{ij}^{(k-1)}-\frac{\sum_{i=1}^{n}w_{ki}\alpha_{ij}^{(k-1)}}{1+\sum_{i=1}^{n}w_{ki}\alpha_{ik}^{(k-1)}}\alpha_{ik}^{(k-1)}\,. \qquad (I.10)$$

Формула (I.10) отражает весь процесс обращения матрицы методом пополнения.

Для дальнейшего анализа процесс вычисления по этой формуле выразим более подробно. Введем обозначения для получающихся в ходе вычислений промежуточных величин. Смысл их будет ясен из самих выражений:

1) $c_{ij}^{(k)}=w_{ki}\alpha_{ij}^{(k-1)}\,,$ (I.11)

2) $d_{ij}^{(k)}=d_{i-1,j}^{(k)}+c_{ij}^{(k)};$ (I.12)
$d_{0j}^{(k)}=0\;;$

обозначим
$$e_j^{(k)}=d_{nj}^{(k)}\,.$$

Так как у матрицы A_{k-1}^{-1} строки с номерами $i\geq k$ те же, что и у матрицы E , то

3) $$\begin{cases} e_j^{(k)}=d_{k-1,j}^{(k)} & (j<k)\;; \\ e_j^{(k)}=d_{k-1,j}^{(k)}+w_{kj} & (j\geq k)\;; \end{cases} \qquad (I.13)$$

4) $f^{(k)}=1+e_n^{(k)}\;;$ (I.14)

5) $g_j^{(k)}=e_j^{(k)}/f^{(k)}\;;$ (I.15)

6) $h_{ij}^{(k)}=g_j^{(k)}\cdot\alpha_{ik}^{(k-1)}\;;$ (I.16)

7) $\alpha_{ij}^{(k)}=\alpha_{ij}^{(k-1)}-h_{ij}^{(k)}\,.$ (I.17)

Придавая индексам значения $i=1,2,\ldots,k$; $j=1,2,\ldots,n$, получаем все элементы матрицы A_k^{-1}

Нетрудно видеть, что базовая точка на характеристической функции для данного алгоритма будет при числе ЭМ, равном n . Действительно, процесс вычисления в этом случае можно распределить между n машинами таким образом, что каждая ЭМ m_j будет вычислять элементы $\alpha_{1j},\alpha_{2j},\ldots,\alpha_{kj}$ j -го столбца матрицы A_k^{-1} (рис. 2). На рис. 2 приведены также зависимости между выполняемыми операциями и дана схема обмена кодами между ЭМ. При этом предполагается, что в памяти ЭМ m_j $(j=1,2,\ldots,n)$ содержатся элементы j -го столбца $w_{jj},w_{2j},\ldots,w_{jj}$ и j -й строки $w_{j1},w_{j2},\ldots,w_{j,j-1}$ матрицы W. (В ЭМ m_1 будет храниться один элемент (w_{11}), а в ЭМ m_n - $(2n-1)$ элементов). Кроме того, в каждой ЭМ m_j запоминаются элементы $\alpha_{jj}^{(k-1)},\ldots,\alpha_{k-1,j}^{(k-1)}$ предыдущей промежуточной матрицы A_{k-1}^{-1}. На данной схеме представлены арифметические операторы. При этом только в двух случаях число одновременно выполняемых операций не равно n . При вычислении $e_j^{(k)}$ оно равно $n-k+1$

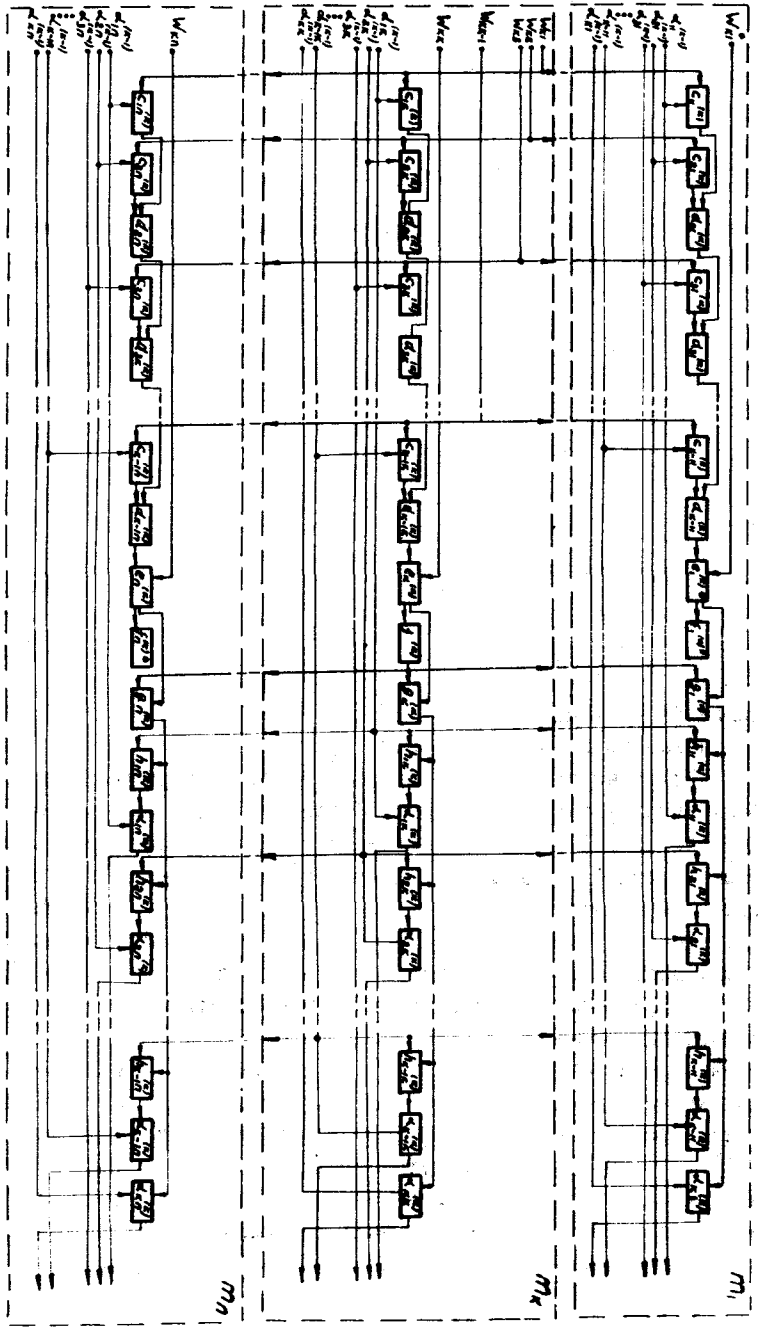


Рис. 2.

операции, при вычислении $f^{(k)}$ -одной операции. (На схеме для единообразия программы введены пустые операции, помеченные звездочкой). Нетрудно видеть, что коэффициент δ в данном случае отличается от 1 на величину, меньшую $1/(2n)$, которой можно пренебречь и считать, что УВС из n элементарных машин по сравнению с УВС из одной машины ускоряет выполнение данной задачи в n раз (без учета выигрыша за счет увеличения суммарного объема оперативной памяти). Беспетлевая схема, показанная на рис. 2, удобна для выявления связей между операторами. Для программирования эту схему нужно свернуть путем введения циклов и добавить операторы, организующие процесс вычислений на системе. В результате логическая схема ρ -алгоритма может быть записана в виде следующей строчки ρ -операторов.

$$\dot{H}_1\dot{P}^{k}_{2}\dot{O}^{k}_{3}\dot{A}^{k}_{4}\dot{O}^{k}_{5}\dot{A}^{k}_{6}\dot{A}^{k}_{7}\dot{P}^{k}_{8}(k)\dot{A}^{k}_{9}\dot{Q}^{k}_{10}\dot{A}^{k}_{11}\dot{O}^{k}_{12}\dot{A}^{k}_{13}\dot{A}^{k}_{14}\dot{P}^{k}_{15}(k)\dot{A}^{k}_{16}\dot{P}^{k}_{17}\dot{Я}_{18}\dot{H}^{k}_{19}\ . \quad (I.I8)$$

Операторы $\dot{H}_1$ и $\dot{H}^{k}_{19}$ $(k=2,3,\ldots,n)$ являются ρ -операторами настройки, устанавливающими состояния коммутаторов ЭМ. Из схемы (рис. 2) видно, что на каждом этапе обмен кодами между ЭМ осуществляется так, что одна ЭМ (номер которой совпадает с номером этапа) передает код во все остальные ЭМ.Таким образом, к системе связи и конструкции коммутатора предъявляются минимальные требования. Например, вполне достаточно соединить ЭМ кольцевым двусторонним каналом связи, к которому каждая ЭМ подключается с помощью простого коммутатора, состоящего из нескольких конъюнкторов и дизъюнкторов (рис. 3).Кроме информационного канала имеется особый канал для передачи управляющего сигнала ω , включающего все ЭМ для приема информации. В блоке управления коммутацией достаточно помнить значения двух двоичных чисел C_{10} и C_{02} . Естественно, что все более сложные виды системы связи, описанные в [2], также могут реализовать указанные функции. На всех этапах необходимо, чтобы у всех ЭМ входы были открыты $C^{(j)}_{10}=1\ (j=1,2,\ldots,n)$, а выходы закрыты, кроме ЭМ m_k , у которой номер совпадает с номером этапа, $C^{(j)}_{02}=0\ (j\neq k)$, $C^{(k)}_{02}=1$. ρ -оператор $\dot{H}_1$ может быть реализован за три такта. На первом - всем $C^{(j)}_{10}$ $(j=1,2,\ldots,n)$ придается значение I, на втором - всем $C^{(j)}_{02}$ $(j=1,2,\ldots,n)$ придается значение 0 и на третьем - $C^{(k)}_{02}$ придается значение I. Управление настройкой может осуществлять-

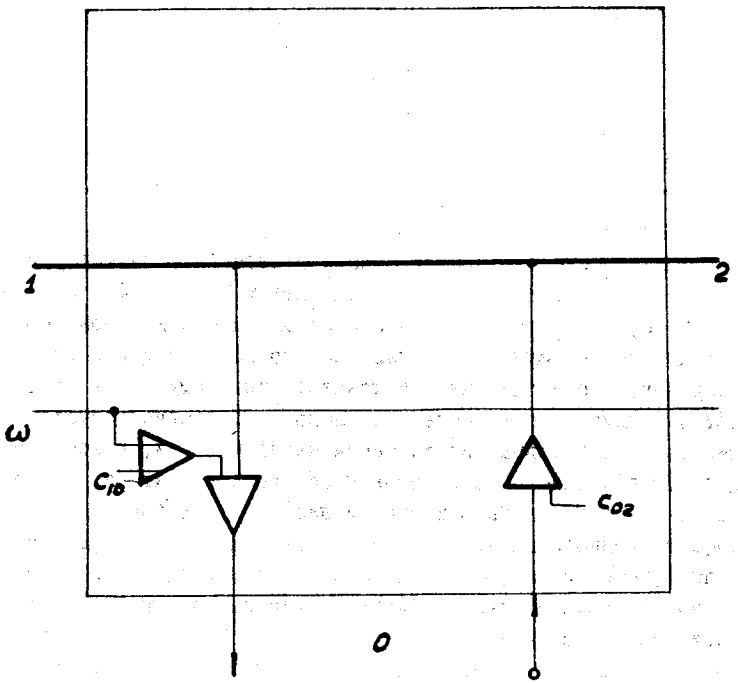


Рис. 3.

ся машиной m_k , номер которой совпадает с номером этапа. Каждый этап оканчивается ρ -оператором $\dot{H}_{19}^{k}$, который можно реализовать следующим образом.

В каждой ЭМ m_j $(j=1,2,\ldots,n-1)$ в ячейках ОП с одними и теми же адресами будем хранить команды настройки: установить в машине m_{j+1} значение $C_{02}^{j+1}=1$, а в самой ЭМ m_j - значение $C_{02}^{j}=0$. При выполнении оператора $\dot{H}_{19}^{k}$ обе команды с адресами машин, которым они предназначаются, посылаются в канал связи. Благодаря тому, что у всех ЭМ, кроме m_k , выходы заперты, в него поступают только команды, хранящиеся в ЭМ m_k. Первая из этих команд воспринимается и исполняется машиной m_{k+1}, вторая - самой ЭМ m_k.

ρ - оператор $\dot{F}_2^{k}$ изменяет параметр k (номер этапа) на единицу.

ρ - операторы $\dot{O}_3^{k}$, $\dot{O}_5^{k}$, $\dot{O}_{12}^{k}$ являются операторами обмена информацией между ЭМ. Каждый из этих ρ - операторов действует одинаковым образом на все ЭМ. При этом код, хранящийся в ОП ЭМ, по заданному адресу посылается в канал связи. Так как выход отперт только у машины m_k , то в канал связи поступает код только из нее. Этот код из канала связи поступает на входы всех ЭМ (в том числе и самой m_k). Выполнение ρ -операторов обмена можно представить в виде двух операций: 1) пересылки кода из ОП в канал связи, 2) пересылки кода из канала связи в арифметическое устройство.

ρ - оператор $\dot{A}_4^{k}$ производит вычисление $C_j^{(k)}$ $(j=1,2,\ldots,n)$.

ρ - операторы $\dot{A}_6^{k}, \dot{A}_7^{k}, \dot{A}_{13}^{k}, \dot{A}_{14}^{k}, \dot{A}_{16}^{k}$ вычисляют соответственно

$C_{ij}^{(k)}, d_{ij}^{(k)}$ $(i=2,3,\ldots,k-1; j=1,2,\ldots,n)$, $h_{ij}^{(k)}, \alpha_{ij}^{(k)}$ $(i=1,2,\ldots,k-1; j=1,2,\ldots,n)$,

$\alpha_{kj}^{(k)}$ $(j=1,2,\ldots,n)$

ρ - операторы $\dot{P}_8^{k}(K)$ и $\dot{P}_{15}^{k}(K)$ являются ρ - операторами обобщенного условного перехода, устанавливающими конец цикла после того, как он повторился соответственно $K-2$ и K раз. Число циклов меняется от этапа к этапу (с помощью оператора $\dot{F}_2^{k}$), поэтому эти ρ - операторы зависят от параметра k . ρ - оператор $\dot{P}_{17}$ осуществляет то же самое по отношению к числу этапов.

ρ - операторы $\dot{A}_9^{k}$, $\dot{Q}_{10}^{k}$ и $\dot{A}_{11}^{k}$ осуществляют вычисление $e_j^{(k)}$, $f_j^{(k)}$ и $g_i^{(k)}$ $(j=1,2,\ldots,n)$. Как упоминалось выше, для единообразия программ всех ЭМ введены пустые операции

$e_j^{(k)}(j=1,2,\dots,k-1)$ и $f_j^{(k)}(j=1,2,\dots,k-1,k+1,\dots,n)$. Для этого в памятях соответствующих ЭМ в ячейках, в которых должны разместиться элементы $w_{kj}(j=1,2,\dots,k-1;\ k=1,2,\dots,n)$ матрицы W помещены нули. Вычисление $f_j^{(k)}$ во всех ЭМ, а не только в ЭМ m_k также ничего не изменяет. ρ - оператор Q_{10}^{k} направляет в канал связи результат вычисления функции $f_j^{(k)}$. Благодаря тому, что состояние коммутаторов таково, что выходы всех ЭМ, кроме m_k, отключены от общего канала связи, в последующий поступит только требуемый код $f_k^{(k)}$. ρ - оператор $\mathcal{A}_{18}$ является оператором конца работы.

Из вышесказанного можно сделать вывод, что программы работы у всех n элементарных машин совпадают с точностью до адресов и значений индекс-регистров. Поэтому составление программы выполнения данного алгоритма на УВС мало чем отличается от обычного программирования. Так как все составляющие ρ - операторов одинаковы, то логическую схему работы системы (I.I8) можно рассматривать как логическую схему работы одной ЭМ, заменяя каждый ρ - оператор соответствующим простым оператором.

