№ 1. Р

Написана апрель 1954 г.
Хранится: кафедра вычислительной математики МГУ.

3-8.1

ДИПЛОМ

МОСКОВСКИЙ ОРДЕНА ЛЕНИНА
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М.В. ЛОМОНОСОВА

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
кафедра вычислительной математики

А.П.ЕРШОВ

ОБРАЩЕНИЕ МАТРИЦ

(дипломная работа)

Научный руководитель
доктор физико-математических
наук профессор А.А.ЛЯПУНОВ.

1954 г.

О Г Л А В Л Е Н И Е

3-8.4 3-348

ГЛАВА I

ОБРАЩЕНИЕ МАТРИЦ НА БЫСТРОДЕЙСТВУЮЩИХ ЭЛЕКТРОННЫХ СЧЁТНЫХ МАШИНАХ.

ВВЕДЕНИЕ. Задача об обращении матрицы, т.е. задача о решении матричного уравнения

$$AX = E,$$

где A данная матрица, E - единичная матрица, X - искомая матрица, самым тесным образом связана с решением систем линейных уравнений и поэтому является одной из важнейших и часто встречающихся задач линейной алгебры, т.к. к ней, в конечном счёте, приводит решение самых различных задач вычислительной математики, как, например, решение интегральных уравнений, решение различных краевых задач разностными методами, прямые вариационные методы, обработка результатов измерений, математическая статистика, некоторые экономические расчёты и т.п. В то же время обращение матрицы как вычислительная задача является одной из наиболее трудоёмких и тяжёлых задач ввиду большого времени, нужного для её решения, и требования большого объёма оперативной памяти. Поэтому практика вычислительного дела настойчиво требует или улучшения существующих методов обращения матриц, или создания новых методов, позволяющих сократить, насколько это возможно, как время обращения, так и расход оперативной памяти.

Из существующих методов обращения матриц для быстродействующих электронных счётных машин (БЭСМ) наиболее удобными, благодаря простоте вычислительной схемы, оказались методы, основанные на последовательном исключении неизвестных из системы линейных уравнений, а именно, метод Жордана и метод перекрёстного умножения. В то же время в работах фон Нёймана [1] и Тюринга [2] было показано, что методы исключения являются достаточно точными, благодаря медленному росту ошибок округления.

3-8.5 3-349

Однако, применять эти методы для обращения матриц на БЭСМ в том виде, как они были созданы, очень нерационально, т.к. классические схемы Жордана и перекрёстного умножения требуют $2n^2$ и $4n^2$ ячеек оперативной памяти, соответственно.

В настоящей работе предлагается новый метод обращения матриц, который мы условно будем называть "методом заполнения". Этот метод в достаточной степени удовлетворяет тем требованиям, о которых говорилось выше. Для обращения матриц методом заполнения на БЭСМ требуется n^2+n или даже только n^2 ячеек оперативной памяти. В то же время вычислительная схема этого метода очень проста и легко программируется.

Хотя метод заполнения основан на совсем другой идее, нежели методы исключения, он оказывается тесно связанным в вычислительном отношении с методами Жордана и перекрёстного умножения. В процессе вычисления получаются одни и те же числа, которые могут в некоторых случаях отличаться лишь знаками.

В работе приводятся различные варианты программ обращения матриц методом заполнения, а также приведены численные примеры, иллюстрирующие применение метода заполнения.

Вторая часть работы, посвящённая вопросам программирования, с первой частью не связана и составляет отдельную главу.

§1. НЕКОТОРЫЕ ФОРМУЛЫ, СВЯЗАННЫЕ С ОБРАЩЕНИЕМ МАТРИЦ.

А. Обозначения. $A=[a_{ij}]$ $(i,j=1,\dots,n)$ - квадратная неособенная матрица порядка n, состоящая из n строк, которые будут рассматриваться, как векторы. Т.е.

$$A = (A_1, A_2, \dots, A_n) = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{pmatrix}$$

Определитель матрицы A обозначается, как $|A|$ или $Д(A)$.

2) $X=[x_{ij}]$ $(i,j=1,\dots,n)$ - матрица, обратная к матрице A и состоящая из n столбцов, которые также будут рассматриваться, как векторы. Т.е.

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} = (X_1, X_2, \dots, X_n)$$

3) $M_A\begin{pmatrix} i_1, i_2, \dots, i_n \\ j_1, j_2, \dots, j_n \end{pmatrix}$ – минор n -го порядка матрицы A, образованный элементами этой матрицы, стоящими на пересечениях строк $i_1, i_2, \dots, i_n$ и стлбцов $j_1, j_2, \dots, j_n$.

4) A_{ij} – алгебраическое дополнение к элементу a_{ij} матрицы A. Как известно,

$$A_{ij} = (-1)^{i+j} M_A\begin{pmatrix} 1,2,\dots,i-1,i+1,\dots,n \\ 1,2,\dots,j-1,j+1,\dots,n \end{pmatrix} \qquad (1)$$

5) Матрицу

$$C = [A_{ji}] = \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix} \qquad (2)$$

назовём матрицей, союзной для матрицы A.

6) Также известной мы будем считать формулу, выражающую элементы обратной матрицы через союзную матрицу и определитель прямой матрицы. По этой формуле

$$x_{ij} = \frac{A_{ji}}{|A|}, \qquad (3)$$

т.е.

$$X = \begin{pmatrix} \frac{A_{11}}{|A|} & \frac{A_{21}}{|A|} & \dots & \frac{A_{n1}}{|A|} \\ \frac{A_{12}}{|A|} & \frac{A_{22}}{|A|} & \dots & \frac{A_{n2}}{|A|} \\ \cdot & \cdot & \cdot & \cdot \\ \frac{A_{1n}}{|A|} & \frac{A_{2n}}{|A|} & \dots & \frac{A_{nn}}{|A|} \end{pmatrix} \qquad (4)$$

7) Диагональным элементом матрицы A мы будем называть элемент, стоящий на главной диагонали A, т.е. имеющий вид a_{ii} .

8) $E = (\delta_{ij})$ – единичная матрица, т.е.

$$\delta_{ij} = \begin{cases} 1 & \text{если } i = j \\ 0 & \text{если } i \neq j \end{cases}$$

9) Мы будем говорить, что некоторая строка матрицы <u>значащая</u>, если она содержит недиагональные элементы, отличные от нуля. Строка будет <u>единичной</u>, если все недиагональные элементы нули, а диагональный элемент равен единице.

3-8.7 .3-357

Часто для наглядности мы будем изображать схематически матрицы квадратами. В этом случае заштрихованные участки будут изображать значащие строки матрицы.

Матрицу, имеющую m значащих строк $(i_1, i_2, \ldots, i_m)$, мы будем изображать как $A(i_1, i_2, \ldots, i_m)$. Если матрица имеет первые m строк значащими, то она изображается, как $A^{(m)}$. Матрицы и определители такого вида мы будем называть прямоугольными.

10) В дальнейшем нам часто придётся матрицу разбивать на 4 клетки, соответствующие 1) $i \leq m, j \leq m$, 2) $i > m, j \leq m$, 3) $i \leq m, j > m$, 4) $i > m, j > m$. Эти клетки, соответственно, мы будем называть верхней левой, нижней левой, верхней правой и нижней правой частями матрицы.

Кроме того заметим, что слова "прямая матрица" и "обратная матрица" у нас будут настолько часто встречаться, что мы их будем писать сокращённо - "пр. м." и "обр.м.".

Б. ТЕОРЕМА. Пусть неособенная матрица имеет вид $A(i_1, i_2, \ldots, i_m)$. Тогда её определитель равен $M_A\begin{pmatrix} i_1, i_2, \ldots, i_m \\ j_1, j_2, \ldots, j_m \end{pmatrix}$, а обратная матрица тоже имеет вид $X(i_1, i_2, \ldots, i_m)$.

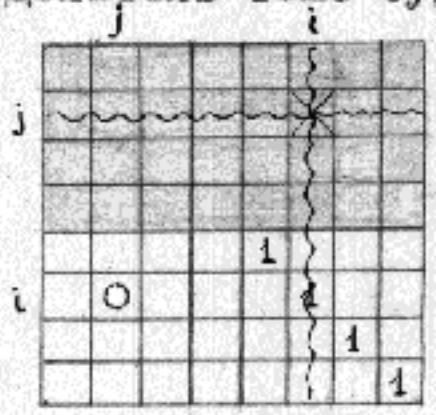
Док-во. Сначала найдём определитель матрицы A. Для этого будем раскладывать Д(A) по элементам единичных строк. Т.к. каждая единичная строка имеет отличный от нуля только диагональный элемент, равный единице, то при разложении определителя по элементам каждой единичной строки его порядок будет уменьшаться на единицу, будет вычёркиваться единичная строка и соответствующий столбец с тем же индексом, а знак будет сохраняться, т.к. сумма индексов диагонального члена всегда чётная. Вычеркнув все единичные строки и соответствующие им столбцы, мы и получим, что

$$|A(i_1, i_2, \ldots, i_m)| = M_A\begin{pmatrix} i_1, i_2, \ldots, i_m \\ j_1, j_2, \ldots, j_m \end{pmatrix} \qquad (5)$$

Для доказательства 2-го утверждения достаточно показать, что каждая единичная строка матрицы A будет единичной и для обр.м.X. Из предыдущего ясно, что алгебраическое дополнение для диагонального члена единичной строки равно определителю матрицы. Отсюда следует, что диагональный элемент соответ-

3-8,8 3-352

ствующей строки обр.м. будет равен единице.Пусть теперь x_{ij} недиагональный элемент обр.м.,и i -я строка п.м.-единичная. Чтобы вычислить x_{ij} ,надо найти алгебраическое дополнение к элементу пр.м. a_{ji} .Однако при образовании этого алгебраического дополнения мы вычеркнем (см.чертёж I) единственную единицу в i -й единичной строке,так что в получившемся определителебудет одна строка,содержащая сплошь нули,отчего определитель тоже будет равен нулю.Т.к. это будет верно для любого недиагонального элемента i -й строки,то утверждение доказано.



Черт. I

Отсюда следует,что число единичных строк при обращении матрицы не только сохраняется,но и что новых единичных строк не добавляется.Действительно,если бы пр.м.имела бы m единичных строк,а обр.м. имела бы $m+k$ единичных строк, то,по доказанному, пр.м.,будучи матрицей,обратной к обр.м.,тоже имела бы $m+k$ единичных строк ,что противоречит условию.Теорема доказана полностью.

В частности следует,что

$$\left(A^{(m)}\right)^{-1} = X^{(m)} \qquad (6)$$

и что

$$\left|A^{(m)}\right| = M_A\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,m\end{pmatrix}. \qquad (7)$$

В. Затем мы выведем фррмулы,выражающие элементы значащих строк обр.м.через миноры пр.м. $A = A^{(m)}$, т.е. когда для $i>m$ $a_{ij}=x_{ij}=\delta_{ij}$

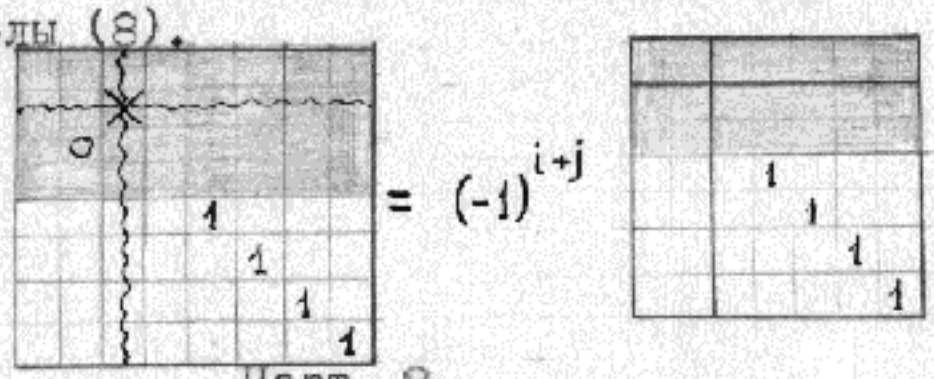
Основной формулой,которой мы будем пользоваться,будет ф-ла

$$x_{ij}^{(m)} = \frac{M_A\begin{pmatrix}1,2,\ldots,j-1,j+1,\ldots,n\\1,2,\ldots,i-1,i+1,\ldots,n\end{pmatrix}}{M_A\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,m\end{pmatrix}} \qquad (8)$$

3-8.9 3-353

Нам будет удобно рассмотреть отдельно два случая: $j \leqslant m$ и $j > m$.

а) $j \leqslant m$. Рассмотрим, какой вид будет иметь минор в числителе ф-лы (8).