3°. Вычисление собственных значений матрицы по методу Данилевского [5]

Задача заключается в нахождении всех корней λ_i характеристического полинома (собственных значений) матрицы A.

$$|A-Et|=\begin{vmatrix} a_{11}-t & a_{12} \dots a_{1n} \\ a_{21} & a_{22}-t \dots a_{2n} \\ \dots\dots\dots\dots \\ a_{n1} & a_{n2} \dots a_{nn}-t \end{vmatrix}=(-1)^n\left[t^n-p_1t^{n-1}-\dots-p_n\right]=0 \quad \text{(I.I9)}$$

Идея метода Данилевского состоит в следующем. Данная матрица A рассматривается как матрица оператора в базисе

$$e_1=(1,0,\dots,0)',\ e_2=(0,1,\dots,0)',\dots,e_n=(0,0,\dots,1)'. \quad \text{(I.20)}$$

В этой формуле штрих (знак транспонирования) означает, что в скобках находятся элементы столбца. Предполагается, что векторы $e_1, Ae_1, \dots, A^{n-1}e_1$ линейно независимы. Тогда

$$Ae_1^n=p_1A^{n-1}e_1+p_2A^{n-2}e_1+\dots+p_ne_1, \quad \text{(I.2I)}$$

где $p_1,\dots,p_n$ - искомые коэффициенты характеристического полинома.

В базисе $e_1, Ae_1, \dots, A^{n-1}e_1$ рассматриваемый оператор имеет матрицу Фробениуса:

$$P=\begin{bmatrix} 0 & 0 & \dots & p_n \\ 1 & 0 & \dots & p_{n-1} \\ 0 & 1 & \dots & p_{n-2} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & p_1 \end{bmatrix} \quad \text{(I.22)}$$

содержащую в явном виде искомые коэффициенты характеристического полинома.

Переход от базиса $e_1, e_2, \dots, e_n$ к базису $e_1, Ae_1, \dots, A^{n-1}e_1$ осуществляется за $n-1$ шагов. Каждый шаг состоит в переходе от базиса $e_1, Ae_1, \dots, A^{k-1}e_1, e_{k+1}, \dots, e_n$ к базису $e_1, \dots, A^{k-1}e_1, A^ke_1, e_{k+2}, \dots, e_n$. При этом предполагается, что все промежуточные системы действительно являются базисами, т.е. состоят из линейно независимых векторов.

Обозначим через $A^{(k)}$ матрицу, полученную на $k-1$ -ом шаге, так что $A=A^{(1)}, P=A^{(n)}$. Столбцами матрицы $A^{(k)}$ являются координаты векторов $Ae_1, Ae_2, \dots, A^ke_1, Ae_{k+1}, \dots, Ae_n$ в базисе $e_1, Ae_1, \dots, A^{k-1}e_1, e_{k+1}, \dots, e_n$. Поэтому первые $k-1$ столбцов матрицы $A^{(k)}$ будут совпадать с одноименными столбцами матрицы Фробениуса P. Имеем

$$A^{(k+1)}=S_k^{-1}A^{(k)}S_k, \quad \text{(I.23)}$$

где S_k - соответствующая матрица преобразования координат:

$$S_k=\begin{bmatrix} 1 \dots S_{1,k+1} \dots 0 \\ 0 \dots S_{2,k+1} \dots 0 \\ \dots\dots\dots \\ 0 \dots S_{k+1,k+1} \dots 1 \\ \dots\dots\dots \\ 0 \dots S_{n,k+1} \dots 1 \end{bmatrix}, S_k^{-1}=\begin{bmatrix} 1 \dots -\frac{S_{1,k+1}}{S_{k+1,k+1}} \dots 0 \\ 0 \dots -\frac{S_{2,k+1}}{S_{k+1,k+1}} \dots 0 \\ \dots\dots\dots\dots\dots \\ 0 \dots \frac{1}{S_{k+1,k+1}} \dots 0 \\ \dots\dots\dots\dots\dots \\ 0 \dots -\frac{S_{n,k+1}}{S_{k+1,k+1}} \dots 1 \end{bmatrix}, \quad \text{(I.24)}$$

где $S_{1,k+1}, \ldots, S_{k+1,k+1}, \ldots, S_{n,k+1}$ – элементы $a_{ik}^{(k)}$ k – го столбца матрицы $A^{(k)}$.

Вычисление матрицы $A^{(k+1)}$ производится в два приема. Сначала вычисляется вспомогательная матрица $B^{(k)} = S_k^{-1} A^{(k)}$, для чего достаточно $(k+1)$ – ую строку матрицы A^k умножить на $\frac{1}{a_{k+1,k}^{(k)}}$, а от каждой из остальных строк отнять полученную $(k+1)$-ую строку матрицы $B^{(k)}$, умноженную на соответствующий элемент k -го столбца матрицы $A^{(k)}$. Затем матрица $B^{(k)}$ умножается справа на S_k.

Матрицы $A^{(k)}$ и $B^{(k)}$ имеют вид:

$$A^{(k)} = \begin{bmatrix} 0 & 0 & \ldots & 0 & a_{1k}^{(k)} & a_{1k+1}^{(k)} & a_{1k+2}^{(k)} & \ldots & a_{1n}^{(k)} \\ 1 & 0 & \ldots & 0 & a_{2k}^{(k)} & a_{2k+1}^{(k)} & a_{2k+2}^{(k)} & \ldots & a_{2n}^{(k)} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & a_{kk}^{(k)} & a_{kk+1}^{(k)} & a_{kk+2}^{(k)} & \ldots & a_{kn}^{(k)} \\ 0 & 0 & \ldots & 0 & a_{k+1k}^{(k)} & a_{k+1k+1}^{(k)} & a_{k+1k+2}^{(k)} & \ldots & a_{k+1n}^{(k)} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & a_{nk}^{(k)} & a_{nk+1}^{(k)} & a_{nk+2}^{(k)} & \ldots & a_{nn}^{(k)} \end{bmatrix}, \quad \text{(I.25)}$$

$$B^{(k)} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 & b_{1k+1}^{(k)} & a_{1k+2}^{(k+1)} & \ldots & a_{1n}^{(k+1)} \\ 1 & 0 & \ldots & 0 & 0 & b_{2k+1}^{(k)} & a_{2k+2}^{(k+1)} & \ldots & a_{2n}^{(k+1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 & b_{kk+1}^{(k)} & a_{kk+2}^{(k+1)} & \ldots & a_{kn}^{(k+1)} \\ 0 & 0 & \ldots & 0 & 1 & b_{k+1k+1}^{(k)} & a_{k+1k+2}^{(k+1)} & \ldots & a_{k+1n}^{(k+1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & 0 & b_{nk+1}^{(k)} & a_{nk+2}^{(k+1)} & \ldots & a_{nn}^{(k+1)} \end{bmatrix} \quad \text{(I.26)}$$

Переход от матрицы $A^{(k)}$ к матрице $A^{(k+1)}$ происходит по формулам:

$$b_{k+1,j}^{(k)} = \frac{1}{a_{k+1k}^{(k)}} a_{k+1j}^{(k)} ;$$

$$b_{ij}^{(k)} = a_{ij}^{(k)} - a_{ik}^{(k)} b_{k+1j}^{(k)} \quad (i \neq k+1) ;$$

$$a_{ij}^{(k+1)} = b_{ij}^{(k)} \quad (j \neq k+1) ; \quad \text{(I.27)}$$

$$a_{ik+1}^{(k+1)} = \sum_{j=1}^{n} b_{ij}^{(k)} a_{jk}^{(k)} .$$

Сначала вычисляются элементы строки матрицы $B^{(k)}$ с номером $k+1$. Затем вычисляются элементы всех остальных строк матрицы $B^{(k)}$. Так как первые k столбцов матрицы $B^{(k)}$ и матрицы $A^{(k+1)}$ совпадают с соответствующими столбцами матрицы Фробениуса, то вычисляются только элементы $n-k$ последних столбцов. Последние столбцы матрицы $B^{(k)}$, начиная с $(k+2)$ -го, совпадают с соответствующими столбцами искомой матрицы $A^{(k+1)}$. Поэтому остается определить только элементы $(k+1)$ -го столбца.

Для перехода к логической схеме напишем эти формулы так, чтобы все операции были простыми:

$$b_{k+1,j}^{(k)} = \frac{a_{k+1,j}^{(k)}}{a_{k+1k}^{(k)}} \quad (j = k+1; k+2, \ldots, n) ; \quad \text{(I.28)}$$

$$c_{ij}^{(k)} = a_{ik}^{(k)} \cdot b_{k+1j}^{(k)} \quad (i = 1,2,\ldots,k,k+2,\ldots,n; j = k+1,k+2,\ldots,n) ; \quad \text{(I.29)}$$

$$b_{ij}^{(k)} = a_{ij}^{(k)} - c_{ij}^{(k)} \quad (i = 1,2,\ldots,k,k+2,\ldots,n; j = k+1,k+2,\ldots,n) ; \quad \text{(I.30)}$$

$$d_{ij}^{(k)} = b_{ij}^{(k)} a_{jk}^{(k)} \quad (i = 1,2,\ldots,n; j = k+1,k+2,\ldots,n) ; \quad \text{(I.31)}$$

$$e_{ij}^{(k)} = e_{ij-1}^{(k)} + d_{ij}^{(k)} \quad (i = 1,2,\ldots,n; j = k+2,k+3,\ldots,n) ; \quad \text{(I.32)}$$

$$e_{ik+1}^{(k)} = 0 ;$$

$$a_{i\,k+1}^{(k+1)} = e_{in}^{(k)} + a_{i-1\,k}^{(k)} \quad (i=2,\dots,k+1);$$

$$a_{i\,k+1}^{(k+1)} = e_{in}^{(k)} \quad (i=1,k+2,k+3,\dots,n). \qquad (I.33)$$

Если, как и в предыдущих задачах, использовать n машин, то ρ -кортеж можно делать близким к оптимальному. Действительно, если в каждой машине m_i будем вычислять элементы i -й строки матрицы $A^{(k+1)}$ (процесс вычислений можно проиллюстрировать схемой (рис.4), на которой показаны пути передачи информации), то логическая схема соответствующего ρ -алгоритма может быть записана в виде:

$$\dot{H}_1 \dot{F}_2^k \dot{A}_3^k \dot{O}_4^k \dot{A}_5^k \dot{A}_6^k \dot{O}_7^k \dot{A}_8^k \dot{O}_9^{kj} \dot{A}_{10}^{kj} \dot{A}_{11}^{kj} \dot{A}_{12}^{kj} \dot{A}_{13}^{kj} \dot{A}_{14}^{kj} \dot{P}_{15}^{kj} \dot{H}_{16} \dot{O}_{17}^k \dot{A}_{18}^k \dot{P}_{19}^{kj} \dot{P}_{20} \qquad (I.34)$$

($k=1,2,\dots,(n-1)$; $j=k+2,k+3,\dots,n$)

В этой схеме $\dot{H}_1$ и $\dot{H}_{16}$ ρ - операторы настройки. Первый настраивает коммутаторы всех ЭМ на прием и передачу информации. Второй настраивает коммутатор каждой ЭМ m_i на прием кодов из ЭМ m_{i-1} и передачу кодов в ЭМ m_{i+1}.

$\dot{F}_2^k$ - ρ - оператор, формирующий очередной этап вычислений.

$\dot{A}_3^k$ - ρ - оператор, вычисляющий элемент $b_{k+1\,k+1}^{(k)}$. Только одна его составляющая ($(k+1)$ -ая) отлична от нуля.

$\dot{O}_4^k$, $\dot{O}_9^{kj}$ - ρ - операторы обмена. Эти ρ - операторы передают коды ($b_{k+1\,k+1}^{(k)}$ и $b_{k+1\,j}^{(k)}$ $(j=k+2,\dots,n)$, соответственно) из машины m_{k+1} во все ЭМ системы.

$\dot{A}_5^k$, $\dot{A}_{10}^{kj}$ - арифметические ρ - операторы, вычисляющие $c_{i\,k+1}^{(k)}$ и $c_{ij}^{(k)}$ $(j=k+2,\dots,n)$, соответственно. У этих ρ - операторов составляющая $(k+1)$ -ая пустая.

$\dot{A}_6^k$ и $\dot{A}_{11}^{kj}$ - арифметические ρ - операторы, вычисляющие $b_{i\,k+1}^{(k)}$ и $b_{ij}^{(k)}$ $(i\neq k+1)$, $b_{k+1\,k+2}^{(k)}$ и $b_{k+1\,j}^{(k)}$ $(j=k+2,\dots,n)$, соответственно. Составляющая $(k+1)$ -ая у этих ρ - операторов вычисляется по иной формуле, чем все остальные составляющие. При $j=n$ в ЭМ m_{k+1} вычисляется ненужная функция $b_{k+1\,n+1}^{(k)}$, введенная для единообразия.

$\dot{O}_7^k$, $\dot{O}_{12}^{kj}$ - операторы обмена. Первый из ЭМ m_{k+1} передает код $a_{k+1\,k}$ во все остальные ЭМ. Второй из ЭМ m_j передает код a_{jk} во все остальные ЭМ $(j=k+2,\dots,n)$.

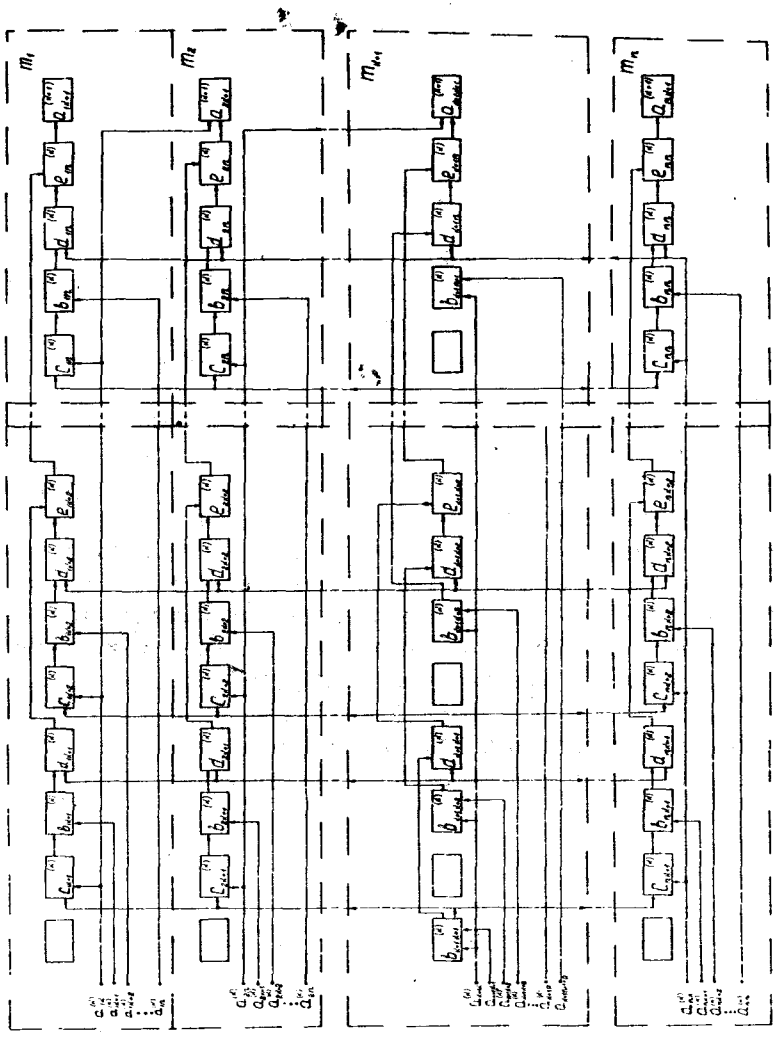


Рис. 4.

A_8^k и A_{13}^{kj} – арифметические ρ – операторы, вычисляющие $a_{i\,k+1}^{(k)}$ и $a_{ij}^{(k)}$ $(j=k+2,\ldots,n)$, соответственно.

A_{14}^{kj} – арифметический ρ – оператор, вычисляющий e_{ij} $(i=1,2,\ldots,n;\ j=1,2,\ldots,n)$.

$P_{15}^k(j)$ и $P_{19}(k)$ ρ – операторы обобщенного условного перехода, первый управляет циклом по параметру j $(j=k+2,k+3,\ldots,n)$, второй – циклом по параметру k $(k=1,2,\ldots,n-1)$.