Черт. 2

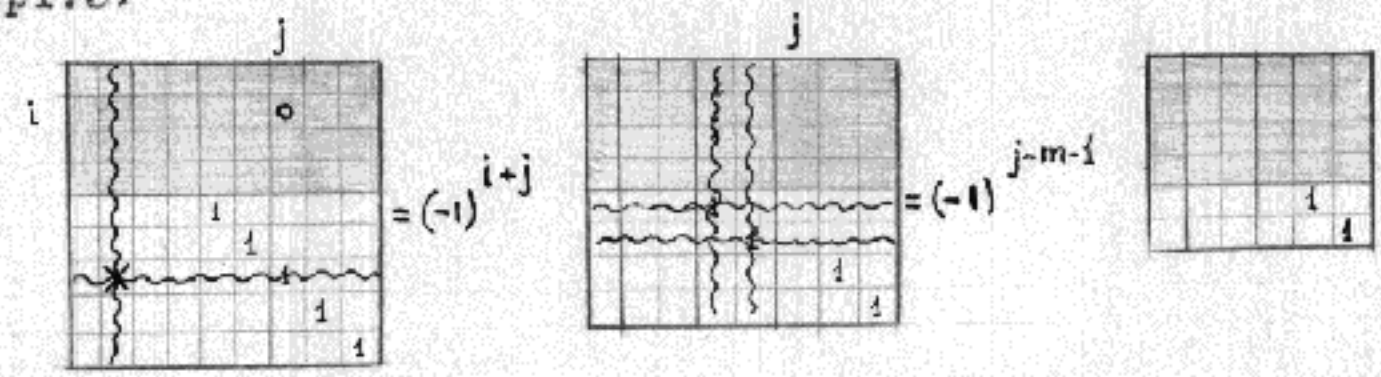
Т.к. $i \leqslant m$ и $j \leqslant m$, то после вычёркивания j-й строки и i-го столбца (см. черт. 2) оставшийся определитель тоже останется прямоугольным, только число значащих строк уменьшится на единицу. Поэтому, для $j \leqslant m$

$$M_A\begin{pmatrix}1,2,\ldots,j-1,j+1,\ldots,n\\1,2,\ldots,i-1,i+1,\ldots,n\end{pmatrix} = M_A\begin{pmatrix}1,2,\ldots,j-1,j+1,\ldots,m\\1,2,\ldots,i-1,i+1,\ldots,m\end{pmatrix}$$

и

$$x_{ij} = (-1)^{i+j}\,\frac{M_A\begin{pmatrix}1,2,\ldots,j-1,j+1,\ldots,m\\1,2,\ldots,i-1,i+1,\ldots,m\end{pmatrix}}{M_A\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,m\end{pmatrix}} \qquad (9)$$

б) $j > m$. В этом случае элемент a_{ji}, к которому надо взять алгебраическое дополнение, будет находиться в нижнем левом углу. После вычёркивания j-й строки и i-го столбца у нас получится прямоугольный определитель $(n-1)$-го порядка, который будет иметь следующее строение: 1-е m строк у него будут значащие, следующие $(j-1-m)$ строк будут содержать по одной единице, но не на главной диагонали, а на одно место левее и последние $n-j$ строк будут обычными единичными строчками (см. черт. 3)



Черт. 3

3-8.10 3-354

Чтобы вычислить этот определитель, мы приведём его к прямоугольному виду, для чего будем разлагать его последовательно по средним ($j-m-1$) строкам, начиная снизу. При первом разложении по элементам ($j-1$)-й строки определитель переменит знак, т.к. в неразложенном определителе этот элемент имел индексы $j-1$, $j-1$, а в разложенном определителе эта единица имеет индексы $j-1$, $j-2$, сумма которых нечётна. При каждом разложении по следующим строкам определитель также будет менять знак, т.к. сумма индексов элемента, к которому будет браться алгебраическое дополнение, будет меняться на чётное число.

После всех $j-m-1$ разложений определитель (j-m-1 раз переменит знак и) будет иметь те же m значащих строк с вычеркнутыми столбцами с номерами i, $m+1, \ldots, j-1$. Т.е. для $j>m$

$$M_A\begin{pmatrix}1,2,\ldots,j-1,j+1,\ldots,n\\1,2,\ldots,i-1,i+1,\ldots,n\end{pmatrix}=(-1)^{j-m-1}M_A\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,i-1,i+1,\ldots,m,j\end{pmatrix}$$

и

$$x_{ij}^{(m)}=(-1)^{i+m}\frac{M_A\begin{pmatrix}1,2,\ldots,i-1,i+1,\ldots,m,j\end{pmatrix}}{M_A\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,m\end{pmatrix}} \qquad (10)$$

Г. Теперь поставим следующую задачу. Пусть известна пр.м. A и обр. к ней м. X. Пусть, далее, к m-й строке м. A-A_m прибавили некоторый вектор δA_m ($\delta a_{m1}, \delta a_{m2}, \ldots, \delta a_{mn}$). Полученную м. обозначим A^*. Предположим теперь, что м. A^* невырожденная и попробуем найти формулы, по которым можно найти м. X^*, обратную к м. A^*, исходя из м.A и м. X. Оказывается, что эти формулы очень просты; они-то и являются основными для нашего метода.

Будем считать, далее, что единичная м.E состоит из n единичных столбцов $E_1, E_2, \ldots, E_n$. Тогда, как известно, столбцы м. X, рассматриваемые как неизвестные, удовлетворяют следующим системам уравнений

$$A X_j = E_j \qquad (j=1,\ldots,n) \qquad (11)$$

Аналогично и для матриц A^* и X^*:

$$A^* X_j^* = E_j \qquad (j=1,\ldots,n) \qquad (12)$$

3-8.11 .3-355

Пусть

$$X^* = X + \delta X$$

и

$$A^* = A + \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \\ \delta a_{m1} & \delta a_{m2} & \cdots & \delta a_{mn} \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

тогда уравнения (12) можно записать в таком виде:

$$\left[\begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \\ \delta a_{m1} & \delta a_{m2} & \cdots & \delta a_{mn} \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} + A \right] (X_j + \delta X_j) = E_j \qquad (13)$$

$$(j = 1, \ldots, n)$$

Раскрыв скобки и выполнив умножение, имея в виду, что ур-я (11) выполняются, получим для вектора-неизвестного δX_j следующую систему:

$$\left[\begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \\ \delta a_{m1} & \delta a_{m2} & \cdots & \delta a_{mn} \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} + A \right] \delta X_j = - \begin{pmatrix} 0 \\ \cdots \\ 0 \\ (\delta A_m X_j) \\ 0 \\ \cdots \\ 0 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow m\text{-я} \\ \text{компонента} \\ \\ \\ \\ (j = 1, \ldots, n) \end{matrix} \qquad (14)$$

или:

$$A^* \delta X_j = -(\delta A_m X_j) E_m \qquad (15)$$

Сравнивая ур-я (15) и m -е ур-е в системе (12), заключаем, что

$$\delta X_j = -(\delta A_m X_j) X_m^* \qquad (j = 1, \ldots, n) \qquad (16)$$

или

$$\delta X_j = -(\delta A_m X_j)(X_m + \delta X_m) \qquad (j = 1, \ldots, n) \qquad (17)$$

Возьмём теперь из системы (17) m -е уравнение, т.е. уравнение, соответствующее номеру изменённой строки. Оно имеет вид:

$$\delta X_m = -(\delta A_m X_m)(X_m + \delta X_m) \qquad (18)$$

3-8.12 3-356

Отсюда сразу можно найти вектор δX_m .Он равен:

$$\delta X_m = - \frac{X_m (\delta A_m \cdot X_m)}{1 + (\delta A_m \cdot X_m)} \qquad (19)$$

Эта формула написана в предположении,что $1+(\delta A_m X_m) \neq 0$. Мы покажем в дальнейшем,что так будет всегда,когда м. A^* невырожденная.Отсюда

$$X_m^* = X_m + \delta X_m = - \frac{X_m}{1+(\delta A_m \cdot X_m)} \qquad (20)$$

Но поскольку X_m^* известно,то δX_j вычисляются сразу по ф-лам (17),и для столбцов м. X^* имеем,что

$$X_j^* = X_j - \frac{X_m (\delta A_m \cdot X_j)}{1+(\delta A_m \cdot X_m)} \qquad (i = 1,\ldots,n) \qquad (21)$$

или в записи по компонентам

$$x_{ij}^* = x_{ij} - \frac{x_{im}(\delta A_m \cdot X_j)}{1+(\delta A_m X_m)} \qquad (i,j = 1,\ldots,n) \qquad (22)$$

причём x_{im}^* (элементы m -го столбца)удовлетворяют ур-ям (22) тождественно.

§2. ОПИСАНИЕ МЕТОДА ЗАПОЛНЕНИЯ И ЕГО ВЫЧИСЛИТЕЛЬНОЙ СХЕМЫ.

Непосредственно из формул (22) можно составить следующую вычислительную схему.В качестве исходных берём две единичных матрицы.Одна из них будет прямой,- другая - обратной. Первую строку одной матрицы заменяем 1-й строкой обращаемой м. A,т.е. $\delta A_1 = A_1 - E_1$. Обозначим исходную м. $X^{(0)} = E$.Вычисляем n скалярных произведений $(X_j^{(0)} \delta A_1)$ $(j = 1,\ldots,n)$ и затем по формулам (22) элементы новой обр. м. X^* .Т.к. прямая матрица вида $A^{(1)}$,то,в соответствии с нашей теоремой, и $X^* = X^{(1)}$. Затем 2-ю строку м. $A^{(1)}$ заменяем 2-й строкой обращаемой м. A, т.е, $\delta A_2 = A_2 - E_2$.Снова вычисляем n скалярных произведений $(X_j^{(1)} \delta A_2)$ $(j = 1,\ldots,n)$ и опять по формулам (22) вычис-

3-8.13 3-357

ляем элементы м. $X^{(2)}$ и т.д.

Таким образом,всё вычисление обратной матрицы разбивается на n совершенно одинаковых этапов.После $(m-1)$-го этапа мы получаем м. $X^{(m-1)}$,у которой заполнены $m-1$ первые строк,и которая является обратной для м. $A^{(m-1)}$.Мы вычисляем n скалярных произведений $(X_j^{(m-1)}\delta A_m)$ $(j = 1,\dots,n)$ и затем вычисляем элементы м. $X^{(m)}$ по по ф-лам:

$$x_{ij}^{(m)} = x_{ij}^{(m-1)} - \frac{x_{im}^{(m-1)}(\delta A_m X_j^{(m-1)})}{1+(\delta A_m X_m)} \qquad (23)$$

Пусть $A - E = \tilde{A}$ (м. $\tilde{A}$ состоит из строк $\tilde{A}_1,\tilde{A}_2,\dots,\tilde{A}_n$).При таких обозначениях ф-лы (23) примут вид:

$$x_{ij}^{(m)} = x_{ij}^{(m-1)} - \frac{x_{im}^{(m-1)}(\tilde{A}_m X_j^{(m-1)})}{1+(\tilde{A}_m X_m)} \qquad (24)$$

Ячейки памяти,отведённые для обр. м. постепенно заполняются значащими строками ~~строками~~,отчего мы и называем предлагаемый метод методом заполнения.После n этапов мы получаем м. X,обратную к м. A.

Очевидно,что можно построить вычислительную схему,позволяющую использовать только n^2+n ячеек оперативной памяти. Действительно,поступим следующим образом.Заполним n^2 ячеек памяти элементами матрицы $\tilde{A} = A - E$. n ячеек будем использовать для помещения скалярных произведений.

Содержимое первой строки нам будет нужно только на первом этапе для подсчёта скалярных произведений $(\tilde{A}_1 \cdot X_j^{(0)})$.Вычисляем их и помещаем в отведённую им группу из n ячеек. После этого ведём вычисления по ф-ле (24).После первого этапа мы получим только одну значащую строку,которую и поставим на место $\tilde{A}_1$.

Поступая так каждый раз,после $(m-1)$-го этапа мы получим,что I-е $(m-1)$ строк заняты элементами м. $X^{(m-1)}$,а $n-m+1$ строк заняты неиспользованными элементами М. $\tilde{A}$.В m-й строке находится $\tilde{A}_m = A_m - E_m$.Вычисляем скалярные произведения $(\tilde{A}_m X_1^{(m-1)}),(\tilde{A}_m X_2^{(m-1)}),\dots,(\tilde{A}_m X_n^{(m-1)})$,затем очищаем m-ю строку и ставим туда $X^{(m-1)} - E_m$.После этого по ф-ле (24) подсчитываем

(m-ю строку м.)

3-8.14 3-358

элементы м. $X^{(m)}$,доводя вычисления только до m -й строки включительно,т.е.подсчитывая только значащие строки.

Совершенно очевидно,что,для того,чтобы по этой схеме вести вычисления,необходимо,чтобы все промежуточные матрицы $A^{(m)}$ были неособенными т.е. чтобы все главные миноры м. А не обращались в нуль.

Сразу укажем на недостатки такой вычислительной схемы:

1 .На каждом этапе нам надо вычислять n скалярных произведений.Это увеличивает время ,необходимое для рещения задачи;в то же время точность вычислений понижается,т.к. , если сомножители в скалярном произведении имеют ошибку ϵ , то скалярное произведение получается с ошибкой,вообще говоря, в n раз большей.

2 .Для вычисления скалярных произведений нужно вводить переменные команды сравнения,оканчивающие вычисление,т.к. на разных этапах столбцы м. $X^{(m)}$ содержат равное число значащих компонент.

3 .Для подсчётов элементов м. $X^{(m)}$ также надо вводить переменные команды сравнения в силу тех же обстоятельств,что и в п.2 .

Однако,как будет показано ниже,некоторое видоизменение вычислительной схемы устраняет отмеченные недостатки и придаёт методу простоту и стройность ,которые позволяют считать его одним из самых удобных методов обращения матриц на БЭСМ.Этот факт устанавливает следующая

ТЕОРЕМА: Пусть А неособенная матрица и пусть строятся по индукции две последовательности матриц: $X^{(0)}, X^{(1)}, \ldots, X^{(n)}$ и $B^{(1)}, B^{(2)}, \ldots, B^{(n)}$ - таких,что

$$B^{(1)} = A - E \qquad \text{и} \qquad X^{(0)} = E \tag{25}$$

$$\text{и} \quad b_{ij}^{(m+1)} = b_{ij}^{(m)} - \frac{b_{im}^{(m)} b_{mj}^{(m)}}{1 + b_{mm}^{(m)}} \qquad \text{и} \quad x_{ij}^{(m)} = x_{ij}^{(m-1)} - \frac{x_{im}^{(m-1)} b_{mj}^{(m)}}{1 + b_{mm}^{(m)}}, \tag{26}$$

$i \leq m$

$$(m = 1, 2, \ldots, n)$$

Тогда $X^{(m)} = (A^{(m)})^{-1}$ и,в частности, (27)

$$X^{(n)} = X = A^{-1}. \tag{28}$$

3-2.15 3-359

Доказательство будем вести в предположении, что все знаменатели в нуль не обращаются, а потом выясним условия, при которых это требование выполняется.

Сравнивая ф-лы (24) и (26), мы видим, что, для того чтобы доказать теорему, достаточно доказать, что

$$(\tilde{A}_i \cdot X_j^{(m)}) = b_{ij}^{(m+1)}. \qquad (29)$$

Докажем это соотношение по индукции. Для $m = 0$ получаем:

$$(\tilde{A}_i \cdot X_j^{(0)}) = (\tilde{A}_i\ E_j) = a_{ij} - \delta_{ij} = b_{ij}^{(1)}.$$

в связи с (24). Пусть теперь

$$(\tilde{A}_i \cdot X_j^{(m-1)}) = b_{ij}^{(m)} \qquad (30)$$

и покажем, что (29) в этом случае выполняется. Подсчитаем, чему равняется $(\tilde{A}_i \cdot X_j^{(m)})$. По ф-ле (24)

$$(A_i\ X_j^{(m)}) = \left(\tilde{A}_i \left[X_j^{(m-1)} - \frac{X_m^{(m-1)} (\tilde{A}_m\ X_j^{(m-1)})}{1 + (A_m\ X_m^{(m-1)})} \right] \right) = (\tilde{A}_i \cdot X_j^{(m-1)}) - \frac{(\tilde{A}_i \cdot X_m^{(m-1)})(\tilde{A}_m\ X_j^{(m-1)})}{1 + (A_m\ X_m^{(m-1)})} =$$

по предположению (30)

$$= b_{ij}^{(m)} - \frac{b_{im}^{(m)} b_{mj}^{(m)}}{1 + b_{mm}^{(m)}} = b_{ij}^{(m+1)}.$$

Теорема доказана.

Из этой теоремы мы видим, что на m-м этапе вычислений нужные нам n скалярных произведений $(\tilde{A}_m\ X_j^{(m-1)})$ $(j = 1, \ldots, n)$ будут сами собой образовываться в m-й строке, если на каждом, i-м, этапе доводить вычисления по ф-лам (24) не до i-й строки, а каждый раз преобразовывать всю матрицу до конца. Это следует из того, что, как показывают ф-лы (25) и (26), вычисление элементов матриц $B^{(m+1)}$ и $X^{(m)}$ производятся по одинаковым формулам.

Назовём на m-м этапе m-й столбец и m-ю строку <u>главным столбцом</u> и <u>главной строкой</u>, соответственно. Получается следующий порядок вычислений:

1. В n^2 ячейках памяти размещаем матрицу $\tilde{A} = A - E$.

2. В добавочных n ячейках размещаем 1-ю главную строку. Это первая серия скалярных произведений для 1-го этапа.

3. Ставим на место главной строки первую строку м. $X^{(0)} - E$.

3-8.16 3-360

4 .К первому произведению прибавляем единицу.Получаем $(\tilde{A}_1 \cdot X_1^{(m)})+1$.

5 .Делим 1-ю компоненту главного столбца на $(\tilde{A}_1 \cdot X_1^{(m)})+1$ и затем по ф-ле (26) просчитываем 1-ю строку новой матрицы.

Делим вторую компоненту главного столбца на $(\tilde{A}_1 \cdot X_1^{(m)})+1$ и затем по той же формуле просчитываем 2-ю строку новой матрицы.

Так поступаем n раз,пока не просчитаем все n строк нашей матрицы.

1-й этап закончен;1-я строка занята значащей строкой матрицы $X^{(1)}$,в остальных строках стоят скалярные произведения векторов $X_j^{(1)}$ $(j = 1,\ldots,n)$ на строки матрицы $\tilde{A}$ - A_i $(i = 2,\ldots,n)$.

m -й этап:

1.Сносим m -ю главную строку в добавочные n ячеек.После $(m-1)$-го этапа здесь находятся нужные нам скалярные произведения.

2.Ставим на место главной строки m -ю строку м. $X^{(m-1)}$ - E_m.

3.К m -му скалярному произведению прибавляем 1.Получаем $(\tilde{A}_1 \cdot X_m^{(m-1)})+1 = b_{mm}^{(m)}+1$.

4.Делим 1-ю компоненту главного столбца на $b_{mm}^{(m)}+1$ и по ф-ле (26) просчитываем первую строку новой матрицы.

Делим 2-ю компоненту главного столбца на $b_{mm}^{(m)}+1$ и т.д.

Эти вычисления повторяем n раз,пока не дойдём до конца матрицы.

m -й этап закончен.После него 1-е m строк заняты значащими строками м. $X^{(m)}$,в нижних $n-m$ строках находятся $n-m$ строк матрицы $B^{(m+1)}$,т.е. скалярные произведения векторов $X_j^{(m)}$ $(j = 1,\ldots,n)$ на строки м. $\tilde{A}$ - $\tilde{A}_i$ $(i = m+1,\ldots,n)$.

После n этапов мы получаем обратную матрицу.

На черт. 4 изображён процесс обращения матрицы 4-го порядка.

3-8.17 -3-361

$a_{11}^{(1)}$	$a_{12}^{(1)}$	$a_{13}^{(1)}$	$a_{14}^{(1)}$
$x_{11}^{(0)}$	$x_{12}^{(0)}$	$x_{13}^{(0)}$	$x_{14}^{(0)}$
$a_{21}^{(1)}$	$a_{22}^{(1)}$	$a_{23}^{(1)}$	$a_{24}^{(1)}$
$a_{31}^{(1)}$	$a_{32}^{(1)}$	$a_{33}^{(1)}$	$a_{34}^{(1)}$
$a_{41}^{(1)}$	$a_{42}^{(1)}$	$a_{43}^{(1)}$	$a_{44}^{(1)}$
$b_{11}^{(1)}$	$b_{12}^{(1)}$	$b_{13}^{(1)}$	$b_{14}^{(1)}$

+1

0	1	1	1
1	0	0	0
2	1	1	1
2	2	2	1
2	2	2	2
0	1	1	1

+1

$x_{11}^{(1)}$	$x_{12}^{(1)}$	$x_{13}^{(1)}$	$x_{14}^{(1)}$
$b_{21}^{(2)}$	$b_{22}^{(2)}$	$b_{23}^{(2)}$	$b_{24}^{(2)}$
$x_{21}^{(1)}$	$x_{22}^{(1)}$	$x_{23}^{(1)}$	$x_{24}^{(1)}$
$b_{31}^{(2)}$	$b_{32}^{(2)}$	$b_{33}^{(2)}$	$b_{34}^{(2)}$
$b_{41}^{(2)}$	$b_{42}^{(2)}$	$b_{43}^{(2)}$	$b_{44}^{(2)}$
$b_{21}^{(2)}$	$b_{22}^{(2)}$	$b_{23}^{(2)}$	$b_{24}^{(2)}$

+1

1	-1	-1	-1
2	0	-1	-1
0	1	0	0
2	0	0	-1
2	0	0	0
2	0	-1	-1

+1

$x_{11}^{(2)}$	$x_{12}^{(2)}$	$x_{13}^{(2)}$	$x_{14}^{(2)}$
$x_{21}^{(2)}$	$x_{22}^{(2)}$	$x_{23}^{(2)}$	$x_{24}^{(2)}$
$b_{31}^{(3)}$	$b_{32}^{(3)}$	$b_{33}^{(3)}$	$b_{34}^{(3)}$
$x_{31}^{(2)}$	$x_{32}^{(2)}$	$x_{33}^{(2)}$	$x_{34}^{(2)}$
$b_{41}^{(3)}$	$b_{42}^{(3)}$	$b_{43}^{(3)}$	$b_{44}^{(3)}$
$b_{31}^{(3)}$	$b_{32}^{(3)}$	$b_{33}^{(3)}$	$b_{34}^{(3)}$

+1

3	-1	-2	-2
-2	1	1	1
2	0	0	-1
0	0	1	0
2	0	0	0
2	0	0	-1

+1

$x_{11}^{(3)}$	$x_{12}^{(3)}$	$x_{13}^{(3)}$	$x_{14}^{(3)}$
$x_{21}^{(3)}$	$x_{22}^{(3)}$	$x_{23}^{(3)}$	$x_{24}^{(3)}$
$x_{31}^{(3)}$	$x_{32}^{(3)}$	$x_{33}^{(3)}$	$x_{34}^{(3)}$
$b_{41}^{(4)}$	$b_{42}^{(4)}$	$b_{43}^{(4)}$	$b_{44}^{(4)}$
$x_{41}^{(3)}$	$x_{42}^{(3)}$	$x_{43}^{(3)}$	$x_{44}^{(3)}$
$b_{41}^{(4)}$	$b_{42}^{(4)}$	$b_{43}^{(4)}$	$b_{44}^{(4)}$

+1

7	-1	-2	-4
-4	1	1	2
-2	0	1	1
2	0	0	0
0	0	0	1
2	0	0	0

+1

x_{11}	x_{12}	x_{13}	x_{14}
x_{21}	x_{22}	x_{23}	x_{24}
x_{31}	x_{32}	x_{33}	x_{34}
x_{41}	x_{42}	x_{43}	x_{44}

15	-1	-2	-4
-8	1	1	2
-4	0	1	1
-2	0	0	1

Черт. 4.

§3. СВЯЗЬ МЕТОДА ЗАПОЛНЕНИЯ С МЕТОДОМ ЖОРДАНА И МЕТОДОМ ПЕРЕКРЁСТНОГО УМНОЖЕНИЯ.

Описание методов Жордана и перекрёстного умножения. Метод Жордана в применению к обращению матриц выглядит следующим образом. В соответствии с матричным уравнением для обратной матрицы

$$AX = E$$

записываются рядом матрица A и единичная матрица E. Первая строка полученной таким образом матрицы с n строками и $2n$ столбцами делится на коэфициент при первом неизвестном в первом уравнении, и производится обычный процесс исключения этого неизвестного из всех уравнений. Затем 2-я строка делится на коэфициент при 2-м неизвестном во втором уравнении, это второе неизвестное опять исключается из всех уравнений и т.д. После n исключений на месте единичной матрицы получается обратная матрица; при каждом исключении матрица преобразуется и "переползает" на одну строку вправо (см. черт.5).

1	1	1	1	1				
2	3	1	1		1			
2	2	3	1			1		
2	2	2	3				1	
	1	1	1	1				
	1	-1	-1	-2	1			
	0	1	-1	-2		1		
	0	0	1	-2			1	
		2	2	3	-1			
		-1	-1	-2	1			
		1	-1	-2	0	1		
		0	1	-2	0		1	
			4	7	-1	-2		
			-2	-4	1	1		
			-1	-2	0	1		
			1	-2	0	0		
				15	-1	-2	-4	
				-8	1	1	2	
				-4	0	1	1	
				-2	0	0	1	

0

1

2

3

4

Черт. 5

3-8.19 3-363

Мы не будем обосновывать метод перекрёстного умножения ([3],[4]),а укажем лишь вычислительную схему,к которой этот метод приводит.Сначала составляют матрицу 2n -го порядка следующим образом: верхний левый угол занимают обращаемой матрицей A ,верхний правый - единичной матрицей,нижний левый угол - минус единичной,а нижний правый угол остаётся пустым.Затем производят такое же исключение неизвестных, что и в схеме Жордана,однако, исключают неизвестные не только из строк матрицы A ,но и из подписанных снизу строк м.-Е. После каждого исключения матрица преобразуется и "переползает" по диагонали на одну строку вниз и на один столбец вправо. После n исключений получается обратная матрица,которая располагается в правом нижнем углу (см. черт: 6)

Сразу заметим,что при обращении матрицы методом Жордана и и методом перекрёстного умножения промежуточные матрицы будут получатся одни и те же с точностью до перестановки строк. Действительно,процесс исключения Жордана от процесса перекрёстного умножения отличается лишь тем,что в первом случае m-я строка,с помощью которой производится исключение,остаётся на своём месте,будучи поделённой на свой главный элемент,в то время как во втором случае эта же строка остаётся не на своём месте,а сносится в (n+m)-ю строку исходной матрицы 2n-го порядка.Это замечание с очевидностью следует из сравнения вычислительных схем для этих методов(см.черт.5 и 6). Обозначим исходную м. 2n-го порядка в методе перекрёстного умножения через $C^{[0]}$.Тогда процесс перекрёстного умножения можно представить как последовательное вычисление n матриц $C^{[1]}, C^{[2]}, \ldots, C^{[m]}, \ldots, C^{[n]}$,причём при переходе от m-1 к m порядок матрицы уменьшается на единицу и $C^{[n]} = X = A^{-1}$. В работе Д.П.Гросмана [5] даётся формула,выражающая элементы $c_{ij}^{[m]}$ м.$C^{[m]}$ через миноры исходной м.$C^{[0]}$.Эта формула такова:

$$c_{ij}^{[m]} = \frac{M_{C^{[0]}}\begin{pmatrix}1,2,\ldots,m,m+i\\1,2,\ldots,m,m+j\end{pmatrix}}{M_{C^{[0]}}\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,m\end{pmatrix}} \qquad (31)$$