O_{17}^k ρ – оператор обмена, осуществляющий прием и передачу кодов a_{ik} в линии связи для ЭМ m_i $(i=2,3,\ldots,k)$, причем только передачу – для машины m_1 и прием – для машины m_{k+1}.

A_{18}^k – арифметический ρ – оператор, производящий вычисление элементов $a_{i\,k+1}^{(k+1)}$. Последние вычисляются в ЭМ m_i $(i=2,3,\ldots,k+1)$ путем сложения $e_{in}^{(k)} \, a_{i-1\,k}^{(k)}$, а во всех остальных ЭМ путем пересылки $e_{in}^{(k)}$ в ячейки памяти $\langle a_{i\,k+1}^{(k)} \rangle$.

Рассмотрение трех указанных задач линейной алгебры показывает, что решение их может быть представлено в виде одинаковых (или почти одинаковых) ветвей вычисления, что упрощает их программирование. Ветви вычислений могут быть распределены между элементарными машинами таким образом, что хранимая информация V при этом равномерно распределяется между ЭМ, то есть в системе из n машин, каждая из них должна хранить объем информации V/n. Обмен информацией между ЭМ при этом может быть осуществлен с помощью простейшей одно- или двухканальной системы связи.

Так как рассмотренные задачи являются типовыми, то можно сделать вывод, что задачи линейной алгебры могут эффективно решаться на УВС.

§ 2. РЕШЕНИЕ ЭКСТРЕМАЛЬНЫХ ЗАДАЧ МЕТОДОМ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

В данном параграфе рассматривается решение на вычислительной системе общей задачи линейного программирования и общей транспортной задачи [9] – [15].

I°. Решение общей задачи линейного программирования модифицированным симплексным методом. x)

Общая задачи линейного программирования заключается в отыскании минимума функции.

$$c_1 x_1 + c_2 x_2 + \ldots + c_j x_j + \ldots + c_n x_n \qquad (2.1)$$

при следующих ограничениях:

$$x_j \geqslant 0, \quad j=1,2,\ldots,n \qquad (2.2)$$

и
$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n = b_1,\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n = b_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n = b_i,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mj}x_j + \ldots + a_{mn}x_n = b_m,
\end{aligned} \qquad (2.3)$$

где c_j, a_{ij} и b_i – заданные постоянные величины, $b_i \geqslant 0$, $m<n$.

Значения $x_1, x_2, \ldots, x_n$, удовлетворяющие условиям (2.2) и (2.3), принято называть п л а н о м, а если при этом функция (2.1) принимает минимальное значение, то – о п т и м а л ь н ы м п л а н о м.

В модифицированном симплексном методе вводятся дополнительные переменные $x_{n+1}, x_{n+2}, \ldots, x_{n+m} \geqslant 0$, а также x_{n+m+1} и x_{n+m+2}:

$$x_{n+m+1} = -\sum_{j=1}^{n} a_{m+1,j}\, x_j, \qquad (2.4)$$

где $a_{m+1,j} = c_j$.

$$x_{n+m+2} = b_{m+2} - \sum_{j=1}^{n} a_{m+2,j}\, x_j, \qquad (2.5)$$

где
$$\left.\begin{aligned}
a_{m+2,j} &= -\sum_{i=1}^{m} a_{ij},\ j=1,2,\ldots,n\\
b_{m+2} &= -\sum_{i=1}^{m} b_i.
\end{aligned}\right\} \qquad (2.6)$$

---

x) См., например, [11].

Задача сводится к задаче максимизации

$$x_{n+m+1}$$

при условиях:

$$\left.\begin{array}{ll}
a_{11}x_1+a_{12}x_2+\ldots a_{1n}x_n+x_{n+1} & =b_1\\
a_{21}x_1+a_{22}x_2+\ldots a_{2n}x_n \quad +x_{n+2} & =b_2\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\
a_{m1}x_1+a_{m2}x_2+\ldots a_{mn}x_n \quad +x_{n+m} & =b_m\\
a_{m+1,1}x_1+a_{m+1,2}x_2+\ldots a_{m+1,n}x_n \quad +x_{n+m+1} & =0\\
a_{m+2,1}x_1+a_{m+2,2}x_2+\ldots a_{m+2,n}x_n \quad +x_{n+m+2} & =b_{m+2}
\end{array}\right\} \quad (2.7)$$

$$\left.\begin{array}{ll}
x_1 & \geqslant 0\\
x_2 & \geqslant 0\\
\ldots & \ldots\\
x_n & \geqslant 0\\
x_{n+1} & \geqslant 0\\
\ldots & \ldots\\
x_{n+m} & \geqslant 0
\end{array}\right\} \quad (2.8)$$

Задача решается в два этапа. На первом добиваются того, чтобы $x_{n+m+2}=0$. При этом, так как на основании (2.5) и (2.7)

$$-x_{n+m+2}=x_{n+1}+x_{n+2}+\ldots+x_{n+m} \qquad (2.9)$$

и $x_{n+1}, x_{n+2}, \ldots, x_{n+m} \geqslant 0$, следует, что все x_{n+1}, $x_{n+2}, \ldots, x_{n+m}$ равны нулю. Это означает, что $x_j\ (j=1,2,\ldots,m)$ удовлетворяют условиям (2.3), то есть образуют план. На втором этапе этот план рассматривается как первоначальный (опорный), последовательным улучшением которого получают оптимальный план.

Рассмотрим процесс выполнения обоих этапов.

Этап I. Здесь повторяется один и тот же цикл вычислений до тех пор, пока не станет $x_{n+m+2}=0$, либо не будет установлена невозможность решения. В качестве исходного плана

берутся значения переменных $x_j=0\ (j=1,\ldots,n)$ и $x_{n+i}=b_i$, $i=1,2,\ldots,m$. Основные операции производятся над элементами матрицы, составленной из коэффициентов системы условий (2.7):

$$\bar{A}=\begin{bmatrix}
a_{11} & a_{12} & \ldots & a_{1j} & \ldots & a_{1n}\\
a_{21} & a_{22} & \ldots & a_{2j} & \ldots & a_{2n}\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\
a_{i1} & a_{i2} & \ldots & a_{ij} & \ldots & a_{in}\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\
a_{m1} & a_{m2} & \ldots & a_{mj} & \ldots & a_{mn}\\
a_{m+1,1} & a_{m+1,2} & \ldots & a_{m+1,j} & \ldots & a_{m+1,n}\\
a_{m+2,1} & a_{m+2,2} & \ldots & a_{m+2,j} & \ldots & a_{m+2,n}
\end{bmatrix} \qquad (2.10)$$

и вспомогательных матриц, получающихся на каждом цикле вычислений

$$U_s=\left[\begin{array}{cccccc|cc}
u_{11}^{(s)} & u_{12}^{(s)} & \ldots & u_{1j}^{(s)} & \ldots & u_{1m}^{(s)} & 0 & 0\\
u_{21}^{(s)} & u_{22}^{(s)} & \ldots & u_{2j}^{(s)} & \ldots & u_{2m}^{(s)} & 0 & 0\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \\
u_{i1}^{(s)} & u_{i2}^{(s)} & \ldots & u_{ij}^{(s)} & \ldots & u_{im}^{(s)} & 0 & 0\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \\
u_{m1}^{(s)} & u_{m2}^{(s)} & \ldots & u_{mj}^{(s)} & \ldots & u_{mm}^{(s)} & 0 & 0\\
\hline
u_{m+1,1}^{(s)} & u_{m+1,2}^{(s)} & \ldots & u_{m+1,j}^{(s)} & \ldots & u_{m+1,m}^{(s)} & 1 & 0\\
u_{m+2,1}^{(s)} & u_{m+2,2}^{(s)} & \ldots & u_{m+2,j}^{(s)} & \ldots & u_{m+2,m}^{(s)} & 0 & 1
\end{array}\right] \qquad (2.11)$$

В качестве исходной матрицы U_1 возьмем единичную матрицу E . Первый этап состоит из следующих шагов:

Ш а г I. Сравниваем $x_{m+n+2}^{(s)}$ с нулем. Если $x_{m+n+2}^{(s)}=0$, переходим ко П этапу. Если $x_{m+n+2}^{(s)}<0$, подсчитываем

$$\delta_j^{(s)}=u_{m+2,1}^{(s)}a_{1j}+u_{m+2,2}^{(s)}a_{2j}+\ldots+u_{m+2,m+2}^{(s)}a_{m+2,j} \qquad (2.12)$$
$$j=1,2,\ldots,n\ .$$

Ш а г П. Сравниваем $\delta_j^{(s)}$ с нулем. Если все $\delta_j^{(s)}\geqslant 0$, то

задача не имеет ни одного плана. Если хотя бы одно $\delta_j^{(s)} < 0$, то определяем

$$\delta_\kappa^{(s)} = \min_j \delta_j^{(s)}. \qquad (2.13)$$

Если минимум достигается на нескольких $\delta_j^{(s)}$, то выбираем $\delta_j^{(s)}$ с наименьшим индексом.

Ш а г III. Подсчитываем

$$x_{i\kappa}^{(s)} = u_{i1}^{(s)} a_{1\kappa} + u_{i2}^{(s)} a_{2\kappa} + \ldots + u_{i,m+2}^{(s)} a_{m+2,\kappa} \qquad (2.14)$$

для $i = 1, 2, \ldots, m+2$

и определяем

$$\theta_0 = \min_i \frac{x_i^{(s)}}{x_{i\kappa}^{(s)}} = \frac{x_e^{(s)}}{x_{e\kappa}^{(s)}}, \quad i = 1, 2, \ldots, m. \qquad (2.15)$$

Минимум берется по $x_{i\kappa} > 0$. Если минимум достигается одновременно для нескольких значений i, выбираем x_i с наименьшим индексом.

Ш а г IУ. Определяем новые значения переменных в опорном плане.

$$x_i^{(s+1)} = x_i^{(s)} - \frac{x_e^{(s)}}{x_{e\kappa}^{(s)}} x_{i\kappa} \quad (i \neq \kappa);$$

$$x_\kappa^{(s+1)} = \frac{x_e^{(s)}}{x_{e\kappa}^{(s)}}. \qquad (2.16)$$

Затем определяем элементы матрицы U_{s+1} по формулам :

$$u_{ij}^{(s+1)} = u_{ij}^{(s)} - \frac{u_{ej}^{(s)}}{x_{e\kappa}^{(s)}} x_{i\kappa}^{(s)} \qquad \begin{matrix} (j = 1, 2, \ldots, m) \\ (i \neq e) \end{matrix}$$

$$u_{ej}^{(s+1)} = \frac{u_{ej}^{(s)}}{x_{e\kappa}^{(s)}} \qquad (2.17)$$

После этого снова переходим к шагу I данного этапа.

Э т а п II. Состоит из следующих шагов:

Ш а г I. Подсчитываем

$$\gamma_j^{(s+1)} = u_{m+1,1}^{(s+1)} a_{1j} + u_{m+1,2}^{(s+1)} a_{2j} + \ldots + u_{m+1,m+2}^{(s+1)} a_{m+2,j}, \qquad (2.18)$$

$$j = 1, 2, \ldots, n,$$

где $u_{m+i,j}^{(s+1)}$ - значение элементов матрицы U_{s+1}, при котором $x_{n+m+2} = 0$.

Ш а г II. Сравниваем $\gamma_j^{(s+1)}$ с 0. Если все $\gamma_j^{(s+1)} \geq 0$, то x_{n+m+1} достигает своего максимума и соответствующий план является оптимальным. Если хотя бы одно из $\gamma_j^{(s+1)} < 0$, то необходимо перейти к новому базису.

Пусть

$$\gamma_\kappa^{(s+1)} = \min_j \gamma_j^{(s+1)}. \qquad (2.19)$$

Тогда переменную $x_\kappa^{(s+1)}$ следует ввести в новый план. Если при этом минимум достигается одновременно на нескольких $\gamma_j^{(s+1)}$, то берем $\gamma_j^{(s+1)}$ с наименьшим индексом.

Ш а г III. Подсчитываем

$$x_{i\kappa}^{(s+1)} = u_{i1}^{(s+1)} a_{1\kappa} + u_{i2}^{(s+1)} a_{2\kappa} + \ldots + u_{i,m+2}^{(s+1)} a_{m+2,\kappa} \qquad (2.20)$$

для $i = 1, 2, \ldots, m, m+1, m+2$.

Переменную $x_e^{(s+1)}$, подлежащую исключению из старого плана, определяем из соотношения:

$$\theta_0 = \min \frac{x_i^{(s+1)}}{x_{i\kappa}^{(s+1)}} = \frac{x_e^{(s+1)}}{x_{e\kappa}^{(s+1)}}, \; i = 1, 2, \ldots, m, \qquad (2.21)$$

где минимум берется по $x_{i\kappa}^{(s+1)} > 0$.

Если все $x_{i\kappa}^{(s+1)} < 0$, то решение прекращаем, так как это означает, что линейная форма задачи не ограничена сверху.

Ш а г IУ. Определяем новые значения переменных:

$$x_i^{(s+2)} = x_i^{(s+1)} - \frac{x_e^{(s+1)}}{x_{e\kappa}^{(s+1)}} x_{i\kappa}^{(s+1)} \qquad \text{для } i \neq \kappa$$

$$x_\kappa^{(s+2)} = \frac{x_e^{(s+1)}}{x_{e\kappa}^{(s+1)}}. \qquad (2.22)$$

Элементы матрицы U преобразуются по той же формуле (2.17). Итерации продолжаем до тех пор, пока не будет определен оптимальный план, либо не будет установлена неограниченность линейной формы задачи.

Заметим, что вычисления на II этапе отличаются от соответствующих вычислений на I этапе только тем, что вместо строки матрицы U_s с номером $m+2$ берется строка той же матрицы с номером $m+1$.

Напишем, как и в предыдущих случаях, процесс решения задачи в виде простых операций.

Ш а г I.I. Сравнение $x^{(s)}_{m+n+2}$ с нулем.

2. $a^{(s)}_{ij}=u^{(s)}_{m+2,i}a_{ij}\ (i=1,2,..,m+2;j=1,2,..,n)$; (2.23)

3. $e^{(s)}_{ij}=a^{(s)}_{i-1,j}+a^{(s)}_{ij}\ (i=1,2,...,m+2;j=1,2,..,n)$. (2.24)

Ш а г II.4. Сравнение $\delta^{(s)}_i=e^{(s)}_{m+2,i}$ с предыдущим $\delta^{(s)}_{i-1}$ запоминание наименьшего из них $\delta^{(s)}_k=\min\delta^{(s)}_i\ (i=1,2,...,n)$.

5. Сравнение $\delta^{(s)}_k$ с нулем, если $\delta^{(s)}_k\geq 0$, то останов.

Ш а г III.6. $f^{(s)}_{ij}=u^{(s)}_{ij}a_{jk}\ (i=1,2,..,m+2;j=1,2,..,m+2)$. (2.25)

7. $g^{(s)}_{ij}=f^{(s)}_{i,j-1}+f^{(s)}_{ij}$. (2.26)

8. Сравнение $g^{(s)}_{i,m+2}=x^{(s)}_{ik}$ с нулем. Для $x_{ik}>0$

9. $h^{(s)}_{ik}=\frac{x^{(s)}_i}{x^{(s)}_{ik}}$. (2.27)

I0. Сравнение h_{ik} друг с другом и нахождение наименьшего из них $\theta^{(s)}_0=\min_i h_{ik}$.

Ш а г IУ.II. $\rho_{ik}=\theta^{(s)}_0 x^{(s)}_{ik}$. (2.28)

I2. $x^{(s+1)}_i=x^{(s)}_i-\rho^{(s)}_{ik}\quad (i\neq k)$;

$x^{(s+1)}_k=\frac{x^{(s)}_e}{x^{(s)}_{ek}}=\theta^{(s)}_0$. (2.29)

I3. $q^{(s)}_{ej}=\frac{u^{(s)}_{ej}}{x^{(s)}_{ek}}$. (2.30)

I4. $t^{(s)}_{ij}=q^{(s)}_{ei}x^{(s)}_{ik}$. (2.31)

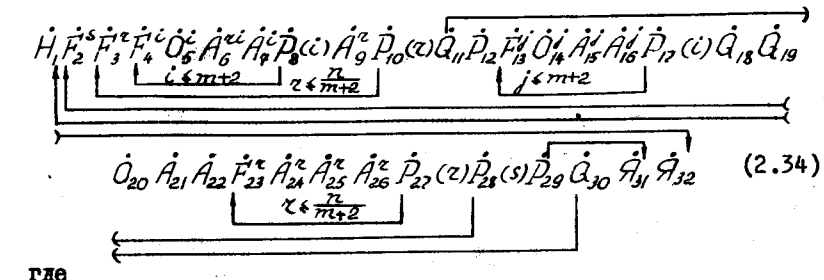
I5. $u^{(s+1)}_{ij}=u^{(s)}_{ij}-t^{(s)}_{ij}\quad (i\neq e)$ (2.32)

$u^{(s+1)}_{ej}=q^{(s)}_{ej}$ (2.33)

I6. Сравнение $x^{(s)}_{m+n+2}$ с нулем. При $x^{(s)}_{m+n+2}<0$ переход к очередной итерации.

I7. При $x^{(s)}_{m+n+2}=0$ во всех командах изменяются адреса элементов $u^{(s)}_{m+2,i}$ на адреса $u^{(s)}_{m+1,i}$ и изменяются признаки остановов, после чего выполняется очередная итерация этапа II.

В данной задаче возьмем число машин равным $m+2$. В памяти каждой машины $m_i\ (i=1,2,...,m+2)$ поместим i-ю строку матрицы U_S, соответствующие значения компонент $x^{(s)}_i$, номер компоненты плана $i^{(s)}$, а также $n/(m+2)$ столбцов матрицы A, так что в ЭМ m_1 будут находиться столбцы с номерами $1,2,..,n/(m+2)$, в ЭМ m_2 – $n/(m+2)+1, n/(m+2)+2,...,2n/(m+2)$ и т.д. В каждой ЭМ будем вычислять количество δ_j, равное $n/(m+2)$. Решение данной задачи может быть представлено в виде следующей операторной схемы ρ-алгоритма.



(2.34)

где

$\dot{H}_1$ – ρ-оператор настройки, соединяющий входы и выходы всех ЭМ системы с общим каналом связи.

$\dot{F}^{s}_2$ – ρ-оператор, формирующий выполнение цикла по s во всех $m+2$ ЭМ.

$\dot{F}^{\tau}_3$ – ρ-оператор, формирующий выполнение цикла по τ во всех ЭМ $(\tau=1,2,...,n/(m+2))$.

$\dot{F}^{i}_4$ – ρ-оператор, формирующий выполнение цикла по i во всех ЭМ $(i=1,2,...,n/(m+2))$.

$\dot{O}^{i}_5$ – ρ-оператор, осуществляющий пересылку кода $u_{m+2,i}$ из машины m_{m+2} во все остальные ЭМ.

$\dot{A}^{\tau i}_6$ – ρ-оператор, осуществляющий вычисление $a^{(s)}_{i\tau}$ во всех $m+2$ ЭМ.

$\dot{A}^{i}_7$ – ρ-оператор, осуществляющий вычисление $e^{(s)}_{i\tau}$ во всех $m+2$ ЭМ.

$\dot{P}_8(i)$ – ρ-оператор обобщенного условного перехода, осуществляющий при $i\leq m+2$ повторение цикла.

$\dot{A}^{\tau}_9$ – ρ-оператор, выполняющий сравнение $e^{(s)}_{m+2,\tau}=\delta^{(s)}_{\tau}$ с $\delta^{(s)}_{\tau-1}$ и оставляющий в ячейке $<\delta^{(s)}_{\tau-1}>$ наименьшее из этих чисел.

$\dot{P}_{10}(\tau)$ – ρ-оператор обобщенного условного перехода, осуществляющий при $\tau\leq n/(m+2)$ повторение цикла.

$\dot{Q}_{11}$ – обобщенный ρ-оператор, который находит $\delta_k=\min\delta_i\ (i=1,2,...,m+2)$. Каждое δ_i находится в ЭМ с соответствующим номером i.

$\dot{P}_{12}$ – ρ -оператор обобщенного условного перехода, который сравнивает δ_k , находящееся в соответствующей ЭМ с нулем.

При $\delta_k > 0$ осуществляется переход к ρ -оператору $\dot{Я}_{32}$.

При $\delta_k < 0$ – переход к следующему ρ -оператору.

$\dot{F}^{j}_{13}$ – ρ -оператор, формирующий выполнение цикла по j.

$\dot{O}^{i}_{14}$ – ρ -оператор обмена, посылающий код a_{jk} из ЭМ (той же, в которой находится δ_k).

$\dot{A}^{i}_{15}$ – арифметический ρ -оператор, вычисляющий $f^{(s)}_{ij}$ во всех ЭМ.

$\dot{A}^{i}_{16}$ – арифметический ρ -оператор, вычисляющий $g^{(s)}_{ij}$ во всех ЭМ.

$\dot{P}_{17}(j)$ – ρ -оператор обобщенного условного перехода, осуществляющий при $j \le m+2$ повторение цикла по j , при $j > m+2$ – переход к ρ -оператору $\dot{Q}_{18}$.

$\dot{Q}_{18}$ – обобщенный ρ -оператор, осуществляющий сравнение $g^{(s)}_{i,m+2} = x^{(s)}_{ik}$ с нулем. При $x^{(s)}_{ik} > 0$ вычисляется $h^{(s)}_{ik}$ при $x^{(s)}_{ik} \le 0$ в ячейку $\langle h^{(s)}_{ik} \rangle$ засылается наибольшее из возможных чисел (например, число (III...II)).

$\dot{Q}_{19}$ – обобщенный ρ -оператор, находящий $\theta^{(s)}_o = \min_i h^{(s)}_{ik}$. Аналогичен ρ -оператору $\dot{Q}_{11}$.

$\dot{O}_{20}$ – ρ -оператор обмена, осуществляющий передачу $\theta^{(s)}_o$ из ЭМ m_e во все ЭМ.

$\dot{A}_{21}$ – арифметический ρ -оператор, осуществляющий определение $p^{(s)}_{ik}$ во всех ЭМ, кроме m_e , в которой выполняется пустая операция.

$\dot{A}_{22}$ – арифметический ρ -оператор, вычисляющий $x^{(s+1)}_i$ во всех ЭМ, кроме m_e , в которой вместо $x^{(s+1)}_k$ подставляется θ_o .

$\dot{F}^{z}_{23}$ – ρ -оператор, формирующий изменение цикла по параметру z ($z = 1,2,\dots, n/(m+2)$).

$\dot{A}^{z}_{24}$ – арифметический ρ -оператор, вычисляющий $q^{(s)}_{ej}$ во всех ЭМ.

$\dot{A}^{z}_{25}$ – арифметический ρ -оператор, вычисляющий $t^{(s)}_{iz}$ во всех ЭМ.

$\dot{A}^{z}_{26}$ – арифметический ρ -оператор, вычисляющий $u^{(s+1)}_{iz}$ во всех ЭМ, кроме m_e , в которой в ячейку $\langle u^{(s+1)}_{ej} \rangle$ засылается $q^{(s)}_{ej}$.

$\dot{P}_{27}(z)$ – ρ -оператор обобщенного условного перехода, осуществляющий при $z \le n/(m+2)$ переход к ρ -оператору $\dot{F}^{z}_{23}$, при $z > n/(m+2)$ – к следующему ρ -оператору.

$\dot{P}_{28}(s)$ – ρ -оператор обобщенного условного перехода, осуществляющий сравнение x_{m+n+2} с нулем и при $x < 0$ выполняющий переход к ρ -оператору $\dot{F}^{(s)}_{2}$, а при $x_{m+n+2} = 0$ переход к ρ -оператору $\dot{P}_{29}$.

$\dot{P}_{29}$ – ρ -оператор обобщенного условного перехода. Если выполняется I этап, то оператор $\dot{P}_{29}$ указывает на переход к следующему ρ -оператору, если – II этап, то – к $\dot{Я}_{31}$.

$\dot{Q}_{30}$ – ρ -оператор, формирующий изменения в программах, связанных с переходом на II этап, во всех ЭМ адрес $m+n+2$ заменяет на $m+n+1$ и изменяет признак ρ-оператора $\dot{Я}_{32}$, после чего передает управление ρ -оператору $\dot{F}^{s}_{2}$.

$\dot{Я}_{31}$ – ρ -оператор останова при окончании решения задачи.

$\dot{Я}_{32}$ – ρ -оператор останова с признаком. Признак указывает, на каком этапе произошел останов.

Как можно видеть из схемы ρ -алгоритма и соответствующих формул, потеря времени за счет простаивания ЭМ при решении данной задачи на УВС будет незначительной, так как основное время затрачивается на вычисление ρ -операторов $\dot{A}_6$ и $\dot{A}_7$, при выполнении которых работают все $m+2$ ЭМ.

Таким образом, УВС из $m+2$ элементарных машин при решении данной задачи дает выигрыш во времени по сравнению с одной ЭМ в $m+2$ раз (без учета различий в объемах оперативных памятей). Увеличение числа ЭМ до n может дать существенный выигрыш во времени, однако ЭМ при этом будут использоваться менее эффективно.

2°. Р е ш е н и е о б щ е й т р а н с п о р т н о й з а д а ч и. Математическая формулировка транспортной задачи состоит в следующем. Определить значения переменных x_{ij} , которые минимизируют стоимость перевозок:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \ . \qquad (2.35)$$

При условиях

$$\sum_{j=1}^{n} x_{ij} = a_i \ , \quad i = 1,2,\dots,m \ , \qquad (2.36)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \ , \quad j = 1,2,\dots,n \ , \qquad (2.37)$$

$$x_{ij} \ge 0 .$$

Для совместности уравнений необходимо, чтобы

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} = \sum_{i}^{m} a_i = \sum_{j}^{n} b_j = A . \qquad (2.38)$$

Транспортная задача является задачей линейного программирования с $m+n$ уравнениями и mn переменными. В связи с этим для ее решения пригоден любой метод линейного программирования. Однако при больших значениях m и n это приводит к огромному количеству вычислений. Для решения транспортной задачи разработан ряд специальных методов [I2] - [I5].

Анализ этих методов показывает, что все они могут быть эффективно реализованы на вычислительной системе. Для примера рассмотрим реализацию на вычислительной системе одной из модификаций распределительного метода [I3].

Условия транспортной задачи можно представить в виде таблицы

i \ j		b_1	b_2		b_j		b_n
	v_j / u_i	v_1	v_2		v_j		v_n
a_1	u_1	c_{11} / x_{11}	c_{12} / x_{12}				
a_2	u_2	c_{21} / x_{21}	c_{22} / x_{22}				
a_i	u_i				c_{ij} / x_{ij}		
a_m	u_m						

(2.39)

в которой u_i и v_j - числа, удовлетворяющие условию

$$u_i + v_j = c_{ij} . \qquad (2.40)$$

Алгоритм решения транспортной задачи описывается следующим образом.

I. Определяем первый выбор (исходный план). Находим элементы первой строки матрицы $X=(x_{ij})$. В первой строке матрицы $C=(c_{ij})$ отыскиваем наименьший элемент. Пусть это будет элемент c_{1j_1} . Тогда $x_{1j_1} = min(a_1, b_{j_1})$ (x -выбранный элемент). Если $a_1 > b_{j_1}$, то отыскиваем в той же строке второй наименьший элемент c_{1j_2} , удовлетворяющий условию $c_{1j_2} \geq c_{1j_1}$; тогда $x_{1j_2} = min(a_1 - x_{1j_1}, b_{j_2})$. Шаги продолжаем до тех пор, пока не удовлетворим первому уравнению $a_1 = \sum_{j=1}^{n} x_{1j}$. Если на некотором шаге k окажется, что остаток от a_1 точно равен соответствующему b_{j_k} , то, положив $x_{1j_k} = b_{j_k}$, дополнительно принимаем следующее, соответствующее по величине, значение переменной $x_{1j_{k+1}} = 0$. После этого переходим ко второй строке, третьей и т.д. Полученное первое решение будет исходным планом и содержит $m+n-1$ элементов.

2. Обращаем x - выбранные элементы в нули.

а) С этой целью для x-выбранных элементов составляем систему уравнений $u_i + v_j = c_{ij}$ и находим значения u_i и v_j .

б) Преобразуем x - выбранные элементы таблицы по формуле

$$\bar{c}_{ij} = c_{ij} + u_i + v_j . \qquad (2.4I)$$

в) Если для всех x невыбранных элементов таблицы элементы $\bar{c}_{ij}$ окажутся неотрицательными, то первый выбор является оптимальным и процесс решения заканчивается (переход к шагу 6). Если нет, то осуществляется переход к следующему шагу 3.

3. Находим в таблице наименьшее из $\bar{c}_{ij}$ (если это число неотрицательное, то переходим к шагу 6).

4. Образуем замкнутую цепь наибольшего отрицательного элемента с обращенными в нуль x - выбранными элементами. Для этого вычеркиваем строки и столбцы, содержащие по одному отмеченному нулю (так как для образования цепи надо иметь в столбце или строке не менее двух нулей). После первого вычеркивания во всей таблице делаем второе и т.д., пока не придем к положению, из которого сразу видна замкнутая цепь.

5. Строим новый выбор, передвигая по цепи число x_{ij} , равное наименьшему в четной полуцепи (элементу x_{ij} , для которого $\bar{c}_{ij}$ было наибольшим по абсолютной величине отрицательным элементом, присваивается первый номер, далее нумерация идет по часовой стрелке). Наименьшее x_{ij} прибавляем к нечетным элементам и вычитаем из четных элементов цепи. Наименьшее в четной полуцепи x_{ij} исключаем из числа x - выбранных элементов. Так как наибольший отрицательный элемент стал x - выбранным, то обращаем его в нуль, но так, чтобы остальные x - выбранные элементы, соответствующие новому выбору, оставались равными нулю. Для этого переходим к шагу 2.

6. Сравниваем x_{ij} элементы с нулем. Если в преобразованной таблице все x - выбранные элементы равны нулю, а остальные неотрицательные, то это означает, что найдено оптимальное решение. Процесс закончен.

Наиболее трудоемкими в смысле выполняемого числа операций являются шаги 2б, 2в, 3. Для их выполнения требуется со-

ответственно $2m \cdot n, m \cdot n, 2m \cdot n$ операций. Для выполнения шагов 2а, 4, 5 требуется не более $2(m+n)$ операций на каждый шаг. Шаг I выполняется однократно за время решения задачи. Следовательно, время при решении задачи в основном затрачивается на выполнение шагов 2б, 2в, 3. При достаточно больших значениях m и n можно ограничиться параллельным счетом только при выполнении этих шагов. Другими словами, при решения транспортной задачи шаги I, 2а, 4, 5 выполняются на одной из машин вычислительной системы, а шаги 2б, 2в, 3 реализуются при одновременной работе всех машин ВС.

Операции шага 2б сводятся к операциям сложения набора чисел u_i и v_j с элементами матрицы (c_{ij}) $u_i + v_j + c_{ij} = \bar{c}_{ij}$, $i = 1,2,\dots,m; j = 1,2,\dots,m$. Так как общее число чисел c_{ij} равно mn, то, в принципе, за один такт можно выполнять mn сложений, то есть весь шаг вычислений можно реализовать за 2 такта. При этом потребуется mn машин ВС. При числе машин в ВС, равном ℓ, где $\ell = m/k$, общее число тактов на выполнение операции шага 2б составит $2kn$.

Операции шага 2в сводятся к операциям сравнения с нулем. Здесь также, в принципе, допускается одновременное выполнение операций над всеми числами c_{ij}, т.е. mn операций могут быть выполнены за один такт. В случае ℓ машин в ВС требуется kn тактов.

Операции шага 3 сводятся к отысканию в матрице наименьшего числа c_{ij} среди mn чисел. Сперва в каждой из ℓ машин выполняются mn/ℓ сравнений, затем для сравнения результатов во всех машинах потребуется еще ℓ операций сравнения. Если вести одновременное попарное сравнение, то это число может быть уменьшено до S, равного округленному до большего целого $\log_2 \ell$. Таким образом, общее число операций на выполнение шага 3 составит $mn/\ell + S$. Отметим, что в случае необходимости шаги I, 4, 5, 2а могут быть также ускорены за счет параллельного выполнения операций, однако это ускорение не является эффективным. Заметим, что при достаточно больших значениях m и n по сравнению с ℓ доля затрат времени на выполнение шагов I, 2а, 4,5 составит

$$a_1(m+n)/(a_2 mn/\ell) \approx \ell/m + \ell/n, \qquad (2.42)$$

где a_1 и a_2 – суммарное число операций. Если $\ell = 0{,}01m$, то затраты времени на шаги I,2а,4,5 составят около I%.