Преобразуем эти формулы так,чтобы в них вместо миноров м.$C^{[0]}$

3-364

3-8.20

1	1	1	1	1			
2	3	1	1		1		
2	2	3	1			1	
2	2	2	3				1
-1							
	-1						
		-1					
			-1				

0

1	-1	-1	-2	1			
0	1	-1	-2		1		
0	0	1	-2			1	
1	1	1	2				
-1							
	-1						
		-1					

1

1	-1	-2	0	1			
0	1	-2	0			1	
2	2	3	-1				
-1	-1	-2	1				
	-1						
		-1					

2

1	-2	0	0	1			
4	7	-1	-2				
-2	-4	1	-1				
-1	-2	0	1				
	-1						

3

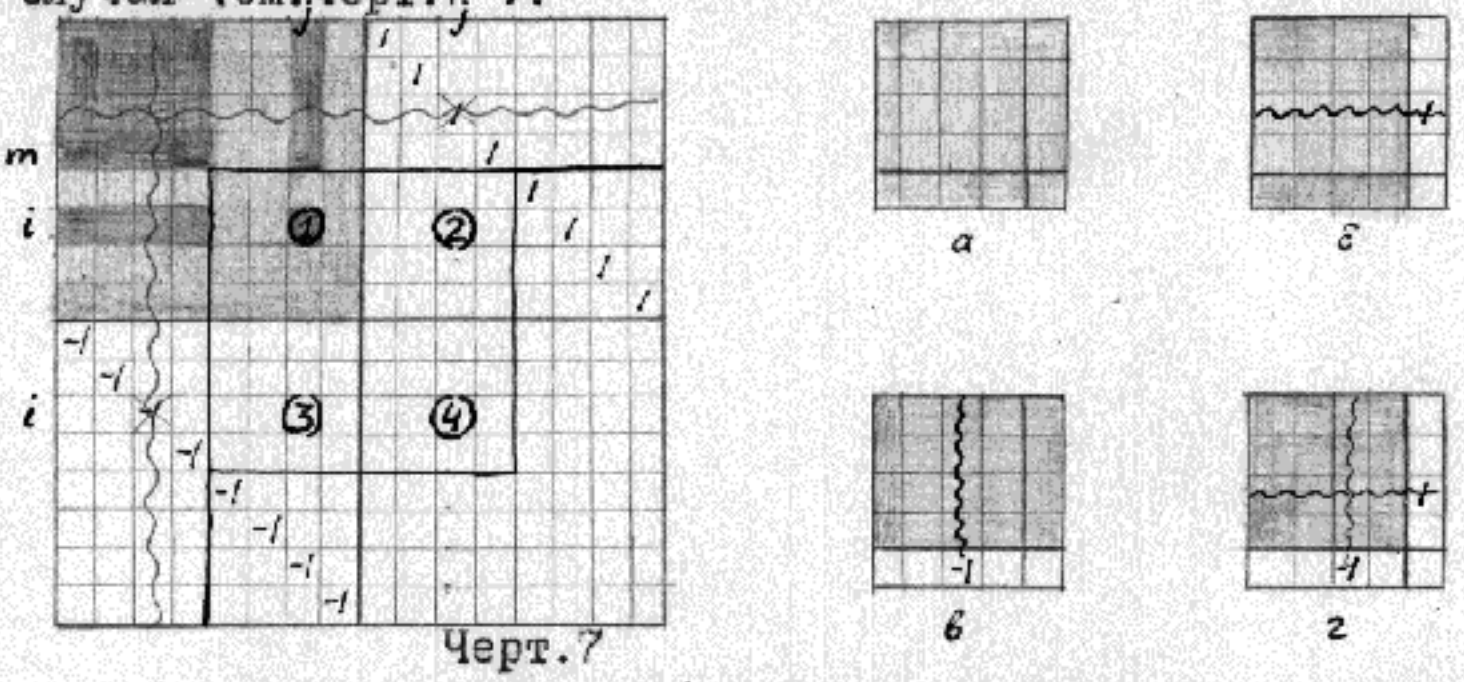
15	-1	-1	-4				
-8	1	1	2				
-4	0	1	1				
-2	0	0	1				

4

Черт. 6

3-8.21 3-365

были бы миноры исходной м. А. Для этого рассмотрим следующие 4 случая (см. черт. 7).



Черт. 7

1). $\underline{i \leq n-m, j \leq n-m}$. В этом случае (см. $7_а$) элементы $(m+i)$-й строки и $(m+j)$-го столбца, содержатся внутри м. А и поэтому

$$C_{ij}^{[m]} = \frac{M_A \begin{pmatrix} 1,2,\ldots,m,m+i \\ 1,2,\ldots,m,m+j \end{pmatrix}}{M_A \begin{pmatrix} 1,2,\ldots,m \\ 1,2,\ldots,m \end{pmatrix}} \qquad (32)$$

2). $\underline{i \leq n-m, j > n-m}$. В этом случае (см. $7_б$) в миноре в числителе ф-лы (31) $(m+1)$-й столбец будет иметь лишь одну единицу, находящуюся в $(j+m-n)$-й строке. Разложив минор по $(m+1)$-му столбцу, получим, что

$$M_{C^{[m]}} \begin{pmatrix} 1,2,\ldots,m,m+i \\ 1,2,\ldots,m,m+j \end{pmatrix} = (-1)^{j+m-n+m+1} M_A \begin{pmatrix} 1,2,\ldots,m-n+j-1,m-n+j+1,\ldots,m+i \\ 1,2,\ldots,\ldots,m \end{pmatrix}$$

и

$$C_{ij}^{[m]} = (-1)^{j-n+1} \frac{M_A \begin{pmatrix} 1,2,\ldots,m-n+j-1,m-n+j+1,\ldots,m+i \\ 1,2,\ldots,\ldots,m \end{pmatrix}}{M_A \begin{pmatrix} 1,2,\ldots,m \\ 1,2,\ldots,m \end{pmatrix}} \qquad (33)$$

3). $\underline{i > n-m, j \leq n-m}$. В этом случае (см. $7_в$) в миноре в числителе ф-лы (31) $(m+1)$-я строка будет иметь лишь одну минус единицу, находящуюся в $(i+m-n)$-м столбце. Разложив минор по $(m+1)$-й строке, получим, что

3-8.22 3-366

$$M_{C^{[m]}}\begin{pmatrix}1,2,\dots,m,m+i\\1,2,\dots,m,m+j\end{pmatrix}=(-1)^{m+1+i+m-n+1}M_A\begin{pmatrix}1,2,\dots,m\\1,2,\dots,m+n+i-1,m+n+i+1,\dots,m,m+j\end{pmatrix}$$

и

$$C_{ij}^{[m]}=(-1)^{i-n}\frac{M_A\begin{pmatrix}1,2,\dots,m\\1,2,\dots,m-n+i-1,m-n+i+1,\dots,m,m+j\end{pmatrix}}{M_A\begin{pmatrix}1,2,\dots,m\\1,2,\dots,m\end{pmatrix}} \qquad (34)$$

4). $i>n-m, j>n-m$. В этом случае (см. 7_g) в числителе ф-лы (31) $(m+1)$-я строка будет иметь одну минус единицу в $(j+m-n)$ столбце, а $(m+1)$-й столбец будет иметь одну единицу в $(i+m-n)$ строке. Раскладывая минор по $(m+1)$-му столбцу, а затем получившийся определитель m-го порядка по m-й строке, получим, что

$$M_{C^{[m]}}\begin{pmatrix}1,2,\dots,m,m+i\\1,2,\dots,m,m+j\end{pmatrix}=(-1)^{(m+1+j+m-n)+(1+m+i+m-n)}M_A\begin{pmatrix}1,2,\dots,m-n+j-1,m-n+j+1,\dots,m\\1,2,\dots,m-n+i-1,m-n+i+1,\dots,m\end{pmatrix}$$

и

$$C_{ij}^{[m]}=(-1)^{i+j}\frac{M_A\begin{pmatrix}1,2,\dots,m-n+j-1,m-n+j+1,\dots,m\\1,2,\dots,m-n+i-1,m-n+i+1,\dots,m\end{pmatrix}}{M_A\begin{pmatrix}1,2,\dots,m\\1,2,\dots,m\end{pmatrix}} \qquad (35)$$

Вернёмся теперь опять к схеме заполнения. У матрицы, которая получается после m-го этапа, 1-е m строк заполнены элементами значащих строк м. $X^{(m)}$ причём

$$x_{ij}^{(m)}=\begin{cases}(-1)^{i+j}\dfrac{M_A\begin{pmatrix}1,\dots,j-1,j+1,\dots,m\\1,\dots,i-1,i+1,\dots,m\end{pmatrix}}{M_A\begin{pmatrix}1,2,\dots,m\\1,2,\dots,m\end{pmatrix}} & \text{для } j\le m \qquad (9)\\[2ex] (-1)^{i+m+1}\dfrac{M_A\begin{matrix}1,\dots,\dots,m\\1,\dots,i-1,i+1,\dots,m,j\end{matrix}}{M_A\begin{pmatrix}1,2,\dots,m\\1,2,\dots,m\end{pmatrix}} & \text{для } j>m, \qquad (10)\end{cases}$$

а нижние $n-m$ строк заняты элементами матрицы $B^{(m+1)}$, причём

$$b_{ij}^{(m+1)} = (\tilde{A}_i X_j^{(m)}). \qquad (28)$$

Мы сейчас выразим $b_{ij}^{(m+1)}$ также через миноры исходной м.А.Т.к. $x_{ij}^{(m)}$ выражаются разными формулами, то и для $b_{ij}^{(m+1)}$ будет удобно рассмотреть отдельно 2 случая:

1). $j \le m,\ i > m$. В этом случае

$$b_{ij}^{(m+1)} = \sum_{k=1}^{m} (a_{ik} - \delta_{ik}) x_{kj}^{(m)} = \sum_{k=1}^{m} a_{ik} (-1)^{k+j+2m} \frac{M_A \begin{pmatrix} 1, \dots, j-1, j+1, \dots, m \\ 1, \dots, k-1, k+1, \dots, m \end{pmatrix}}{M_A \begin{pmatrix} 1,2,\dots,m \\ 1,2,\dots,m \end{pmatrix}} =$$

$$= \frac{(-1)^{j+m} \sum_{k=1}^{m} (-1)^{m+k} a_{ik} M_A \begin{pmatrix} 1, \dots, j-1, j+1, \dots, m \\ 1, \dots, k-1, k+1, \dots, m \end{pmatrix}}{M_A \begin{pmatrix} 1,2,\dots,m \\ 1,2,\dots,m \end{pmatrix}} = (-1)^{\cdots} \frac{M \begin{pmatrix} \cdots \\ \cdots \end{pmatrix}}{M \begin{pmatrix} \cdots \\ \cdots \end{pmatrix}}$$

Это выражение можно рассматривать, как определитель m-го порядка, разложенный по своей m-й строке, состоящей из m первых элементов i-й строки м. А.Т.е.

$$b_{ij}^{(m+1)} = (-1)^{j+m} \frac{M_A \begin{pmatrix} 1,2,\dots,j-1,j+1,\dots,m,i \\ 1,2,\dots,\dots,\dots,m \end{pmatrix}}{M \begin{pmatrix} 1,2,\dots,m \\ 1,2,\dots,m \end{pmatrix}} \qquad (36)$$

2). $j > m,\ i > m$. В этом случае

$$b_{ij}^{(m+1)} = \sum_{k=1}^{n} (a_{ik} - \delta_{ik}) x_{kj}^{(m)} = \sum_{k=1}^{m} a_{ik} (-1)^{k+m+1} \frac{M_A \begin{pmatrix} 1,2,\dots,m \\ 1,\dots,k-1,k+1,\dots,m,i \end{pmatrix}}{M_A \begin{pmatrix} 1,2,\dots,m \\ 1,2,\dots,m \end{pmatrix}} + a_{ij} - \delta_{ij} =$$

$$= \frac{\sum_{k=1}^{m} (-1)^{m+1+k} a_{ik} M_A \begin{pmatrix} 1,2,\dots,m \\ 1,2,\dots,k-1,k+1,\dots,m,j \end{pmatrix} + a_{ij} M_A \begin{pmatrix} 1,2,\dots,m \\ 1,2,\dots,m \end{pmatrix}}{M_A \begin{pmatrix} 1,2,\dots,m \\ 1,2,\dots,m \end{pmatrix}} - \delta_{ij}.$$

3-824 3-368

Рассматривая это выражение, как определитель $(m+1)$-го порядка, разложенный по своей $(m+1)$-й строке, получим, что

$$\beta_{ij}^{(m+1)} = \frac{M_A\begin{pmatrix}1,2,\ldots,m,i\\1,2,\ldots,m,j\end{pmatrix}}{M_A\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,m\end{pmatrix}} \qquad (37)$$