Операторная схема ρ-алгоритма может быть составлена по

аналогии с предыдущими задачами, и мы ее приводить не будем.

Таким образом, и для транспортной задачи применение УВС из ℓ ЭМ дает выигрыш во времени по сравнению с одной ЭМ примерно в ℓ раз.

Итак, решение экстремальных задач линейного программирования может эффективно выполняться на УВС. При этом выигрыш во времени в довольно широких пределах пропорционален числу машин в системе при значениях коэффициента эффективности δ, близких к единице.

§ 3. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ И ДИФФЕРЕНЦИРОВАНИЕ

В данном параграфе будет рассмотрено решение на УВС весьма распространенных задач математического анализа – интегрирования и дифференцирования функций.

I°. Численное интегрирование

По определению интеграла

$$\int_c^d f(x)dx = \lim_{\substack{n\to\infty \\ \Delta x_i \to 0}} \left(\sum_{i=1}^{n-1} f(x_i)\Delta x_i \right) \qquad (3.1)$$

откуда следует, что с помощью интегральной суммы $S_n = \sum_{i=1}^{n} f(x_i)\Delta x_i$ можно найти интеграл с любой степенью точности, хотя сходимость S_n к $\int_c^d f(x)dx$ может быть очень медленной.

Для получения высокой точности приходится брать n весьма больших порядков, особенно для сложных функций. При численном интегрировании $f(x)$ заменяется интерполяционной функцией $y(x)$.

Рассмотрим, например, вычисление интегралов $\int_c^d f(x)dx$ по формулам Ньютона-Котеса (см., например, [I7]), которые получаются путем замены $f(x)$ интерполяционным полиномом Лагранжа с узлами, разбивающими интервал интегрирования на равные части длиной h.

Для случая, когда узлы интегрирования содержат точки c и d (формулы замкнутого типа),

$$x_i = a + ih \quad (i = 1,2,\dots,n), \qquad (3.2)$$
$$c = a + h, \quad d = a + nh;$$

обозначим $F(y) = f(a + hy)$.

Тогда

$$\int_c^d f(x)dx = h\int_1^n F(y)dy =$$

$$= (d-c)\sum_{i=1}^{n} I_i^{(n)} f(a+ih) + h\int_1^n (y-1)(y-2)\dots(y-n)\times$$

$$\times F(y,1,2,\dots,n)\,dy, \qquad (3.3)$$

где $$I_i^{(n)} = \frac{(-1)^{n-i}}{(n-1)(i-1)!(n-i)!}\int_1^n \frac{(y-1)(y-2)\dots(y-n)}{y-i}dy. \qquad (3.4)$$

Значения $I_i^{(n)}$ не зависят от интервала интегрирования и могут быть вычислены раз и навсегда.

Отрезок интегрирования может быть разделен на части, и для каждой из них может быть применена формула (3.3), после чего результаты по всем частям суммируются. Этим можно воспользоваться для параллельного выполнения численного интегрирования с помощью УВС. Если число ЭМ в УВС равно l, то отрезок интегрирования может быть разделен на l частей. В этом случае на решение задач потребуется число тактов:

$$(N-l)/l + S, \qquad (3.5)$$

где N - число тактов, требующихся на решение задачи на одной ЭМ, S - округленный до большего целого $\log_2 l$, первое слагаемое соответствует интегрированию каждой из частей, а второе - суммированию результатов этого интегрирования.

Если $N \gg S$, то общий выигрыш во времени при применении системы можно полагать пропорциональным числу машин в системе. В принципе число ЭМ можно увеличивать до n, однако при l, близких к N, коэффициент использования ЭМ ϑ может заметно отличаться от оптимального из-за увеличения доли второго слагаемого в (3.5).

Указанные соображения справедливы и при применении других формул численного интегрирования, например, обобщенной формулы трапеций, обобщенной формулы Стирлинга, которые получаются из (3.3) при n = 2 и 3, соответственно, а также для случая, когда узлы интегрирования не содержат точек c и d (формулы открытого типа). Особенно важную роль УВС могут играть при вычислении кратных интегралов, вычисление которых требует выполнения огромного числа операций, часто превышающих возможности существующих ЭВМ. Существующие методы вычисления кратных

интегралов, например, метод кубатурных формул [18], позволяют легко распараллелить процесс вычислений на большое число однородных ветвей.

2°. Численное дифференцирование

Пусть $f(x)$ - функция, заданная таблично либо с помощью очень сложного аналитического выражения. Тогда для нахождения ее производной можно заменить функцию $f(x)$ интерполирующей функцией $\varphi(x)$, производная от которой может быть легко вычислена, т.е.

$$f(x) = \varphi(x) + R(x), \qquad (3.6)$$

где $R(x)$ - остаточный член интерполяционной формулы. Очевидно, что в том случае, когда $f(x)$ и $\varphi(x)$ имеют производную k-го порядка, дифференцируя (3.6) k раз, мы получим

$$f_{(x)}^{(k)} = \varphi_{(x)}^{(k)} + R_{(x)}^{(k)}. \qquad (3.7)$$

Рассмотрим в качестве примеров численное дифференцирование для неравноотстоящих узлов интерполирования и для равноотстоящих узлов. Для неравноотстоящих узлов воспользуемся интерполяционной формулой Ньютона для неравных промежутков.

$$f(x) = f(x_0) + (x-x_0)f(x_0,x_1) + (x-x_0)(x-x_1)f(x_0,x_1,x_2) + \dots$$

$$\dots + (x-x_0)(x-x_1)\dots(x-x_{n-1})f(x_0,x_1,\dots,x_n) +$$

$$+ (x-x_0)(x-x_1)\dots(x-x_n)f(x,x_0,x_1,\dots,x_n). \qquad (3.8)$$

Дифференцируя (3.8) k раз ($k<n$), получаем приближенное значение для k-й производной.

$$f^{(k)}(x) = k!\,[f(x_0,x_1,\dots,x_k) + (\alpha_0+\alpha_1+\dots+\alpha_k)f(x_0,x_1,\dots,x_{k+1}) +$$

$$+ (\alpha_0\alpha_1+\alpha_0\alpha_2+\dots+\alpha_k\alpha_{k+1})f(x_0,x_1,\dots,x_{k+2}) + \dots + (\alpha_0\alpha_1\dots\alpha_{n-k}+\dots$$

$$\dots+\alpha_{k+1}\alpha_{k+2}\dots\alpha_{n-1})f(x_0,x_1,\dots,x_n)], \qquad (3.9)$$

где $\alpha_i = x - x_i$. Остаточный член

$$R = \sum_{i=0}^{n} \frac{k!}{(k-i)(n+i+1)!} f^{(n+i+1)}(\xi_i)\cdot\omega_n^{(k-i)}(x), \qquad (3.10)$$

где ξ_i - некоторые точки, заключенные в интервале между наибольшим и наименьшим из чисел $x, x_0, x_1, \dots, x_n$.

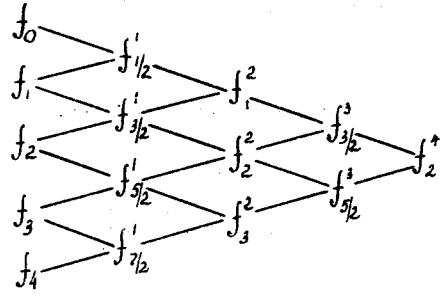
Для равноотстоящих узлов воспользуемся интерполяционной формулой Ньютона для интерполирования вперед:

$$f(x) = (x_0+ht) = f_0 + t f'_{1/2} + \frac{t(t-1)}{2!} f^2_1 + \frac{t(t-1)(t-2)}{3!} f^3_{3/2} + \ldots + \frac{t(t-1)\ldots(t-n+1)}{n!} f^n_{n/2}, \quad (3.11)$$

где $t=(x-x_0)/h$, $h=x_i-x_{i-1}$ $(i=1,2,\ldots,n)$, а f с верхним и нижним индексом – конечные разности:

$$\left.\begin{array}{l} f^0_{1/2}=\frac{f_0+f_1}{2};\ f'_1=\frac{f'_{1/2}+f'_{3/2}}{2};\ f^2_{3/2}=\frac{f^2_1+f^2_2}{2}; \\ f^0_{3/2}=\frac{f_1+f_2}{2};\ f'_2=\frac{f'_{3/2}+f'_{5/2}}{2};\ f^2_{5/2}=\frac{f^2_2+f^2_3}{2}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f^0_{i+1/2}=\frac{f'_i+f'_{i+1}}{2};\ f'_i=\frac{f'_{i-1/2}+f'_{i+1/2}}{2};\ f^2_{i+1/2}=\frac{f^2_i+f^2_{i+1}}{2} \end{array}\right\}. \quad (3.12)$$

Вычисление конечных разностей можно вести по следующей схеме:



(3.13)

Формула для вычисления κ -ой производной весьма сложна, и для ее запоминания обычно пользуются операторно-символической формулой

$$\left(h\frac{d}{dx}\right)^\kappa f(x_0) = \left\{\ell n(1+\Delta)\right\}^\kappa f(x_0). \quad (3.14)$$

Формальное разложение (3.14) $\ell n(1+\Delta)=\Delta-\Delta^2/2+\Delta^3/3-\ldots$, доведенное до постоянных разностей, формально возводится в степень как многочлен. Под Δ понимается оператор вычисления разности первого порядка, Δ^2 – второго и т.д.

Применять вычислительную систему для численного дифференцирования выгодно, если нужно вычислить значение производной во многих точках. Тогда можно применить число ЭМ, равное числу точек, и вычислить каждую точку на своей ЭМ либо, если число точек больше числа ЭМ, распределить их равномерно между ЭМ. В этом случае выигрыш во времени будет пропорционален числу ЭМ.

Применение УВС для нахождения значения производной в отдельной точке может быть оправдано для очень сложных функций. В этом случае, по-видимому, трудно добиться полной загрузки всех ЭМ. При вычислениях по формуле (3.14) значительное время тратится на отыскание конечных разностей, так как коэффициенты при них могут быть вычислены раз и навсегда. Как видно из (3.13), число разностей i -го порядка равно $n-i+1$ $(i=1,2,\ldots,n)$. Таким образом, если при вычислении разностей использовать n машин, то коэффициент эффективности δ будет равен примерно 0,5 и будет улучшаться при уменьшении числа ЭМ.

При использовании формулы (3.9) процесс вычислений также может быть распараллелен за счет одновременного вычисления как функций, так и коэффициентов при них. В этом случае коэффициент использования машин будет не очень высок. Однако, по-видимому, имеются возможности повышения этого коэффициента за счет усложнения программирования.

§ 4. Решение обыкновенных дифференциальных уравнений

Обыкновенные дифференциальные уравнения встречаются довольно часто в различных прикладных задачах.

Решение систем обыкновенных дифференциальных уравнений при достаточно высоком порядке системы и наличии сложных нелинейных зависимостей требует, как правило, выполнения большого числа операций. В данном параграфе будет рассмотрено решение на УВС задачи Коши для систем обыкновенных дифференциальных уравнений методом Рунге-Кутта, решение граничных задач для систем линейных и нелинейных дифференциальных уравнений [16], [17].

1°. Решение задачи Коши для системы дифференциальных уравнений методом Рунге-Кутта

Система дифференциальных уравнений может быть приведена к нормальной форме Коши:

$$\left.\begin{aligned} y_1' &= f_1(t, y_1, y_2, \dots, y_m);\\ y_2' &= f_2(t, y_1, y_2, \dots, y_m);\\ &\dots\dots\dots\dots\dots\\ y_m' &= f_m(t, y_1, y_2, \dots, y_m); \end{aligned}\right\} \qquad (4.1)$$

при начальных условиях

$$t = t_0,\ y_1 = y_{10}, \dots, y_m = y_{m_0};$$

$f_1, f_2, \dots, f_m$ – известные в общем случае нелинейные функции $t, y_1, \dots, y_m$. Если t рассматривать как зависимую переменную $y_{m+1} = y_n = t$ и добавить к системе уравнений (4.1) уравнение $y_n' = 1$, то система примет вид:

$$\left.\begin{aligned} y_1' &= f_1(y_1, \dots, y_n),\\ y_2' &= f_2(y_1, \dots, y_n),\\ &\dots\dots\dots\dots\\ y_n' &= f_n(y_1, \dots, y_n), \end{aligned}\right\} \qquad (4.2)$$

где $n = m+1$, $y_n = t$, $f_n(y_1, \dots, y_n) = 1$.

Согласно методу Рунге-Кутта решение задачи Коши для системы дифференциальных уравнений (4.2) производится последовательными шагами. Значения функций на $(s+1)$-м шаге при $t = t_{s+1}$ вычисляем по формулам:

$$k_{i1}^s = h f_i(y_1^s, y_2^s, \dots, y_n^s), \qquad (4.3)$$

$$k_{ji}^s = h f_i(y_1^s + k_{j-1,1}^s, y_2^s + k_{j-1,2}^s, \dots, y_n^s + k_{j-1,n}^s),\quad j = 2,3,4;\ i = 1,2,\dots,n \qquad (4.4)$$

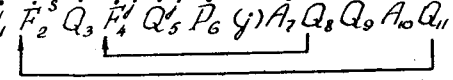
h – **константа (1/2 длины шага интегрирования);**

$$\Delta y_i^s = \frac{1}{3}(k_{1i}^s + 2k_{2i}^s + 2k_{3i}^s + k_{4i}^s)\quad i = 1,2,\dots,n; \qquad (4.5)$$

$$y_i^{s+1} = y_i^s + \Delta y_i^s. \qquad (4.6)$$

Нетрудно видеть, что каждая последующая формула использует результаты предыдущей и счет по каждой из них ведется n раз. Поэтому данную задачу естественно решать на УВС из n ЭМ, причем в каждой машине m_i производить вычисления и хранить переменные, соответствующие значению параметра i.

Схема ρ -алгоритма в этом случае может быть записана в виде следующей строчки:



$$\dot{H}_1 \dot{F}_2^s \dot{Q}_3 \dot{F}_4' \dot{Q}_5' \dot{P}_6 (y) \dot{A}_7 \dot{Q}_8 \dot{Q}_9 \dot{A}_{10} \dot{Q}_{11}(s) \dot{A}_{12}, \qquad (4.7)$$

где

$\dot{H}_1$ – ρ -оператор настройки, соединяющий входы и выходы всех ЭМ с общим каналом связи;

$\dot{F}_2^s$ – ρ – оператор, формирующий переход к новому шагу;

$\dot{Q}_3$ – обобщенный ρ -оператор обмена, передающий код y_i^s, находящийся в ЭМ m_i, во все ЭМ $(i = 1,2,\dots,n)$;

$\dot{F}_4'$ – ρ -оператор, формирующий переход к вычислению следующих k_{ji}^s;

$\dot{Q}_5'$ – обобщенный ρ -оператор, выполняющий вычисление функций $h f_i(z_{j1}^s, z_{j2}^s, \dots, z_{jn}^s)$;

$\dot{P}_6$ – ρ -оператор обобщенного условного перехода, проверяющий, все ли k_{ji}^s вычислены, если все, то осуществляет переход к $\dot{Q}_9$ ρ -оператору, если нет, то – к следующему ρ -оператору;

$\dot{A}_7$ – арифметический ρ -оператор, согласно которому в каждой ЭМ m_i вычисляются суммы $z_{ji}^s = y_i^s + k_{j-1,i}^s$;

$\dot{Q}_8$ – обобщенный ρ -оператор обмена, передающий во все ЭМ переменные z_{ji}^s, каждая из которых находится в ЭМ с соответствующим номером i. (Этот ρ -оператор выполняется за n тактов; его выполнение может быть совмещено с выполнением оператора $\dot{Q}_5$);

$\dot{Q}_9$ – обоощенный ρ -оператор, вычисляющий во всех ЭМ Δy_i^s;

$\dot{A}_{10}$ – арифметический ρ -оператор, вычисляющий во всех ЭМ y_i^{s+1};

$\dot{Q}_{11}(S)$ – обобщенный ρ -оператор, устанавливающий конец вычислений;

$\dot{A}_{12}$ – ρ -оператор конца.