В частности,

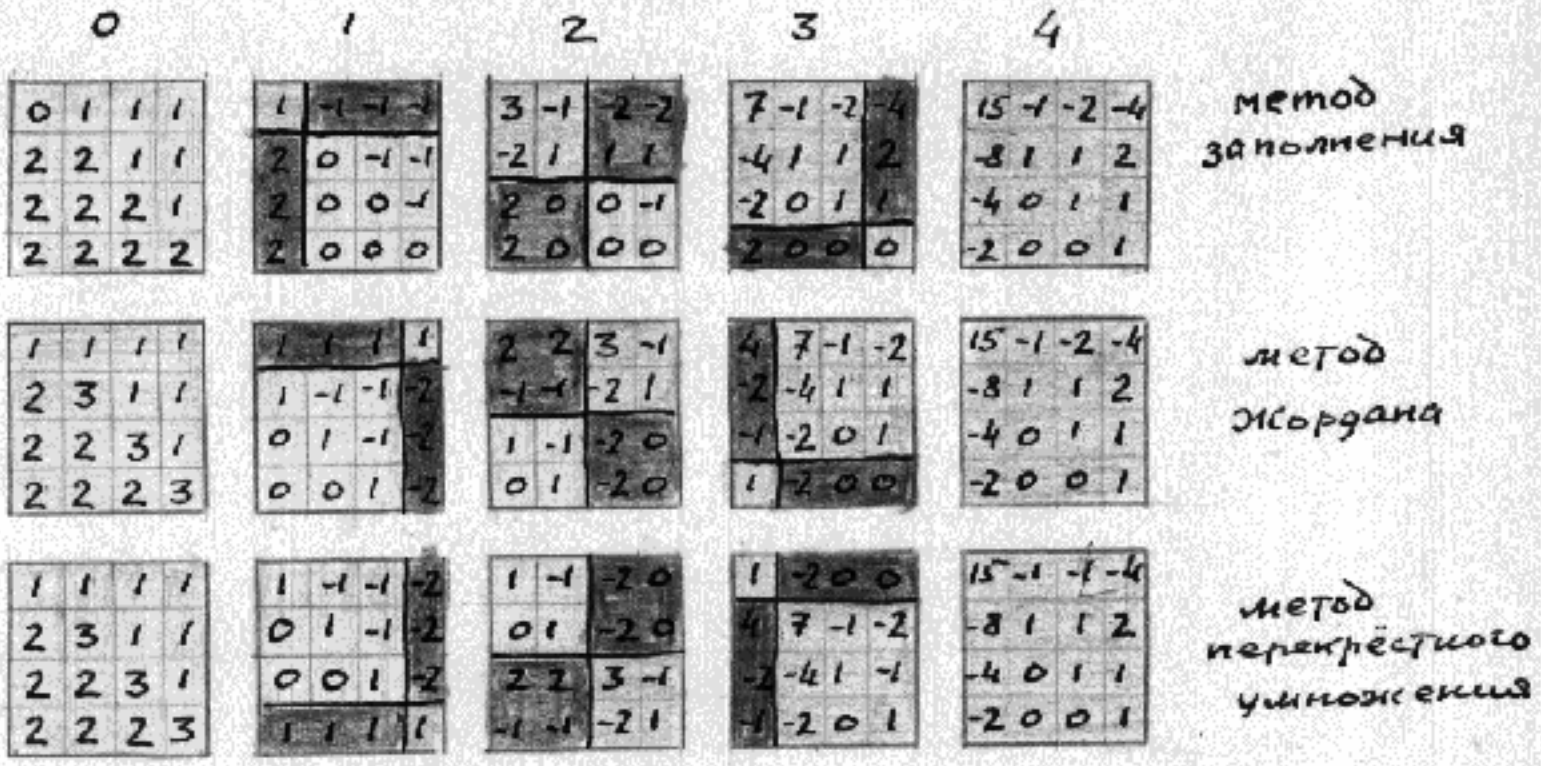
$$\beta_{m+1,m+1}^{(m+1)} = \frac{M_A\begin{pmatrix}1,2,\ldots,m,m+1\\1,2,\ldots,m,m+1\end{pmatrix}}{M_A\begin{pmatrix}1,2,\ldots,m\\1,2,\ldots,m\end{pmatrix}}$$

Мы получили формальное потверждение того, что необходимым и достаточным условием ~~об~~ возможности обращения матрицы методом заполнения является неравенство нулю главных миноров обращаемой матрицы.

Сравнивая ф-лы (32), (33), (34) и (35) с ф-лами (37), (36), (10) и (9), получим следующие соотношения между элементами матрицы $C^{[m]}$ и матриц $X^{(m)}$ и $B^{(m+1)}$:

$$c_{ij}^{[m]} = \begin{cases} \beta_{m+i,m+j}^{(m+1)} + \delta_{ij} & \text{для } i \le n-m, j \le n-m \\ -\beta_{m+i,m+j-n}^{(m+1)} & \text{для } i \le n-m, j > n-m \\ -x_{m+i-n,m+j}^{(m)} & \text{для } i > n-m, j \le n-m \\ x_{m+i-n,m+j-n}^{(m)} & \text{для } i > n-m, j \le n-m \end{cases}$$

На черт. 8 изображены промежуточные матрицы, получающиеся при вычислениях по методу заполнения, по методу перекрёстного умножения и по методу Жордана, и указаны клетки, на которые разбиваются эти матрицы и внутри которых элементы матриц либо совпадают, либо отличаются знаками, либо отличаются только диагональными элементами.



Черт.8. Голубой - отлич. диаг. эл; красный - разл. знаками розовый - совпадают

§4. ОЦЕНКА НАДЁЖНОСТИ ПОЛУЧАЕМЫХ РЕЗУЛЬТАТОВ.

В настоящем параграфе не предлагаются какие-либо новые методы проверки правильности полученных результатов и неда-ются принципиально новые оценки ошибок округления. Все приёмы, о которых тут говорится, основаны на хорошо известных и даже тривиальных свойствах обратных матриц; здесь ставится чисто практическая задача наиболее целесообразного выбора метода.

Использование оценок фон Нёймана и Тюринга. Фон Нёйманом и Гольдштейном [1] и Тюрингом [2] были даны известные оценки ошибок округления, возникающие при обращении матриц различными методами различными методами исключения. Общим для этих методов является вычисление последовательности матриц $C^{(1)}, C^{(2)}, \ldots, C^{(n)}$ по т.наз. формулам исключения

$$c_{ij}^{(m+1)} = c_{ij}^{(m)} - \frac{c_{im}^{(m)} c_{mj}^{(m)}}{c_{mm}^{(m)}}$$

Теорема §2 и результаты §3 указывают на тесную связь метода заполнения с методами исключения и позволяют сделать вывод, что для метода заполнения эти оценки тоже применимы, во всяком

3-8.26 3-370

случае,мы можем считать,что порядок роста ошибок округления будет тем же самым.Существенным недостатком этих оценок является то,что они не учитывают возможного"провала" точности из-за близости к нулю какого-нибудь главного минора м.А.

Мы укажем,однако,простой приём с помощью которого можно этот недостаток устранить:а именно, мы будем знать,во сколько раз могут увеличиться относительные ошибки округления,если какой-нибудь главный минор м. А окажется близким к нулю.

В связи с этим нам потребуется несколько предварительных замечаний.Большинство современных БЭСМ работают с "плавающей запятой" и в двоичной системе счисления,т.е. число в машине представлено в т.наз. нормальном виде

$$x = \pm a \cdot 2^{\pm b} \qquad \left(\frac{1}{2} \leq a < 1\right)$$

$\pm a$ называется цифровой частью числа,или его мантиссой,а $\pm b$ называется порядком числа.Мы будем считать правила ~~правила~~ действий над нормализованными числами известными.При выполнении какого-нибудь арифметического действия происходит округление в последнем разряде цифровой части;таким образом ошибка округления при выполнении арифметических действий над нормализованными числами характеризуется некоторой относительной ошибкой округления,которая во всех случаях,кроме,быть может,вычитания, не превышает 2^{-n},где n-число разрядов в цифровой части,т.е.,другими словами,машины с плавающей запятой производят действия с фиксированным числом значащих цифр,в отличие от машин с фиксированной запятой,которые работают с фиксированным числом двоичных знаков.

Пусть теперь δ_1 и δ_2 (Верхние оценки) относительные ошибки чисел $\tilde{a}_1$ и $\tilde{a}_2$, а a_1 и a_2 их приближённые значения,т.е.

$$|\tilde{a}_1| \leq |a_1(1+\delta_1)|\,, \quad |\tilde{a}_2| \leq |a_2(1+\delta_2)|$$

Тогда,как известно,

$$\delta_+ \leq \frac{a_1\delta_1 + a_2\delta_2}{a_1 + a_2} \qquad \delta_- \leq \frac{a_1\delta_1 + a_2\delta_2}{a_1 - a_2} \qquad (40)$$

$$\delta_\times \leq \delta_1 + \delta_2 \qquad \delta_: \leq \delta_1 + \delta_2$$

где через $\delta_+, \delta_-, \delta_:$ и $\delta_\times$ обозначены относительные ошибки, получающиеся при выполнении соответствующей операции (a_1 и a_2 считаются положительными). Таким образом в качестве верхних оценок ошибок в определении результата арифметического действия можно взять:

$$\bar{\delta}_+ = \max(\delta_1, \delta_2) \qquad \bar{\delta}_- = \frac{a_1\delta_1 + a_2\delta_2}{a_1 - a_2}$$

$$\bar{\delta}_\times = \delta_1 + \delta_2 \qquad \bar{\delta}_: = \delta_1 + \delta_2$$

Несколько подробнее исследуем случай вычитания, т.к. это единственная операция, в которой ошибка зависит от величины уменьшаемого и вычитаемого. Вынесем за скобки большее число (пусть это будет a_1), тогда

$$\bar{\delta}_- = \frac{\delta_1 + \left(\frac{a_2}{a_1}\right)\delta_2}{1 - \frac{a_2}{a_1}}$$

Рассмотрим 2 крайних случая. Пусть $a_1 \gg a_2$. Тогда

$$\bar{\delta}_- < \left[\delta_1 + \frac{a_2}{a_1}\delta_2\right]\left(1 + \frac{a_2}{a_1}\right) < \max(\delta_1, \delta_2)\left[1 + \frac{a_2}{a_1}\right]^2$$

В этом случае рост ошибки, как мы видим, весьма умерен. Но если $a_1 \simeq a_2$, то сразу из (40) получаем, что

$$\bar{\delta}_- \approx \frac{\delta_1 + \delta_2}{1 - \frac{a_2}{a_1}} \qquad (41)$$

т.е. если вычитаемое и уменьшаемое близки друг к другу, то малая разность может получиться с большой относительной ошибкой. Если ошибка при вычитании не превышает $\delta_1 + \delta_2$, то мы назовём её нормальной ошибкой вычитания. Тогда, если a_1 и a_2 имеют, скажем, k совпадающих двоичных знаков и общий порядок, то непосредственно из (41) следует, что в этом случае ошибка в вычитании возрастёт против нормальной в 2^k раз.

Основными формулами для метода заполнения являются формулы

$$x_{ij}^{(m)} = x_{ij}^{(m-1)} - \frac{x_{im}^{(m-1)} b_{mj}^{(m)}}{1 + b_{mm}^{(m)}} \qquad (42)$$

3-8.28 . 3-372

Близости к нулю m-го главного минора будет соответствовать близость к нулю $1+b_{mm}^{(m)}$ или ■ элемента $b_{mm}^{(m)}$ к -1. Нам важно знать, во сколько раз увеличивается ошибка в определении элементов матрицы из-за близости главных миноров к нулю. Из предыдущего рассмотрения мы можем сразу это узнать. Действительно, последним действием при определении знаменателя в (42) является вычитание из 1 $-b_{mm}^{(m)}$. Именно это действие может внести большую относительную ошибку, если $b_{mm}^{(m)} \approx -1$. Если получившийся знаменатель записанный в ненормализованном виде, будет иметь после запятой k_m нулей, то, как следует из (41), относительная ошибка увеличится в 2^{k_m} раз ($a_i = 1$).

Во время обращения получающиеся $1+b_{mm}^{(m)} (m = 1, \ldots, n)$ можно печатать. Каждому $1+b_{mm}^{(m)}$ будет соответствовать своё число k_m. Тогда можно сформулировать такое правило:

Если оценки фон Неймана или Тюринга позволяют полагать, что после вычислений элементы обратной матрицы будут иметь τ верных двоичных знаков, то на самом деле число верных знаков будет не меньше, чем

$$\tau - \sum_{m=1}^{n} k_m \qquad (43)$$

Разумеется, это довольно грубая оценка и не следует доверять ни очень плохим результатам. В таком случае нужно провести дополнительное исследование получившихся результатов, с тем чтобы точнее оценить полученную погрешность.

Методы проверки, требующие дополнительных вычислений. Эти методы основаны на следующих свойствах матриц:

1)
$$AA^{-1} = E \qquad (44)$$

2)
$$(A^{-1})^{-1} = A \qquad (45)$$

3) Обозначим сумму элементов i-й строки прямой матрицы A через a_i и сумму элементов j-го столбца точной обратной матрицы X через x_j, т.е.

$$a_k = \sum_{j=1}^{n} a_{kj}$$
$$x_k = \sum_{i=1}^{n} x_{ik} \qquad (46)$$

3-8.29 3-373

a_k можно рассматривать как k-ю компоненту вектора $\vec{A}$ и x_k-как k-ю компоненту вектора $\vec{X}$. Тогда

$$(\vec{A}\,\vec{X}) = n\,, \text{ где } n \text{ - порядок матрицы.} \qquad (47)$$

Действительно, пусть XA = C = E. Тогда

$$(\vec{A}\,\vec{X}) = \sum_{k=1}^{n}\Big(\sum_{j=1}^{n} a_{kj}\Big)\Big(\sum_{i=1}^{n} x_{ik}\Big) =$$

$$= \sum_{j=1}^{n}\sum_{i=1}^{n}\Big(\sum_{k=1}^{n} x_{ik}\cdot a_{kj}\Big) = \sum_{j=1}^{n}\sum_{i=1}^{n} c_{ij} = n$$

1. Подсчёт AX. Анализ матрицы AX даёт, пожалуй, наиболее точное представление об ошибках, с которыми вычислены элементы обратной матрицы. Кроме того, вычисление этой матрицы совершенно необходимо, если по смыслу задачи, приведшей к обращению матрицы, нам важно не столько точное знание элементов обратной матрицы X, сколько возможно более близкое приближение матрицы AX к E. Совершенно ясно, что, особенно в случае плохо обусловленных матриц, эти два требования могут оказаться неэквивалентными. Знание матрицы AX позволяет улучшить значение вычисленной обратной матрицы по известным формулам Готеллинга [3]. Однако, недостатком этого метода является необходимость подсчёта произведения матриц. В случае малых возможностей вычислительной машины выполнение этого умножения может оказаться затруднительным, т.к. здесь в вычислениях участвуют сразу две матрицы.

2. Образование матрицы, обратной к вычисленной обратной матрице. Такая проверка хороша тем, что её очень просто осуществить, т.к. для этого достаточно по окончании вычислений, "запустить" снова программу, не меняя никаких установок в машине. Однако этот метод будет давать хорошие результаты лишь в том случае, если мы можем утверждать, что прямая и обратная матрицы примерно одинаково обусловлены. От этого ограничения можно отчасти избавиться, если совместить этот метод проверки с исследованием получающихся главных миноров так, как об этом говорилось выше.

3. Вычисление скалярного произведения $(\vec{A},\vec{X})$. Этот метод проверки удобен тем, что требует сравнительно малого объёма дополнительных вычислений. Связь его с 1-м способом проверки выража-

ется в том,что он даёт нам сумму элементов матрицы ХА.Однако при ответственных вычислениях этот метод оказывается недостаточным,т.к. погрешности в элементах м.ХА могут взаимно уничтожаться.

§5. ПРОГРАММА ОБРАЩЕНИЯ МАТРИЦ МЕТОДОМ ЗАПОЛНЕНИЯ

Предварительные замечания. Программа будет составлена для условной трёхадресной машины обычного типа со следующими оперативными возможностями ($[\alpha]$ обозначает содержимое ячейки α).

	Вид команды				Её содержательное значение
1	+	α	β	γ	$[\alpha]+[\beta]=[\gamma]$
2	−	α	β	γ	$[\alpha]-[\beta]=[\gamma]$
3	×	α	β	γ	$[\alpha]\cdot[\beta]=[\gamma]$
4	:	α	β	γ	$[\alpha]:[\beta]=[\gamma]$
5	Пер	α		γ	$[\alpha]\longrightarrow[\gamma]$
6	+′	α	β	γ	К цифровой части α прибавить цифр.часть β ;рез-тат с порядком числа α поставить в γ
7	−′	α	β	γ	Вычитание,аналогичное 6-й ком-де
8	$\leq$	α	β	γ	Если $[\alpha]<[\beta]$,то переход к ком. $[\gamma]$, если $[\alpha]\geqslant[\beta]$,то переход к следующей команде.
9	стоп				Останов машины.

Программа будет сделана восстанавливающейся,что будет достигаться тем,что начальное значение каждой переменной команды,прежде чем она будет выполняться,заносится командой (5) на своё место в программе.Изменение адресов переменных команд производится командами (6) и (7).Для повторения цикла программы столько раз ,сколько это нужно,переменная команда,рассматриваемая как число,сравнивается со своим значением,которое она принимает по окончании цикла.

1-й вариант программы получается сразу из непосредственного осуществления вычислительной схемы,описанной в конце §2. Пусть матрица А находится в ячейках с $\alpha+1$ до $\alpha+n^2$ (нумерация по строкам),скалярные произведения - в ячейках с $\beta+1$ по $\beta+n$.На каждом,m-м этапе,нам надо осуществить следующие операции:

3-8.31 . 3-375

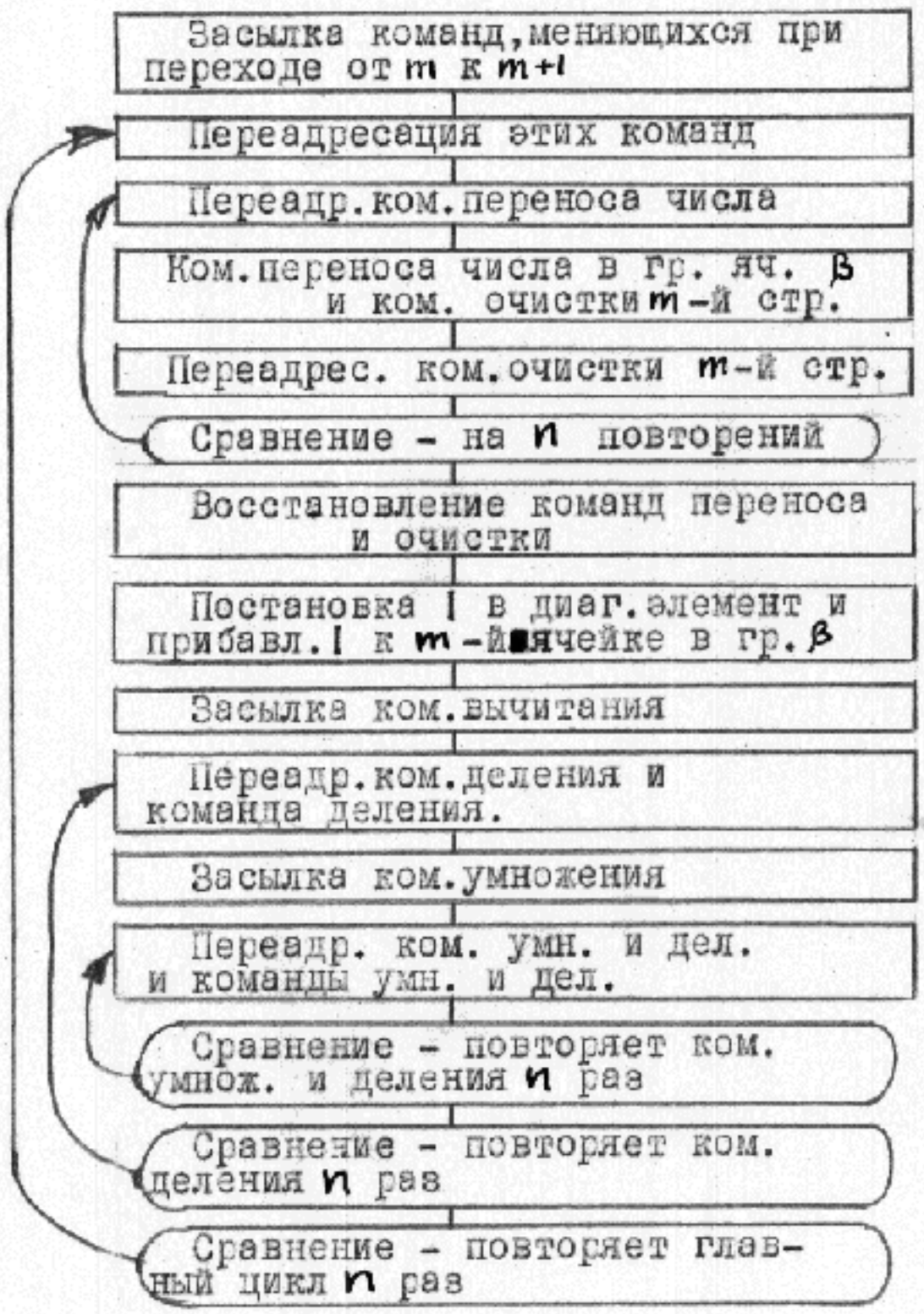
1.Снести m -ю строку м.А в ячейки $\beta - \beta+n$

2.Очистить m-ю строку м.А и поставить 1 в диагональную ячейку.

3.Прибавить 1 к m-ой ячейке в группе $\beta - \beta+n$

4.Провести вычисления по ф-лам (26)

Получается следующая блок-схема программы:



	места команд.
Засылка команд, меняющихся при переходе от m к m+1	K – K+4
Переадресация этих команд	K+5 – K+7
Переадр.ком.переноса числа	K+8
Ком.переноса числа в гр. яч. β и ком. очистки m-й стр.	K+9 , K+10
Переадрес. ком.очистки m-й стр.	K+11
Сравнение - на n повторений	K+12
Восстановление команд переноса и очистки	K+13 , K+14
Постановка 1 в диаг.элемент и прибавл.1 к m-йячейке в гр.β	K+15 , K+16
Засылка ком.вычитания	K+18
Переадр.ком.деления и команда деления.	K+19 , K+20
Засылка ком.умножения	K+21
Переадр. ком. умн. и дел. и команды умн. и дел.	K+22 – K+25
Сравнение - повторяет ком. умнож. и деления n раз	K+26
Сравнение - повторяет ком. деления n раз	K+27
Сравнение - повторяет главный цикл n раз	K+28
СТОП	K+29

Замечания к программе:

1) ℓ, $\ell+1$, $\ell+2$ - ячейки для промежуточных результатов

2) Ячейка К+17 оставлена пустой;в ней может быть помещена команда отсылки к программе печатания вычисленного главного минора.

3)Оперативная часть программы содержит всего 28 команд. Для обращения одной матрицы нужно произвести $5n^3+9n^2+9n+5$ операций.

4)При обозрении этой программы особенно заметно преимущество метода заполнения при обращении матриц на БЭСМ.Программа требует n^2+n ячеек и соединяет с малым количеством команд простоту вычислительной схемы и малое время обращения (порядок числа операций - $5n^3+ O(n^2)$).

2-й вариант позволяет обратить матрицу,используя всего n^2 ячеек оперативной памяти.Саму программу мы приведём во 2-й главе,а здесь укажем лишь,как должна быть видоизменена вычислительная схема.В отличие от первого варианта подсчёт новых элементов матрицы будет вестись не по строчкам,а по столбцам, и вычисления на m-м этапе организуются следующим образом:

1.Прибавление 1 к $b^{(m)}_{m,m}$ и вычисление всего главного столбца.

2.Снос 1-го элемента m-й строки в ячейку памяти,очистка соответствующей ячейки в матрице и подсчёт 1-го столбца новой матрицы.

Снос 2-го элемента m-й строки в ту же ячейку памяти, очистка соответствующей ячейки в матрице и подсчёт 2-го столбца новой матрицы.

3.Так мы подсчитываем все столбцы новой матрицы,кроме главного,который уже вычислен и который при 2-й группе вычислений должен быть пропущен.

Мы видим,что программа по сравнению с первым вариантом усложнится лишь тем,что придётся ввести ещё одно переменное сравнение,которое обеспечит "пропуск" главного,m-го, столбца при m-м этапе вычислений.Т.к. программа во 2-й главе приводится без примечаний,отметим сразу,что программа в целом занимает 61 ячейку памяти (оперативная часть - 35 команд);обращение одной матрицы требует $5n^3+ n^2+ n+$ операций.

3-377

3-8.33

§6. ЧИСЛЕННЫЕ ПРИМЕРЫ

I). Ниже приводится пример, иллюстрирующий вычислительную схему метода заполнения при ручном счёте и показывающий, насколько точнее окончательный вариант вычислительной схемы из §2 по сравнению с первоначальным, когда скалярные произведения подсчитывались непосредственно. В качестве исходной была взята матрица из книги Фаддеевой "Вычислительные методы линейной алгебры", 1950 г. (стр.91).

Вычислительная схема метода заполнения при ручном счёте.

E,	0,	0,	0,
0,00	0,42	0,54	0,66
0,42	0,00	0,32	0,44
0,54	0,32	0,00	0,22
0,66	0,44	0,22	0,00
1,00	0,42	0,54	0,66
1,00	-0,42	-0,54	-0,66
0,	1,	0,	0,
0,42	-0,1764	0,0932	0,1628
0,54	0,0932	-0,2916	-0,1364
0,66	0,1268	-0,1364	-0,4856
0,42	0,8236	0,0932	0,1628
1,2141 8164	-0,5099 5629	-0,4924 7207	-0,5769 7912
-0,5099 5627	1,2141 8160	-0,1131 6173	-0,1976 6876
0,	0,	1,	0,
0,4924 7207	0, 1131 6173	-0,3021 4667	-0,1548 2273
00,5769 7912	0,1976 6877	-0,1548 2273	-0,4677 8048
0,4924 7207	0,1131 6173	0,6978 5333	-0,1548 2273
1,5617 1705	-0,4300 9855	-0,7056 9567	-0,6862 3685
-0,4300 9853	1,2325 3156	-0,1621 5690	-0,2227 7433
-0,7056 9568	-0,1621 5690	1,4329 6590	0,2218 5569
0,	0,	0,	1,
0,6862 3685	0,2227 7434	-0,2218 5569	-0,5021 2878
0,6862 3685	0,2227 7434	-0,2218 5569	0,4978 7122
2,5075 8619	-0,1230 3930	-1,0114 8871	-1,3783 4210
-0,1230 3930	1,3322 1277	-0,2614 2705	-0,4474 5372
-1,0114 8872	-0,2614 2706	1,5318 2670	0,4456 0859
-1,3783 4205	-0,4474 5373	0,4456 0858	2,0085 5150

3-8.34 3-378

Результаты, полученные при подсчёте по первоначальному варианту вычислительной схемы:

2,5075 8695	-0,1230 4342	-1,0114 8844	-1,3783 4080
-0,1230 4343	1,3322 1068	-0,2614 2047	-0,4474 5152
-1,0144 8840	-0,2614 2046	0,5218 2410	0,4456 0608
-1,3783 4080	-0,4474 5153	0,4456 0607	2,0085 5029

Ошибка в первом случае:

$$10^{-8}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & -2 \\ 1 & 1 & 0 & 2 \\ -1 & -1 & 1 & 0 \end{pmatrix}$$

Ошибка во втором случае:

$$10^{-8}\begin{pmatrix} -75 & 402 & -87 & -130 \\ 403 & 211 & -658 & -223 \\ -41 & -659 & 260 & -249 \\ -130 & -223 & -250 & 131 \end{pmatrix}$$

2). Методом заполнения было обращено несколько несимметричных хорошо обусловленных матриц 27-го порядка. Вычисления велись с 9-ю значащими цифрами. В большинстве случаев ответ имел 6 верных знаков. Для контроля правильности вычислений применялось обращение обратной матрицы и подсчёт скалярного произведения $(\vec{A}\cdot\vec{X})$, что и позволило сделать выводы о числе верных знаков.