Как видно из схемы ρ -алгоритма, процесс вычислений на УВС состоит из параллельного счета всех машин и обмена информацией между машинами. По сравнению с обычной ЭВМ получается выигрыш во времени решения примерно в n раз. Потери времени на обмен информацией между ЭМ пренебрежимо малы.

2°. Решение граничных задач для систем линейных дифференциальных уравнений методом сопряженных уравнений

Пусть имеется система n линейных дифференциальных уравнений первого порядка:

$$y'(t) = A(t)\,y(t) + F(t), \qquad (4.8)$$

где $y'(t)$ – матрица – столбец

$$\begin{bmatrix} y_1'(t) \\ \vdots \\ y_n'(t) \end{bmatrix}$$

$y(t)$ – матрица – столбец из неизвестных

$$\begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix}$$

$A(t)$ – квадратная матрица $(n \times n)$ из коэффициентов, которые могут быть функциями независимой переменной t, но не зависят от y_i.

$F(t)$ – матрица-столбец свободных членов

$$\begin{bmatrix} f_1(t) \\ \vdots \\ f_n(t) \end{bmatrix}$$

Предположим, что первые r из y_i заданы при $t=0$ и что $n-r$ из y_i заданы при $t=T$. Обозначим их значения через $y_{i_\kappa}(T)$ где $\kappa = 1,2,\ldots,n-r$. Так как условия заданы для различных значений t, то граничная задача не может быть непосредственно решена методом Рунге-Кутта, и ее сперва сводят к задаче Коши. Применим для этого метод сопряженных уравнений. Идея этого метода заключается в следующем.

Сначала определяется система дифференциальных уравнений, сопряженная с системой (4.8)

$$-x'(t) = \bar{A}(t)\,x(t), \qquad (4.9)$$

где $\bar{A}(t)$ – транспонированная матрица матрицы $A(t)$, а $x(t)$ – новая матрица-столбец из неизвестных:

$$\begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

Можно показать, что

$$\frac{d}{dt}(x_i y_i) = x_i f_i. \qquad (4.10)$$

Интегрируя (4.10) от 0 до T, получаем

$$x_i(T)\,y_i(T) - x_i(0)\,y_i(0) = \int_0^T x_i(t) f_i(t)\,dt. \qquad (4.11)$$

Для того, чтобы свести задачу к начальной, надо найти значения при $t=0$ $n-r$ переменных, не заданных в этой точке $y_{r+1}(0),\ldots,y_n(0)$. Для их вычисления нужны $n-r$ уравнений (4.11). Метод определения $y_{r+1}(0),\ldots,y_n(0)$ состоит в следующем.

Сопряженная система уравнений (4.9) интегрируется $n-r$ раз от T до 0. При этом используются подобранные начальные значения $x_i(T)$, которые специально подбираются так, что при m-ом интегрировании $(1 \le m \le n-r)$

$$\left.\begin{aligned} x_i(T) &= 0 \quad \text{для } i \neq i_m, \\ x_{i_m}(T) &= 1 \end{aligned}\right\}, \qquad (4.12)$$

где $x_{i_m}(T)$ – коэффициент в формуле (4.11) при одном из $n-r$, заданных $y_{i_m}(T)$. В результате каждого интегрирования системы (4.9) получаем значения соответствующих значений ${}_m x_i(t)$, где параметр m указывает номер интегрирования, что может быть записано в виде матрицы столбца ${}_m X(t)$.

Из условий (4.12) вытекает, что все $x_i(T)$ равны нулю, кроме одного $x_{i_m}(T)$, которое для удобства выбрано равным единице. Уравнение (4.11) принимает вид

$${}_m x_i(0)\,y_i(0) - y_{i_m}(T) = \int_0^T {}_m x_i(t) f_i(t)\,dt. \qquad (4.13)$$

Полагая m равным последовательно $1,2,\ldots,n-r$, получаем систему $n-r$ линейных алгебраических уравнений для неизвестных $y_{r+1}(0),\ldots,y_n(0)$. После решения этой системы становятся известными значения всех переменных в одной и той же точке $t=0$, и исходная задача может рассматриваться как начальная. Решение этой задачи на вычислительной системе из n машин можно представить следующим образом.

I ЭТАП. Вычисление $y_{r+1}(0),\ldots,y_n(0)$.

Ш а г I. Интегрирование сопряженной системы (4.9) $n-r$ раз от T до 0 при условиях (4.12). В общем виде, если коэффициенты матрицы $\bar{A}(t)$ являются функциями времени t, заданными таблично, решение сопряженной си-

стемы сводится к решению системы конечно-разностных уравнений при данных начальных условиях. Для общности будем предполагать также, что коэффициенты матрицы-столбца $\bar{F}(t)$ также заданы таблично.

Разместим в каждой машине m_i i -ю строку матрицы $\bar{A}(t)$, а также $x_i(t)$ и $f_i(t)$.

Вычисление $x_i(t)$ для $i=1,2,\dots,r$ в вычислительной системе при m -м интегрировании системы (4.9) будет происходить следующим образом.

Система дифференциальных уравнений (4.9) при начальных значениях (4.I2) может быть преобразована по следующей схеме.

На $(k+1)$ -м шаге (при $t=t_{k+1}$) сначала вычисляются приращения $\Delta x_i^{(k)} = x_i^{(k+1)} - x_i^{(k)}$ по формуле

$$\Delta x_i^{(k)} = (a_{i1}^{(k)} x_1^{(k)} + a_{i2}^{(k)} x_2^{(k)} + \dots + a_{in}^{(k)} x_n^{(k)}) \Delta t$$

$$i = 1,2,\dots,r, \qquad (4.I4)$$

где $a_{ij}^{(k)}$ - элементы i -ой строки матрицы $\bar{A}$ при $t=t_k$.

Затем вычисляются значения переменных

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k)} \qquad (4.I5)$$

Первый шаг повторяется $n-r$ раз. Вычисленные значения $x_{i_m}^{(k)}$ запоминаются в машинах с номерами i_m.

Ш а г 2. Вычисление правых частей выражения (4.I3) по формуле $b_{i_m} = \int_m x_i(t) f_i(t) dt$. В машинах с номерами, соответствующими индексам i_m $(1 \le m \le n-r)$, одновременно вычисляются интегралы по формулам численного интегрирования.

Ш а г 3. Определение $y_{r+1}(0),\dots,y_n(0)$.
В машинах, соответствующих номерам i_m $(1 \le m \le n-r)$, определяются $y_{r+1}(0),\dots,y_n(0)$ по формуле:

$$m\, x_i(0)\, y_i(0) - y_{i_m}(T) = b_{i_m} . \qquad (4.I6)$$

ЭТАП II. Решение системы уравнений (4.8).

Решение системы уравнений (4.8) может осуществляться методом Рунге-Кутта, так как все начальные условия определены при одном и том же значении t.

Основное время решения данной задачи приходится на первый шаг I этапа и II этап.

При их выполнении коэффициент использования ЭМ системы близок к единице. На остальных шагах I этапа использование ЭМ

несколько хуже, однако на эти шаги приходятся ничтожные затраты в общем балансе времени. Кроме того, существуют дополнительные возможности по их распараллеливанию. Поэтому можно считать, что УВС из r машин дает выигрыш во времени по сравнению с одной ЭМ примерно в r раз.

3°. Решение граничных задач для нелинейных систем

Общая нелинейная система имеет вид:

$$y_i'(t) = g_i(y_1,\dots,y_n,t), \quad i=1,\dots,n. \qquad (4.I7)$$

Предполагается, что все функции g_i дифференцируемы по каждому из y_i и что граничные условия определяют $y_1(0),\dots,y_r(0)$ и $y_{i_k}(T)$, где $k=1,\dots,n-r$. Применяемый для нелинейных уравнений метод состоит в нахождении приближенных значений $y_{r+1}(0),\dots,y_n(0)$. Зададимся исходными значениями $y_{r+1}^*(0),\dots,y_n^*(0)$. После этого задача может быть решена как начальная для системы уравнений (4.I7). Компоненты решения, получаемые с помощью приближенных начальных условий, обозначаются $y_i^*(t)$. Так как вычисленные таким образом значения $y_{i_k}(T)$ не совпадают с заданными, то в дальнейшем необходимо использовать такие поправки начальных приближенных значений, чтобы все величины $|y_{i_k}(T) - y_{i_k}^*(T)|$ получились возможно более малыми. Определим $\delta y_i(t)$ посредством равенства

$$\delta y_i(t) = y_i(t) - y_i^*(t), \quad i=1,2,\dots,n. \qquad (4.I8)$$

Если подставить найденные из этих равенств $y_i(t)$ в систему уравнений (4.I7), учесть, что $y_i^*(t)$ является решением этой же системы при некоторых граничных условиях, и пренебречь членами порядка выше первого, то получим

$$\delta y_i'(t) = \left(\frac{\partial g_i}{\partial y_j}\right) \delta y_i , \qquad (4.I9)$$

где $\delta y_i'(t) = \frac{d}{dt}[\delta y_i(t)]$.
Уравнения (4.I9) представляют собой однородную систему линейных дифференциальных уравнений, сопряженная система для которой имеет вид

$$-x_i' = \left(\frac{\partial g_j}{\partial y_i}\right) x_j . \qquad (4.20)$$

При этом формула (4.II) будет

$$x_i(T)\,\delta y_i(T) - x_i(0)\,\delta y_i(0) = 0. \qquad (4.2I)$$

В этих уравнениях $\delta y_i(0) \equiv 0$ для $i = 1, \ldots, r$, а остальные $\delta y_i(0)$ представляют собой $n-r$ неизвестных величин. Чтобы получить $n-r$ уравнений для этих неизвестных, необходимо решить для уравнений (4.20) $n-r$ начальных задач. Решение m-й задачи есть вектор ${}_m X(t)$ с компонентами ${}_m x_i(t)$. При отыскании m-го решения все значения $x_i(T)$ принимаются равными нулю, кроме коэффициента при $y_{i_m}(T)$, который выбирается равным единице.

Таким образом, уравнения (4.21) приводятся к виду

$$\delta y_{i_m}(T) = {}_m x_{r+i}(0)\,\delta y_{r+i}(0) \qquad (4.22)$$
$$i = 1, 2, \ldots, n-r$$

Решая уравнения (4.22), находим поправки $\delta y_{r+i}(0)$, которые вместе с приближенными значениями $y^*_{r+i}(0)$ позволяют вычислить уточненные приближения. Процесс повторяется до тех пор, пока начальные условия не обеспечат достаточно малых значений погрешностей

$$|y_{i_K}(T) - y^*_{i_K}(T)| < \varepsilon. \qquad (4.23)$$

Решение этой задачи на вычислительной системе из n машин происходит следующим образом.

ЭТАП I. По полученным приближенным значениям

$$y^*_{r+1}(0), \ldots, y^*_n(0)$$

вычисляются $y^*_i(t)$.

Ш а г 1. Решается задача Коши для системы из n дифференциальных уравнений. Решение системы дифференциальных уравнений на ВС описано выше.

Ш а г 2. По найденным значениям проверяется условие точности результатов (4.23). Если для всех $K = 1, \ldots, (n-r)$ оно выполнено, то вычисления окончены, в противном случае осуществляется переход ко II этапу.

ЭТАП П. Вычисление поправок δy_{i_m} $m = 1, \ldots, n-r$.

Ш а г 1. Решается $n-r$ начальных задач для сопряженной системы однородных линейных уравнений.

$$-x_i' = \left(\frac{\partial g_i}{\partial y_i}\right) x_j .$$

Решение на ВС производится так, как это описано выше для системы однородных линейных уравнений.

Ш а г 2. По найденным значениям x_j из уравнения (4.21) определяются $\delta y_{r+i}(0)$ $(i = 1, \ldots, n-r)$.

В каждой из $n-r$ вычислительных машин находится $\delta y_{r+i}(0)$.

Ш а г 3. По исходным значениям $y^*_{r+i}(0)$ и $\delta y_{r+i}(0)$ вычисляются $y_{r+i}(0) = y^*_{r+i}(0) + \delta y_{r+i}(0)$ в $n-r$ машинах уточненные приближения. После чего совершается переход к этапу I.

В данной задаче наиболее трудоемкие вычисления (на этапе I и шаге I этапа П) могут одновременно выполняться на r ЭМ при ничтожном простое ЭМ. На шагах 2 и 3 этапа П $n-r$ ЭМ используются полностью. Чтобы использовать остальные r ЭМ, надо применить более сложную схему распараллеливания. Но даже их простой мало повлияет на общий коэффициент использования машин, вследствие небольшого числа вычислений на этих шагах.

Таким образом, в этой задаче, как и в других, связанных с решением систем обыкновенных дифференциальных уравнений, выигрыш во времени по сравнению с одной ЭМ при применении УВС из n ЭМ будет примерно равен n.

§ 5. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ

В качестве примеров рассмотрим решение на УВС линейных дифференциальных уравнений в частных производных второго порядка с двумя независимыми переменными. Общее уравнение такого рода имеет вид:

$$H\frac{\partial^2 u}{\partial x^2} + 2K\frac{\partial^2 u}{\partial x \partial y} + L\frac{\partial^2 u}{\partial y^2} + M = 0, \qquad (5.1)$$

где H, K, L - функции x, y, u; M - функция $x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}$.

Различают три типа таких уравнений: эллиптическое $(K^2 - HL < 0)$, параболическое $(K^2 - HL = 0)$ и гиперболическое $(K^2 - HL > 0)$. При решении на УВС уравнений всех трех типов, ограничимся такими случаями, когда тип уравнения не изменяется при переходе от одной точки области к другой.

I°. Решение задачи Дирихле для эллиптических уравнений итеративным методом (см. например, [I6], [I7])

Рассмотрим в качестве примера решение уравнения Пуассона

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = g(x, y) \qquad (5.2)$$

для прямоугольной области $0 \le x \le a$, $0 \le y \le b$; на границе заданы значения функции u. Разобьем область на квадраты с помощью сетки с шагом h. Обозначим через $u^{s}_{j,k}$ и $u^{s+1}_{j,k}$ значения функций в узловой внутренней точке (j,k) на s-м и $(s+1)$-м последовательных приближениях. Для вычисления $u^{s+1}_{j,k}$ воспользуемся формулой

$$u^{s+1}_{j,k} = \frac{1}{4}(u^{s}_{j-1,k} + u^{s}_{j+1,k} + u^{s}_{j,k-1} + u^{s}_{j,k+1} - h^2 q_{j,k}) . \quad (5.3)$$

Процесс решения данным методом весьма прост. Выбираем исходное приближение. Последовательные приближения вычисляются по формуле (5.3). Затем для всех точек производится сравнение нового и предыдущего значений функции.

Если хотя бы в одной точке не удовлетворяется условие

$$|u^{s+1}_{j,k} - u^{s}_{j,k}| < \varepsilon , \quad (5.4)$$

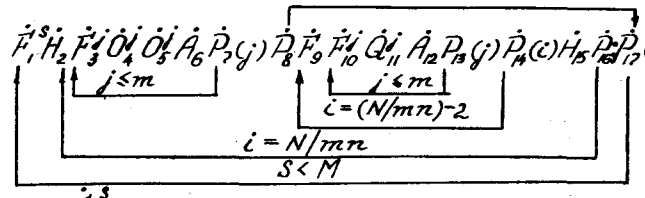
то производится следующая итерация и т.д., пока не будет удовлетворено условие (5.4).

Вычисления по формулам (5.3) и (5.4) можно производить сразу для всех N точек области. Поэтому базовая точка на характеристической функции ρ-алгоритма данной задачи будет соответствовать количеству ЭМ, равном N, которое обычно велико, и данная задача практически не накладывает ограничений на число используемых ЭМ в системе.

Пусть число ЭМ будет $n < N$. Тогда имеет смысл разбить область на n полос (вертикальных или горизонтальных) и значения функций в узловых точках каждой i-й полосы вычислять на ЭМ m_i $(i = 1,2,\ldots,n)$. При этом в памяти каждой ЭМ m_i будут храниться значения функции для N/n узловых точек, то есть всех точек, находящихся внутри каждой полосы; данные, необходимые для вычисления функции, будут находиться в памяти той же ЭМ. Для точек, находящихся на границе с соседней полосой, один из аргументов будет находиться в соседней ЭМ. Если в каждой полосе будет $2m$ пограничных точек, кроме I-й и n-й, где их будет m, то каждая ЭМ должна будет на данном шаге итерации обмениваться со своими соседями $2m$ кодами. Чтобы крайние ЭМ не составляли исключения из этого правила и имели ту же программу работы, можно правые граничные точки области поместить в ЭМ m_1, а левые - в ЭМ m_n.

Будем производить вычисление значений функции в каждой полосе по колонкам; сначала I-й, состоящей из пограничных точек, затем $(N/nm)-2$ колонок с внутренними точками, и, наконец, (N/nm)-ю колонку, состоящую из пограничных точек.

Схема ρ-алгоритма может быть записана в виде, где все ρ-операторы состоят из n одинаковых составляющих.



где $\dot{F}^{s}_{1}$ — ρ-оператор, формирующий переход к новой итерации;

$\dot{H}_2, \dot{H}_{15}$ — ρ-операторы настройки, соединяющие соответственно вход ЭМ m_i с выходом ЭМ m_{i-1} и вход ЭМ m_i с выходом ЭМ m_{i+1} $(i = 1,2,\ldots,n)$;

$\dot{F}^{i}_{3}, \dot{F}^{i}_{10}$ — ρ-операторы, формирующие переход от j-й точки колонки к $(j+1)$-ой;

$\dot{Q}^{i}_{4}$ — ρ-оператор обмена информацией, который посылает код в канал связи и принимает код из канала связи;

$\dot{Q}^{i}_{5}, \dot{Q}^{i}_{11}$ — обобщенные арифметические ρ-операторы, ведущие счет по формуле (5.3);

$\dot{A}_6, \dot{A}_{12}$ — арифметические ρ-операторы, ведущие счет по формуле (5.4);

$\dot{P}_7(j), \dot{P}_{13}(j)$-$\rho$-операторы обобщенного условного перехода, определяющие цикл счета точек в одной колонке;

$\dot{P}_8$ — ρ — оператор обобщенного условного перехода, определяющий, все ли колонки просчитаны;

$\dot{F}^{i}_{9}$ — ρ — оператор, формирующий переход от одной колонки к другой;

$\dot{P}_{14}(i)$-ρ-оператор обобщенного условного перехода, определяющий цикл счета по колонкам;

$\dot{P}_{16}$-ρ-оператор обобщенного безусловного перехода;

$\dot{P}_{17}$-ρ-оператор обобщенного условного перехода, определяющий, удовлетворено ли неравенство (5.4) во всех ЭМ, и в зависимости от этого передающий уравнение ρ-оператору $\dot{F}^{s}_{1}$ либо $\dot{Я}_{18}$;

$\dot{Я}_{18}$-ρ-оператор конца работы.

2°. Решение краевой задачи для параболических уравнений

Рассмотрим решение типичного параболического уравнения – уравнения теплопроводности

$$\frac{\partial^2 u}{\partial x^2} = c\,\frac{\partial u}{\partial t}\ , \qquad (5.6)$$

удовлетворяющее условиям: 1) при $t=0$ u принимает заданные значения для всех x на отрезке $0 \le x \le L$ и 2) u принимает заданные значения при любых t в точках, где $x=0$ и $x=L$.

Рассмотрим метод решения уравнений (5.6), состоящий в замене производной второго порядка конечноразностным приближением (см., например, [I6], [I7]).

Обозначим $u(jh,t)$, $j=0,1,2,\ldots,N+1$, через $u_j(t)$, где h – длина шага в направлении оси x; при этом $(N+1)h=L$; $u_j(t)$ – функция только одной переменной t, функции $u_0(t)$ и $u_{N+1}(t)$ известны из граничных условий.

Уравнение (5.6) можно приближенно заменить уравнениями

$$\frac{du_j}{dt} = \frac{1}{ch^2}(u_{j+1} - 2u_j + u_{j-1}),\ j=1,2,\ldots,N, \qquad (5.7)$$

где u_j являются функциями только одной переменной t. Уравнения (5.7) образуют систему дифференциальных уравнений первого порядка с заданными значениями неизвестных при $t=0$. Решая эту систему, например, методом Рунге–Кутта, как было показано в § 4, можно определить все N неизвестных функций $u_j(t)$.

Данная система проще, приведенной в § 4, так как производная u_j' зависит только от трех переменных, причем одна переменная имеет тот же индекс j, а две другие – индекс, отличающийся на единицу. Это существенно упрощает вычисление правых частей, а также снижает требования к обмену информацией между ЭМ.

Как и в предыдущем случае, процесс решения данной задачи может быть представлен в виде N одинаковых (или почти одинаковых) параллельных ветвей вычислений. Наиболее просто процесс решения будет выглядеть на УВС из N машин.

На практике, по-видимому, число ЭМ n будет меньше N. Но в этом случае, как и в предыдущем, область может быть разделена на n частей и для пограничных точек также потребуется обмен кодами между соседними ЭМ.

Таким образом, при решении дифференциальных уравнений параболического типа выигрыш во времени будет пропорционален числу ЭМ, когда последнее изменяется в довольно широкой области.

3°. Решение задачи Коши для линейных дифференциальных уравнений гиперболического типа

В качестве примера рассмотрим решение на УВС задачи Коши методом сеток (см. [I6], [I7]) для простейшего дифференциального уравнения гиперболического типа:

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 0 \qquad (5.8)$$

с начальными условиями

$$u\Big|_{y=0} = \varphi(x)\ ,\ \frac{\partial u}{\partial y}\Big|_{y=0} = \psi(x)\ \ (a \le x \le b). \qquad (5.9)$$

Будем использовать прямоугольную сетку с шагами h по x и с ℓ по y.

$$\ell = \alpha h\ ,\quad \alpha = \frac{\ell}{h} \ge 1\ . \qquad (5.10)$$

Разностное уравнение имеет вид:

$$u_{i,j+1} = 2u_{ij} - u_{ij-1} + \alpha^2(u_{i+1,j} - 2u_{ij} + u_{i-1,j}). \qquad (5.11)$$

Зная значения решения в узлах первых двух горизонтальных рядов (определяемые из начальных условий), будем последовательно вычислять значения решения на втором, третьем и т.д. рядах (рис. 5).

Как видно из рис. 5, в каждом последующем ряду число вычисляемых узлов уменьшается на 2. При числе машин в УВС, равном $n < N$, отрезок $[a,b]$ можно разбить на n частей (для простоты примем, что все части равны). При этом в первой ЭМ будут находиться узлы $1,2,\ldots,\ell$ и $1^{(1)}, 2^{(1)}, \ldots, (\ell-1)^{(1)}$, во второй $\ell+1, \ell+2, \ldots, 2\ell$ и $\ell^{(1)}, (\ell+1)^{(1)}, \ldots, (2\ell-1)^{(1)}$ и т.д., наконец, в n-ой ЭМ $N-\ell+1, \ldots, N$ и $(N-\ell-1)^{(1)}, \ldots, (N-2)^{(1)}$ $(\ell = \frac{N}{n})$.

По формуле (5.11) в каждой ЭМ производится вычисление узлов последующего ряда. При вычислении узлов на границе между ЭМ производится двусторонний обмен информацией между соседними ЭМ. Число вычисляемых узлов в каждом последующем ряду уменьшается на два. В связи с этим загрузка ЭМ становится неравномерной и коэффициент эффективности $\delta \approx 0{,}5$. Значение δ может быть увеличено путем перераспределения узлов между ЭМ системы. Для этого перед вычислением 3-го ряда узлов из второй ЭМ пересылаются в первую узлы $\ell+1$, $\ell^{(1)}$, а из ЭМ $n-1$ в ЭМ n – узлы $N-\ell, (N-\ell-2)^{(1)}$. Перед вычислением 4-го ряда

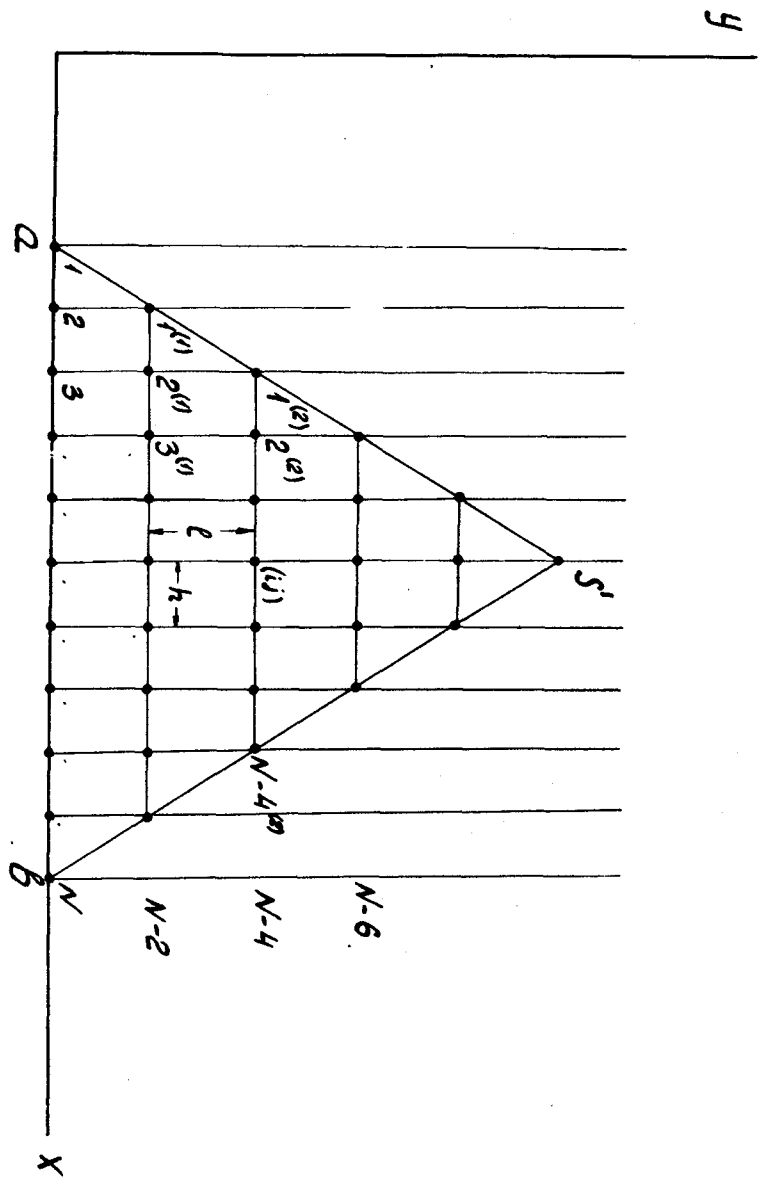


Рис.5.

производится пересылка соответствующей пары узлов из третьей во вторую ЭМ, из второй ЭМ в первую, аналогично производится пересылка из $(n-2)$-й ЭМ в $(n-1)$-ю ЭМ и из $(n-1)$-й ЭМ в n-ю и т.д. Указанный процесс заканчивается, когда пересылка узлов сделана из $n/2$-й ЭМ в $n/2+1$-ю при четном числе ЭМ в УВС и из $\frac{n+1}{2}$-ой ЭМ в $\frac{n+1}{2}+1$-ю – при нечетном. Вблизи точки S (рис. 5) вычисление можно продолжать на нескольких или даже одной ЭМ.

Из приведенного примера видно, что при $\ell \gg n$ простаивание ЭМ будет незначительным, а выигрыш во времени пропорционален числу ЭМ в системе.

Рассмотренные примеры дифференциальных уравнений в частных производных эллиптического, параболического и гиперболического типов и методы их решения являются типичными, поэтому решение подобных задач на УВС по сравнению с одной ЭВМ, эквивалентной по параметрам ЭМ системы, дает выигрыш во времени, пропорциональный числу ЭМ в системе при изменении числа ЭМ в широких пределах.

§ 6. РЕШЕНИЕ НЕКОТОРЫХ ЗАДАЧ ТЕОРИИ ВЕРОЯТНОСТЕЙ И МАТЕМАТИЧЕСКОЙ СТАТИСТИКИ

Рассмотрим реализацию на УВС методов статистических испытаний (метод Монте-Карло) и статистических решений.

I°. Метод статистических испытаний[x)]

Идея метода статистических испытаний состоит в том, что искомым величинам задачи ставятся в соответствие параметры некоторого случайного процесса, моделируемого на ЭВМ.

Общая математическая схема метода статистических испытаний описывается с помощью цепей Маркова. Цепью Маркова называется система S с конечным множеством состояний $\{S_1, S_2, \ldots, S_e\}$. В каждый момент времени $t=0,1,2,\ldots,n$ система S находится в одном определенном состоянии S_i, которому соответствует набор условных вероятностей $p_{i1}, p_{i2}, \ldots, p_{ie}$, где p_{ij} – вероятность того, что система, находящаяся в момент времени t в состоянии S_i, в момент $t+1$ перейдет в состояние S_j.

x) См., например, [19].

Особенность цепей Маркова состоит в том, что вероятность перехода определяется лишь исходным состоянием S_i и не зависит от предыстории системы. Совокупность всех условных вероятностей p_{ij} образует матрицу P, полностью определяющую свойства данной цепи.

Состояние S_i, для которого $p_{ij}=1$ при $i=j$ и $p_{ij}=0$ при $i \neq j$, называется о с о б ы м. Однажды попав в это состояние, система пребывает в нем сколь угодно долго.

Для практических приложений важную роль играют останавливающиеся цепи Маркова, у которых для каждого состояния S_i существует отличная от нуля вероятность перехода в особое состояние. Останавливающиеся цепи Маркова через конечное число шагов переходят в новое состояние.

Для большинства задач схема ее решения методом статистических испытаний выглядит следующим образом. Данной задаче ставится в соответствие останавливающаяся цепь Маркова, которая затем моделируется. Моделирование процесса последовательных переходов этой цепи протекает по схеме:

$$x \sim S_{i_0} \longrightarrow S_{i_1} \longrightarrow S_{i_2} \longrightarrow \ldots \longrightarrow S_{i_e}, \qquad (6.I)$$

где S_{i_0} - начальное состояние, а S_{i_e} - одно из особых состояний. При этом определяется значение некоторой функции $F(x)$, зависящей от последовательности переходов (6.I). Функция $F(x)$ является случайной величиной, математическое ожидание которой требуется найти. После того, как будет зафиксировано состояние $F(x)$, система S возвратится в состояние S_{i_0} и процесс переходов повторяется. После достаточно большого числа испытаний N получается сумма

$$\frac{1}{N}\sum_{x} F(x) \approx MF(x), \qquad (6.2)$$

взятая по всем реализованным последовательностям переходов (6.I). Полное время решения задачи на одной ЭВМ составляет

$$T \approx N(M\tau)T_0, \qquad (6.3)$$

где $M\tau$ - математическое ожидание числа переходов в последовательности (6.I);

T_0 - среднее время реализации переходов на ЭВМ.

Рассмотрим в качестве примера решение задачи прохождения однородного потока нейтронов через плоскую пластину H.

Пусть пластина H будет однородна и не содержит делящихся веществ. Нейтроны при взаимодействии с атомами пластинки

могут либо рассеиваться ими, либо поглощаться. Количественно эти процессы принято характеризовать сечениями Σ_s и Σ_c, соответственно. Их сумма (при отсутствии деления) равна полному сечению Σ_t. Отношения Σ_s/Σ_t и Σ_c/Σ_t являются вероятностями, соответственно, рассеяния и поглощения.

Рассмотрим теперь моделирование физических траекторий. Выберем ось OZ перпендикулярно к плоскости пластинки. Пусть поверхности пластины H, через которую поступают нейтроны, соответствует координате $Z=0$, а другой поверхности - $Z=h$. Состояние нейтрона при данных условиях характеризуется тремя величинами: координатой z, энергией E и направлением полета $\mu=\cos\theta$ (рис. 6). Начальное значение координаты для траекторий всегда постоянно $(z_0=0)$, а значения энергии E_0 и направление μ_0 зависят от свойств падающего потока. На практике часто приходится иметь дело с четырьмя случаями:

а) Моноэнергетический поток с заданным значением E_0.