ЛИТЕРАТУРА

[1] J.v. Neumann and H.H. Goldstine Numerical inverting of matrices of high order.
Bull. Amer. Math. Soc. 53, №11(1947), 1021 (см. также УМН, VI, №4(1951), 123.

[2] Тюринг Ошибки округления в матричных процессах.
УМН, VI, №1 (1951)

[3] Фаддеева Вычислительные методы линейной алгебры
Москва, 1950

[4] Фокс, Хаски, Вилкинсон Заметки о решении систем алгебраических совместных системах уравнений
УМН, V, №3, (1951)

[5] П.Ф. Гроссман К задаче численного решения систем совместных линейных алгебраических уравнений.
УМН, V, №3(1951), 87.

3-8.35 3-379

ГЛАВА 2

СОСТАВЛЕНИЕ ПРОГРАММ ДЛЯ БЫСТРОДЕЙСТВУЮЩИХ ЭЛЕКТРОННЫХ СЧЁТНЫХ МАШИН

§1. НЕКОТОРЫЕ ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ.

Программист при составлениипрограммы обычно придерживается следующего порядка действий:

(1).Получив задание и выбрав [redacted] численный метод, которым будет решаться данная задача,он определяет,какие арифметические действия и в какой последовательности должны выполняться, - т.е. он составляет схему счёта.

(2).В соответствии со схемой счёта программист определяет логические условия,при которых должны выполняться те или иные этапы вычислений,выясняет связь и взаимозависимость этих логических условий и составляет блок-схему программы. Блок-схема отличается от схемы счёта тем,что в ней различные этапы арифметических действий,или операторы счёта,указаны не в том порядке,в каком они выполняются,а так,как они должны стоять в программе.Кроме того,блок-схема дополняется логическими командами условного и безусловного перехода,а также командами управления,о которых мы скажем ниже.

(3).После программист расписывает программу по отдельным командам,производит распределение ячеек памяти и заполняет адреса команд в соответствии с этим распределением.

Расписы■вание адресов программы по ячейкам - процесс утомительный и неинтересный,но в тоже время требующий напряжённого внимания.При проверке программ оказывается,что ■ ошибки и описки при расписывании по ячейкам составляют большой процент от всех возможных ошибок программы.В связи с этим во 2-м параграфе предлагается рациональная схема действий при заполнении адресов программы,которая позволяет сделать этот процесс почти механическим и резко снизить

количество ошибок за счёт расписывания по ячейкам.

Другой существенной стороной работы программиста является использование стандартных подпрограмм. Для того чтобы включить стандартную подпрограмму в основную программу, если только она не находится на раз и навсегда отведённых для неё ячеек памяти в качестве элемента, "встроенного" в машину, программист должен скопировать программу с некоторого образца, хранящегося в библиотеке стандартных программ, заполнить те адреса подпрограммы, которые зависят от её места в оперативной памяти, приготовить программу для ввода и, наконец, ввести её в машину. Переписка, простановка новых адресов и приготовление программы для ввода в машину - все эти этапы, как правило, вносят много ошибок и тем самым почти сводят на нет преимущества использования готовых стандартных подпрограмм.

В 3-ем параграфе предлагается вариант программы, которая производит автоматически ввод и объединение стандартных подпрограмм в главной программе на самой машине, причём полностью исключаются ошибки, которые происходят при "ручном" вводе подпрограмм в машину.

Так же, как и в первой главе мы будем иметь дело с условной трёхадресной машиной, в отношении которой мы будем дополнительно считать известным следующее:

I. Машина имеет две равноправных системы управления, $У_1$ и $У_2$. В связи с этим наш код операций пополнится командами перехода от одной системы управления к другой:

$ПУ_1$	--	--	α
$ПУ_2$	--	--	α

Эти команды являются командами безусловной передачи управления на I-е или 2-е управления.

ВУ	--	--	--
ВУ	--	--	--

Эти команды являются командами возврата на I-е или 2-е управление. При выполнении этих команд происходит возврат, напр., на первое управление к команде, следующей за командой, осуществившей переход на второе управление.

2.В отношении ввода программ в машину мы будем считать,что окончательно готовая программа определённым образом кодируется,затем каждая команда пробивается на ленту и затем с перфоленты программа вводится в машину.Кроме того,некоторое количество чисел может вводиться в оперативную память вручную.

В заключение введём некоторую терминологию в отношении составных элементов программы.

С точки зрения своей структуры все команды программы делятся на постоянные и переменные команды.Переменные команды - это команды,чьи адреса меняются по ходу решения задачи. Их адреса меняются прибавлением или вычитанием из адресной части команды некоторых чисел,которые мы назовём константами переадресации. Команды,которые осуществляют переадресацию переменных команд, называются командами переадресации. Самовосстанавливаемость программы достигается тем,что в памяти в специальных ячейках хранятся начальные значения переменных команд.Постановка начальных значений переменных команд осуществляется командами засылки.Работа команды сравнения,или просто сравнения,повторяющего некоторый цикл вычислений нужное число раз,заключается в том, что некоторое переменное число,или переменная сравнения x , сравнивается или по своей величине,или по модулю с некоторым числом,называемым константой сравнения X.В качестве переменной сравнения берётся или текущее значение переменной команды,которое зависит от числа повторений цикла k ,или само число k ,которое в этом случае нужно подсчитывать в особой ячейке,называемой счётчиком.Мы будем считать(обычно это так и бывает),что с ростом k переменная сравнения тоже растёт.Тогда,в соответствии с нашим определением команды сравнения,если x стоит в 1-м адресе сравнения, а X - во 2-м,то при $x<$ X будет происходить "скачок",а при $x\geqslant$ X произойдёт переход к следующей за сравнением команде,Такую работу сравнения мы назовём нормальной. Если же x стоит во 2-м адресе,а X - 1-м,то при $x\leqslant$ X будет происходить переход к следующей команде,а при $x>$ X произойдёт скачок.Такую работу сравнения мы назовём инверсной.

С точки зрения своего содержательного значения все команды программы делятся на арифметические,т.е. команды,которые непосредственно выполняют нужные арифметические действия,и команды управления.Команды управления делятся на логические - сравне-

ния и команды передачи управления - и собственно команды управления,к которым относятся команды засылки и переадресации.Заметим,что для большего однообразия полезно отождествлять счётчик с переменной командой,тогда команда очистки счётчика будет соответствовать команде засылки,а прибавление единицы - команде переадресации.

Из предыдущего следует,что каждая программа,не считая начальных величин,нужных для решения задачи,содержит кроме оперативной части ещё начальные значения переменных команд,константы переадресации,и константы сравнения,образующие вспомогательную часть программы.

§2. ПОРЯДОК РАСПИСЫВАНИЯ ПРОГРАММЫ ПО ЯЧЕЙКАМ ОПЕРАТИВНОЙ ПАМЯТИ.

Исходным этапом нашей работы является полностью составленная оперативная часть программы,записанная на бланке №I следующим образом

I	II	III	IV	V	VI_1	VI_2	VI_3	VII

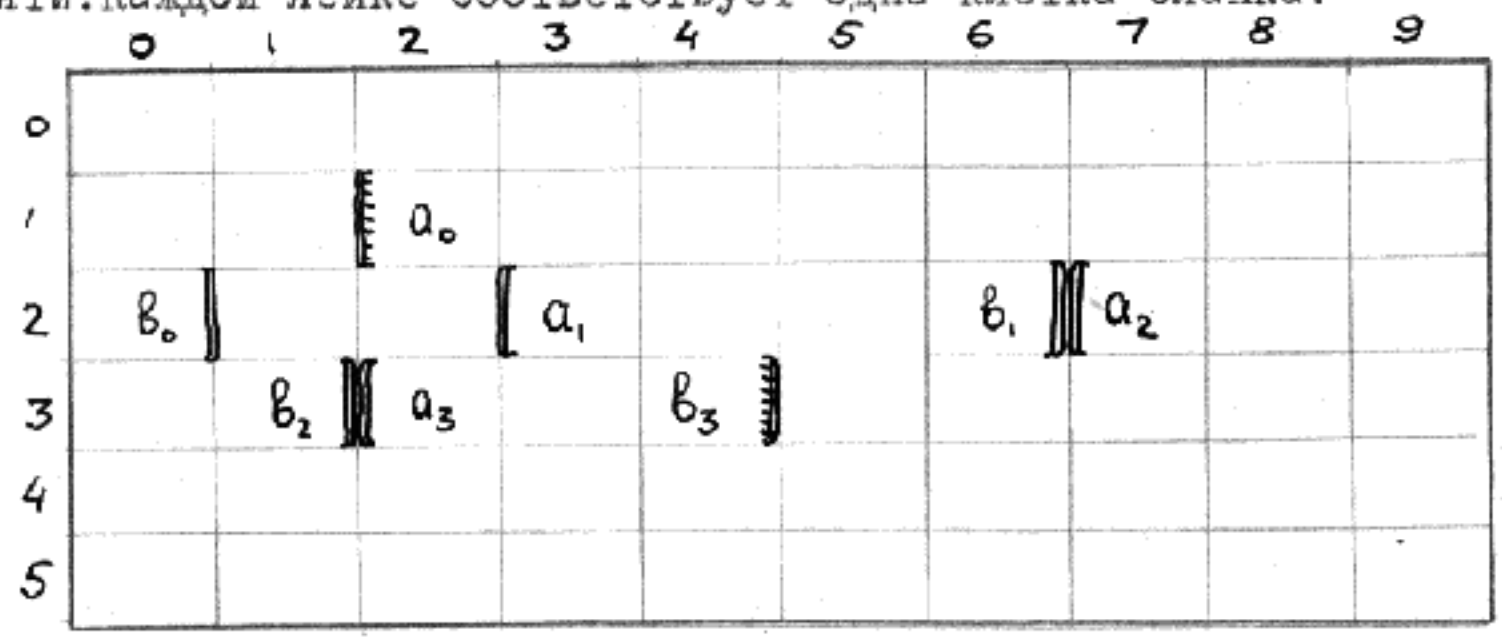
Бланк № I.

Каждая строка бланка соответствует одной ячейке памяти.В графе III словами выписано содержательное значение каждой команды.Все передачи управления,происходящие вследствие выполнения команд сравнения и безусловного перехода,чётко указываются стрелками,расположенными в графе IV. В каждом сравнении должно быть указано,нормальное оно или инверсное,а также указана переменная сравнения.

3-3.39 3-323

НА 1-М ЭТАПЕ сортируем все команды программы на: 1)постоянные арифметические команды,2)переменные команды,3)команды засылки, 4)команды переадресации,5а)нормальные сравнения,5б)инверсные сравнения,6)команды безусловной передачи,отмечая команду каждой из шести групп чётким признаком,который ставится в графу IV . Кроме того,в переменных командах находим переменные адреса,отмечаем их скобками в графах VI_1 , VI_2, VI_3 ,соответствующих 1-му, 2-му и 3-муадресам,и в скобки ставим число,на которое меняется адрес при повторении цикла.Тем самым мы находим вид константы переадресации.Если различные адреса программы по-разному меняются в разных циклах,то это обстоятельство нужно ясно отмечать. В связи с этим иногда целесообразно вид констант переадресации писать не только в ячейках переменных команд,но ещё и в графе VII напротив соответствующих команд переадресации.Т.к. строки переменных команд в графах V и VI играют особую роль и заполняться адресами не будут,их полезно отличать каким-нибудь цветом. На этом же этапе мы ставим в графу V коды операций команд управления в соответствии с нашей сортировкой.

НА 2-М ЭТАПЕ мы подсчитываем число начальных значений команд (по числу переменных команд),число констант переадресации,число констант сравнения (по числу команд сравнения) и тем самым точно определяем объём программы.После этого мы намечаем ячейки памяти,в которых будет находиться программа.Это делается с помощью бланка № 2,который представляет собой схему загрузки памяти.Каждой ячейке соответствует одна клетка бланка.



Бланк № 2.

После этого в графе I нумеруем все строки в соответствии с расположением программы в памяти. Пусть после этой нумерации оперативная часть программы находится в ячейках с a_0 по b_0, константы переадресации - с a_1 по b_1, начальные значения в ячейках с a_2 по b_2 и константы сравнения - с a_3 по b_3.

На этом подготовительная часть работы закончена, и начинается непосредственное заполнение программных граф (V и VI).