б) Поток с заданным энергетическим спектром $n(E)$. Тогда значение E_0 р а з ы г р ы в а е т с я по формуле:

$$\int_{E_{min}}^{E_0} n(E)dE = \gamma \int_{E_{min}}^{E_{max}} n(E)dE, \qquad (6.4)$$

где $0 \leq \gamma \leq 1$ - случайное число.

в) Направленный поток с заданным значением μ_0.

г) Пространственный изотропный процесс. Значение μ_0 разыгрывается по формуле $\mu_0=\gamma$.

Знание начальных значений позволяет произвести дальнейший расчет траектории. Для этого

I) Разыгрываем координату n -го столкновения $(n=1,2,\ldots)$

$$z_n = z_{n-1} - \frac{\mu_{n-1}\ell n\,\gamma}{\Sigma_t(E_{n-1})}. \qquad (6.5)$$

2) Проверяем выполнение условий

$$0 \leq z_n \leq h. \qquad (6.6)$$

Если условия (6.6) не выполнены, то нейтрон покинул пластину. Это фиксируется, и затем разыгрывается следующая траектория.

3) Если $0 \leq z_n \leq h$, то находится очередное γ, если $\gamma \leq \Sigma_c/\Sigma_t$, то нейтрон считается поглощенным и его траектория на этом заканчивается, о чем делается запись.

4) Если $\gamma > \Sigma_c/\Sigma_t$, то считается, что нейтрон испытывает рассеяние и нужно разыграть его направление и энергию.

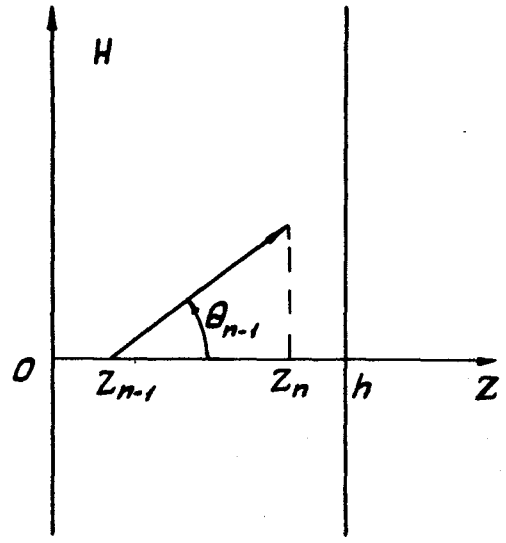


Рис.6.

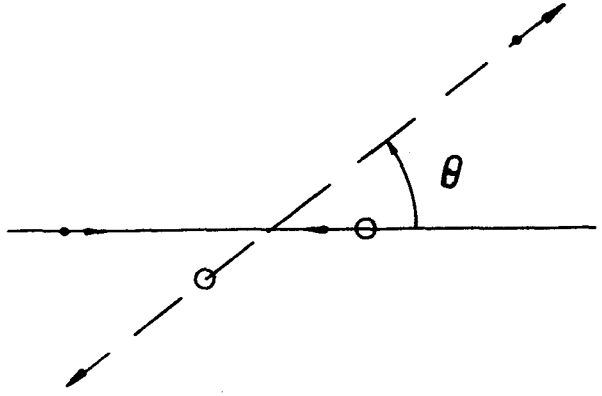


Рис.7.

В частности, если рассеяние упругое, то изменение энергии определяется направлением рассеяния. Пусть рассеяние нейтрона происходит с ядром, массовое число которого равно A. Такое рассеяние определяется двумя случайными величинами, в качестве которых обычно берут угол рассеяния θ в системе координат, связанной с центром масс пары нейтрон-ядро (рис. 7) и азимутальный угол рассеяния χ:

$$0 \le \theta < \pi, \quad 0 \le \chi \le 2\pi .$$

На основании законов сохранения импульса и энергии можно вычислить угол ψ, на который отклонился нейтрон от своего первоначального движения в системе координат, связанной с пластинкой

$$\cos\psi = \frac{A\cos\theta + 1}{\sqrt{A^2 + 2A\cos\theta + 1}}, \qquad (6.7)$$

а затем определить энергию, которую он сохраняет, и новое направление полета:

$$E_n = E_{n-1}\,\frac{A^2 + 2A\cos\theta + 1}{(A+1)^2}, \qquad (6.8)$$

$$\mu_n = \mu_{n-1}\cos\psi + \cos\chi\,\sqrt{(1-\mu_{n-1}^2)(1-\cos^2\psi)}. \qquad (6.9)$$

Обычно рассеяние предполагается изотропным в системе центра масс: $\cos\theta$ распределен равномерно в интервале $(-1,+1)$, а угол χ распределен равномерно в интервале $(0,2\pi)$.

Правило розыгрыша упругого рассеяния, изотропного в системе центра масс, состоит в следующем: находят два случайных числа γ и γ_1 $(0 \le \gamma, \gamma_1 \le 1)$ и полагают

$$\cos\theta = 2\gamma - 1, \quad \chi = 2\pi\gamma_1 .$$

Затем по формулам (6.7) – (6.9) находят $\cos\psi$, E_n и μ_n.

Таким образом, нейтрон из состояния $(z_{n-1}, \mu_{n-1}, E_{n-1})$ перешел в состояние (z_n, μ_n, E_n). Расчет траектории продолжается до тех пор, пока нейтрон не покинет пластинку, либо не поглотится. Затем выбираются (или разыгрываются) значения z_0, μ_0, E_0, с которых начинается расчет следующей траектории, и так до тех пор, пока не будет произведено обусловленное число испытаний N.

Из описанного примера видны следующие особенности вычислительного процесса: 1) траектории могут вычисляться независимо друг от друга, 2) время счета каждой траектории – пере-

менная величина, 3) информация, установленная в ходе каждого испытания суммируется, 4) в ходе вычислений нужно проверять выполнение условия, определяющего завершение вычислений.

Из этих особенностей вытекает и схема работы УВС. При числе ЭМ, равном r, одновременно вычисляют r траекторий. После окончания вычисления траектории каждая ЭМ посылает в одну из ЭМ информацию, на основе которой устанавливается окончание процесса решения. ЭМ, воспринимающая эту информацию, определяет момент завершения вычислений. Одновременно эта же ЭМ может вести суммирование результатов, полученных всеми ЭМ.

Нетрудно видеть, что при решении задач методом статистических испытаний загрузка УВС будет полной, то есть выигрыш во времени будет пропорционален числу ЭМ.

2°. Решение задач теории статистических решений[x)]

Теория статистических решений представляет большой интерес для многих отраслей науки и техники, так как она позволяет вырабатывать правила поведения в условиях неопределенности.

Теория статистических решений тесно связана с теорией игр. Раздел теории игр – "игры против природы" – по существу относится к теории статистических решений, так как в этом случае одному из партнеров игры противостоит некоторая не полностью известная ему обстановка – "состояние природы". Многие задачи теории игр (а следовательно, и теории статистических решений) могут успешно решаться методами линейного программирования [21], эффективность реализации которых на УВС была нами показана выше.

Рассмотрим в качестве примера приложение теории статистических решений к проблеме распознавания образов. На языке теории статистических решений эта проблема носит наименование "множественной классификации или дискриминантного анализа". Теория множественной классификации рассматривает задачу отнесения данного объекта к одной из нескольких возможных групп, к которым он может быть отнесен на основании множества наблюдений его измеримых характеристик.

С теоретической точки зрения задача дискриминантного анализа есть отыскание решений при фиксированном объеме выборки и конечном числе возможных состояний природы Q. Эта задача

x) См., например [20].

имеет следующие отличительные особенности: а) решение обычно принимается на основании единичного наблюдения нескольких коррелированных характеристик и б) допускается существование априорных вероятностей.

Пусть число возможных состояний природы $Q=(1,2,\dots,k)$, число возможных решений $A=(1,2,\dots,k)$ и потери равны постоянному числу, например, ω, если состояние природы есть j, а принимается решение, что оно (состояние) $i \neq j$. Если $i=j$, то потери равны нулю.

Распределение вероятностей для заданного $j \in Q$ будем обозначать через $p_j(z)$, где z – координата точки в выборочном пространстве. Пусть любое априорное распределение вероятностей для k состояний будет $\xi=(\xi(1),\dots,\xi(k))$.

Тогда для любого решения i апостериорный риск ε_z равен

$$\varepsilon_z(i) = \frac{\omega \sum_{(j\neq i)=1}^{k} p_j(z)\xi(j)}{\sum_{j=1}^{k} p_j(z)\xi(j)} . \qquad (6.10)$$

Байесово правило заключается в том, что нужно принимать решение i $(i=1,2,\dots,k)$, если

$$\sum_{(j\neq i)=1}^{k} p_j(z)\xi(j) \leqslant \sum_{(j\neq K)=1}^{k} p_j(z)\xi(j) \quad (i,K=1,2,\dots,k;\ i\neq K) \qquad (6.11)$$

или, что равнозначно:

$$\frac{p_K(z)}{p_i(z)} \leqslant \frac{\xi(i)}{\xi(K)} . \qquad (6.12)$$

При решении этой задачи на УВС из $n \leqslant k$ машин в каждой ЭМ можно одновременно производить вычисления по формулам (6.12) и (6.11) для $z=\frac{k}{n}$ решений. При этом на заключительной стадии для принятия окончательного решения i выполняется операция сравнения результатов отдельных ЭМ.

Таким образом, при решении задач теории статистических решений также можно получить выигрыш во времени решения, пропорциональный числу ЭМ как в том случае, когда задача сводится к соответствующей задаче линейного программирования, так и при непосредственном решении методами математической статистики.

§ 7. РЕШЕНИЕ ИНФОРМАЦИОННО-ЛОГИЧЕСКИХ ЗАДАЧ

Характерным для информационно-логических задач является обработка больших массивов информации по одним и тем же алгоритмам. Хотя эти алгоритмы являются довольно сложными, тем не менее они допускают, как правило, расчленение общего массива информации на независимые части, к каждой из которых применяется один и тот же алгоритм. Следовательно, информационно-логические задачи могут эффективно решаться на вычислительной системе. Действительно, расчленяя массив информации на n частей по числу машин в вычислительной системе, можно получить выигрыш во времени при обработке всего массива на УВС в n раз по сравнению с решением на одной ЭМ.

Рассмотрим примеры. Одной из важных задач, возникающих при решении информационно-логических проблем, является поиск слова в словаре при неупорядоченных массивах. При разбиении общего массива словаря на n частей можно ускорить поиск слова в n раз, если в каждой элементарной машине разместить соответствующую часть словаря и производить поиск слова сразу всеми элементарными машинами. В этом случае при решении задачи на УВС между элементарными машинами не происходит обмена информации и все машины работают независимо по своей программе до того, как одной или несколькими ЭМ выдается результат.

Весьма распространенной задачей является сравнение двух массивов информации с целью выяснения определенных закономерностей. Если массивы достаточно большие, то каждый из массивов может быть разбит на n частей. Пусть, например, требуется сравнить массив информации $M_1, M_2, \ldots, M_m$ с массивом информации $N_1, N_2, \ldots, N_n$. Пусть это сравнение производится по некоторому алгоритму A и пусть этот алгоритм требуется выполнить над каждой парой слов $b_j \in N_i$ и $c_k \in M_j$. Разместим в начале вычислений в каждой m_i машине части массивов M_i, N_i и алгоритм A. Тогда сначала алгоритм A в каждой элементарной машине m_i будет осуществляться над словами частей массивов N_1 и M_i. Одновременно с обработкой информации по алгоритму A в машине m_1 будет осуществляться передача обрабатываемых слов b_j массива N_1 во все остальные машины, где производится их сравнение со словами других массивов. После завершения операций для всех $i \neq 1$ M_i над словами массива N_1 включится на передачу вторая машина и будут производиться операции сравнения слов массива N_2 со словами массивов M_i всех машин. Процесс окончится после сравнения всех массивов N_i. Так как на каждом шаге будут работать все ЭМ, то и в этом случае общий выигрыш во времени будет пропорционален числу ЭМ в УВС.

Из рассмотрения приведенных выше типов задач можно сделать следующие выводы.

1. Для широкого круга задач существуют параллельные алгоритмы, эффективно реализуемые на универсальных вычислительных системах.

2. Время решения всех рассмотренных типов задач на УВС уменьшается прямо пропорционально числу машин в системе, причем их число изменяется в широком диапазоне.

3. Для всех рассмотренных типов задач удалось найти такие схемы параллельных алгоритмов, при которых общий объем хранимой информации распределяется равномерно между элементарными машинами. При этом практически не происходит увеличения общего объема хранимой информации и дополнительных затрат времени на обмен информацией между элементарными машинами по сравнению с решением данной задачи на одной ЭВМ.

4. Рассмотрение схем параллельных алгоритмов решения задач различных классов показывает, что программирование для универсальных вычислительных систем имеет тот же порядок сложности, что и для обычных ЭВМ.

Поступила в редакцию
18.XI.1964г.

ЛИТЕРАТУРА

1. Евреинов Э.В., Косарев Ю.Г. О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во Сиб. отд. АН СССР, 1962.
2. Евреинов Э.В. Универсальные вычислительные системы с переменной структурой. Данный сборник. стр.3-60.
3. Косарев Ю.Г. О методике решения задач на вычислительных системах. Данный сборник, стр.61-99.
4. Евреинов Э.В., Косарев Ю.Г. Матричный ρ -язык для описания параллельных алгоритмов. Данный сборник, стр.100-105.
5. Фадеев А.К. и Фадеева В.Н. Вычислительные методы линейной алгебры. М., Физматгиз, 1960.
6. Бекишев Г.А. О распараллеливании вычислительных алгоритмов. В сб.: "Вычислительные системы", 1963, вып. 5, 22-30 (Сиб. отд. АН СССР, Ин-т математики).

7. Бекишев Г.А. Об алгоритмах,эффективно реализуемых на вычислительных системах. В сб.: "Вычислительные системы" , 1963, вып.7,24-37 (Сиб. отд. АН СССР, Ин-т математики).
8. Бекишев Г.А. К вопросу эффективности решения простейших задач алгебры матриц на вычислительной системе,состоящей из четырех машин. В сб.: "Вычислительные системы", 1963, вып. 9, 3-29 (Сиб. отд. АН СССР, Ин-т математики).
9. Канторович Л.В. Математические методы организации и планирования производства. Изд-во ЛГУ, 1939.
10. Канторович Л.В. Экономический расчет наилучшего использования ресурсов. М., Изд. АН СССР, 1959.
11. Гасс С. Линейное программирование. Физматгиз. М.,1961.
12. Канторович Л.В., Гавурин М.К. Применение математических методов в вопросах анализа грузопотоков. В сб."Проблемы повышения эффективности работы транспорта". Изд-во АН СССР, 1949.
13. Dantzing G .B. Application of the simplex method to a transportation problem.В сб.под.ред. Koopmans T C Activity analysis of production and allocation, New York, John Wiley Sons, 1951, 359 - 373.
14. Лурье А.Л. Методы достижения наименьшего пробега грузов при составлении перевозочных схем. В сб. "Применение математики в экономических исследованиях". М.,Изд-во Соц. Эк., 1959, 354-389.
15. Kuhn H.W. The Hungarian method for solvind the assignment problem. Naval Res. Logist. Quart.2, 1955,83-97.
16. Березин И.С., Жидков Н.П. Методы вычислений. М., Физматгиз. 1959, т.I и II.
17. Ланс Д.Н. Численные методы для быстродействующих вычислительных машин. М., Изд-во иностр. лит., 1962.
18. Соболев С.Л. Лекции по теории кубатурных формул., 1964, Изд-во Новосибирского государственного университета.
19. Бусленко Н.П., Голенко Д.И., Соболь И.М., Срагович В.Г., Шрейдер Ю.А. Метод статистических испытаний. М., Физматгиз, 1962.
20. Блекуэлл Д. и Гиршик М.А. Теория игр и статистических решений. М., 1958, Изд-во иностр. лит.
21. Dantzig G.B. A proof of the equivalense of the program - ming problem and the game problem .В сб.под редакцией Koopmans T.C. Activity analysis of production and allocation, New York, John Wiley Sons, 1951, 330-335 .