НА 3-ЕМ ЭТАПЕ мы заполняем [redacted] 1-е и 3-и адреса команд переадресации, 3-и адреса команд засылки, 1-е адреса нормальных и 2-е адреса инверсных сравнений в соответствии с номерами переменных команд и 3-и адреса команд сравнений и безусловных переходов в соответствии с указанием логических связей в графе .

НА 4-М ЭТАПЕ мы заполняем остальные адреса команд управления. 2-е адреса команд переадресации мы заполняем в соответствии с расстановкой констант переадресации, в ячейках $[a_1, b_1]$, которая тут же и производится; 1-е адреса команд засылки нумеруем цифрами от a_2 до b_2 и, наконец, 2-е адреса нормальных и 1-е адреса инверсных сравнений нумеруем цифрами от a_3 до b_3.

НА 5-М ЭТАПЕ мы готовимся к заполнению вспомогательной части программы. Предварительно мы должны определить, к каким ячейкам оперативной части программы относятся ячейки вспомогательной части. Заметим, что в ячейках $[a_1, b_1]$ константы переадресации уже расставлены.

Поступаем следующим образом. Отмечаем ячейки $[a_2, b_2]$ в программной части бланка и в графе VII ставим 1-е число, равное номеру команды засылки, в которой данная ячейка из $[a_2, b_2]$ используется, а 2-е число в скобках - номер переменной комнады, чьё начальное значение хранится в этой ячейке, т.е. 1-е число - это номер команды засылки, а 2-е число - это 3-й адрес в этой команде. Таким же образом отмечаем ячейки $[a_3, b_3]$ на программной части бланка № 1 и в графе VII ставим 1-е число, равное номеру команды сравнения, для которого содержимое данной ячейки является константой сравнения, и 2-е число в скобках - адрес переменной сравнения.

Затем мы повторяем эту процедуру в точности, только ставим 1-е и 2-е числа не в графу VII вспомогательной части программы,

3-8.41 3-385

я в клетки бланка № 2, соответствующие ячейкам $[a_2 b_2]$ и $[a_3 b_3]$. Затем мы сравниваем между собой соответствующие ячейки вспомогательной части программы и клетки бланка № 2 и если видим, что выписанные числа совпадают, считаем, что эта часть работы сделана правильно. Этот этап работы дублируется потому, что, по нашему мнению, определение связей между оперативной и вспомогательной частями программы является одним из самых ответственных этапов, обычно вносящим много ошибок.

НА 6-М ЭТАПЕ мы распределяем ячейки, используемые в арифметических командах, и заполняем арифметические команды и вспомогательную часть программы.

На вкладке изображена последовательность работы при составлении программы обращения матрицы на η^2 ячейках (см. гл. I, §5) методом заполнения.

Этим методом автором было составлено 8 самых различных программ с числом команд от 35 до 250 с общим объёмом, включая вспомогательную часть программ, примерно в 1000 команд. В общей сложности в программах было найдено 16 ошибок, из них ошибок, полученных за счёт расписывания по ячейкам, оказалось только 2, причём обе они были допущены [illegible] на 5-м этапе составления программы, когда дублирования работы ещё не производилось.

Удобством такого порядка составления программ является разделение всего процесса работы на расчленённые этапы, каждый из которых легко контролируется. Кроме того, превращение расписывания по ячейкам в твёрдо установленную последовательность действий, совершаемую по точно сформулированным правилам, исключает влияние индивидуальности программиста на ход составления программы и позволяет производить эту работу двум разным людям и использовать, таким образом, все преимущества независимой работы в две руки для целей контроля. Кроме того, т.к. почти весь процесс не связан с реальным содержанием программы, построение всех управляющих команд можно поручать менее квалифицированным работникам. Для этого, конечно, необходима разработка точной терминологии или какой-нибудь символики для правильного и однозначного прочтения содержимого III -ей графы программного бланка.

Кроме того, дальнейшее развитие подобных идей позволит превратить это словесное описание правил работы в точный и матема-

3-8.42 - 3-386

тически формулируемый алгорифм, для которого можно составить программу, заставив, тем самым, саму машину готовить для себя программы.

Однако на пути к решению этой проблемы стоят нерешёнными две задачи. Это:

1) Выбор удобной и гибкой аналитической записи программы с указанием логических связей и взаимного расположения переменных команд и команд управления в программе, - записи, позволяющей перевести на "машинный язык" то что мы сейчас изображаем словами и стрелками на программном бланке.

2) Формализация всех этапов работы по расписыванию программы и превращение этих правил в программируемый алгорифм.

Отметим, что наиболее трудной и существенной задачей является первая, от решения которой зависит весь успех дела. За последнее время наметились некоторые успехи в решении этой задачи на пути использования теории рекурсивных функций, однако о результатах говорить ещё рано.

§3. ОПИСАНИЕ ПРОГРАММЫ, ВЫПОЛНЯЮЩЕЙ ОБЪЕДИНЕНИЕ И ВВОД СТАНДАРТНЫХ ПОДПРОГРАММ В БЭСМ.

Все стандартные программы, имеющиеся при некоторой БЭСМ, мы будем делить на три категории: 1) Стандартные программы, составленные раз навсегда для решения некоторой самостоятельной задачи. Они вводятся в машину всегда на одно и то же место в памяти и используют для решения задачи всегда одни и те же ячейки оперативной памяти. Для того, чтобы этой программой можно было пользоваться для решения различных задач данного класса, например, для различного числа уравнений, мы будем считать, что в программу входит некоторый специальный параметр n, который может означать число уравнений, порядок матрицы, число членов ряда и т.п. в зависимости от конкретного содержания задачи. После ввода стандартной программы в машину значение специального параметра вводится в машину вручную. Для того, чтобы количество ручных вводов было минимальным, вручную параметр вводится только в одну ячейку памяти, а программа сама вычисляет из этой "заготовки" все необходимые числа и расставляет их на нужные места.

2) Встроенные программы, не имеющие специальных параметров,

3-843 3-387

введённые в машину раз навсегда и использующие в своей работе всегда одни и те же ячейки памяти.

Использование стандартных и встроенных программ не представляет никаких затруднений.

3) Стандартные подпрограммы - это заранее составленные вспомогательные программы, входящие как часть в некоторую главную программу. Они могут помещаться в машину на разные места памяти, использовать в своей работе различные от задачи к задаче ячейки оперативной памяти и содержать специальные параметры. О них и будет итти речь в этом параграфе.

Прежде, чем точно поставить задачу, сформулируем требования, которые мы предъявим к стандартным подпрограммам.

1. Подпрограмма содержит в себе все константы, необходимые для выполнения той вспомогательной задачи, которую решает данная подпрограмма.

2. Оперативная часть программы всегда кончается командой передачи управления на центральную программу. Это будет команда возврата на 1-е управление (предполагается, что главная программа работает на первом управлении).

3. Подпрограмма занимает ячейки памяти, следующие друг за другом и начинающиеся с номера К. Оперативные ячейки, которые используются подпрограммой, тоже расположены подряд и начинаются с номера α.

4. Подпрограмма содержит не больше одного специального параметра n, который, как входное данное, ставится вручную тоже только в одну ячейку. Мы будем считать, что это параметр, там, где он есть, ставится в ячейку К, а оперативная часть программы в этом случае начинается с ячейки К+1.

Задача ставится следующим образом:

В главной программе должны быть использованы m различных стандартных подпрограмм. Считаются известными параметры подпрограмм: $K_1, \alpha_1, n_1;\ K_2, \alpha_2, n_2; \ldots; K_m, \alpha_m, n_m$ и начало расположения введённых подпрограмм.

Требуется составить программу, осуществляющую автоматически ввод программ в машину, размещение стандартных подпрограмм на своих местах и подгонку адресов в соответствии с расположением подпрограммы и выбором ячеек оперативной памяти.

3-844 3-388

В связи с построением такой программы мы дополнительно введём для простоты в код машины ещё две операции – сдвиг влево и сдвиг вправо:

← [α] ℓ [β]
→ [α] ℓ [β]

При выполнении этой команды число [α] сдвигается влево или вправо на ℓ разрядов и результат ставится в β .

Рассмотрим в качестве примера стандартную подпрограмму суммирования n чисел, расположенных подряд в ячейках, начиная с α+1 с постановкой суммы в α . Сама программа находится в ячейках, начиная с К. Тогда, по нашим правилам построения программ эта подпрограмма будет выглядеть следующим образом

				n
К				
К+1	←	К	ℓ	К+12
К+2	+	К+11	К+12	К+11
К+3	−	α	α	α
К+4	Пер	К+9		К+6
К+5	+	К+6	К+10	К+6
К+6				
К+7	≤	К+6	К+11	К+9
К+8	ВУ1			
К+9	+	α	α	α
К+10		0	1	0
К+11	+	α	α+■	α
К+12				

(здесь ℓ – число разрядов одного адреса)

Цифровую часть каждой команды такой программы можно рассматривать как сумму двух чисел – одного, независящего от К и α, и характеризующего, так сказать, внутренние свойства команды, и другого, зависящего от К и α и характеризующего связь программы с оперативной памятью. Например, команду К+7 можно представить как сумму таких чисел (суммирование понимается как операция сложение цифровых частей):

К+7	{	6	10	5
		К	К	К

То же самое можно сказать о любой команде, входящей в программу. Назовём первое число – <u>стандартным значением команды</u>, а второе число назовём <u>приводящим числом для данной команды</u>. Тогда процесс подгонки стандартной программы к данным значениям К и α

3-8.45 3-389

можно рассматривать, как прибавление к стандартным значениям команд соответствующих им приводящих чисел. Для программы суммирования стандартные значения и приводящие числа будут выглядеть следующим образом:

К		0	0	И	0	0	0
К+1	←	0	ℓ	12	К	0	К
К+2	+	10	11	10	К	К	К
СК+3	−	0	0	0	α	α	α
К+4	Пер	8	0	6	К	0	К
К+5	+	6	9	6	К	К	К
К+6		0	0	0	0	0	0
К+7	≤	6	10	5	К	К	К
К+8	$ВУ_1$	0	0	0	0	0	0
К+9	+	0	0	0	α	α	α
К+10		0	1	0	0	0	0
К+11	+	0	0	0	α	α	α
К+12		0	0	0	0	0	0

Рассмотрим структуру приводящих чисел. Очевидно, что, вообще, они могут быть такого вида:

1)	К	К	К	9)	К	К	0	15)	К	α	0	21)	0	0	К
2)	К	К	α	10)	К	0	К	16)	К	0	α	22)	0	К	0
3)	К	α	К	11)	0	К	К	17)	0	К	α	23)	К	0	0
4)	α	К	К	12)	α	α	0	18)	α	К	0	24)	0	0	α
5)	К	α	α	13)	α	0	α	19)	α	0	К	25)	0	α	0
6)	α	К	α	14)	0	α	α	20)	0	α	К	26)	α	0	0
7)	α	α	К									27)	0	0	0
8)	α	α	α												

Однако в программе могут встречаться приводящие числа только такого вида:

1)	К	К	К	7)	α	α	К	13)	0	0	α
2)	К	К	α	8)	α	α	α	14)	α	0	α
3)	К	α	К	9)	К	0	К	15)	0	0	0
4)	α	К	К	10)	К	0	α				
5)	К	α	α	11)	α	0	К				
6)	α	К	α	12)	0	0	К				

Принцип действия предлагаемой программы, которую мы назовём приводящей программой, состоит в том, что она для каждой вводимой подпрограммы формирует из чисел К и α все 15 при-

водящих чисел,а затем осуществляет сложение стандартных значений команд с нужными приводящими числами.

Опишем программу подробнее.

Мы будем считать,что приводящая программа вводится в машину на одни ите же места памяти и использует в качестве рабочих также одни и те же ячейки памяти.Пусть приводящие числа будут образовываться в ячейках с c+1 по c+15 и располагаться в них следйющим образом:

c+1	0	0	0	c+6	α	0	K	c+11	α	K	K
c+2	0	0	α	c+7	K	0	α	c+12	K	α	K
c+3	0	0	K	c+8	K	α	α	c+13	K	K	α
c+4	α	0	α	c+9	α	K	α	c+14	K	K	K
c+5	K	0	K	c+10	α	α	K	c+15			

Пусть,далее, в ячейке γ+1 формируется команда приводимой подпрограммы,а в ячейку γ передаётся стандартное значение команды подпрограммы.Сама стандартная подпрограмма хранится в библиотеке стандартных программ и вводится в машину в следующем виде:сначала идёт стандартное значение комнды,а за ней идёт команда +' в таком виде,в каком она должна быть,чтобы сложить данное стандартное значение команды с соответствующим приводящим числом.Например,программа суммирования будет храниться в библиотеке стандартных программ и вводиться в машину в таком виде:

1		0	0	n
2	+'	γ	c+1	γ+1
3	+'	10	11	10
4	+'	γ	c+5	γ+1
5	−	0	0	0
6	+'	γ	c+15	γ+1
7	Пер	8	0	6
8	+'	γ	c+5	γ+1
9	+'	6	9	6
10	+'	γ	c+5	γ+1
11		0	0	0
12	+'	γ	c+1	γ+1
13	≤	6	10	5
14	+'	γ	c+5	γ+1
15	ВУ	0	0	0
16	+'	γ	c+1	γ+1
17	+	0	0	0
18	+'	γ	c+15	γ+1

3-8.47 3-391

19		0	I	0
20	+'	γ	c+I	γ+I
21	+	0	0	0
22	+'	γ	c+15	γ+I
23		0	0	0
24	+	γ	c+I	γ+I
25		условное число		

Условное число, которое ставится в конце подпрограммы будет тем признаком, который определит конец данной подпрограммы.

Специальными параметрами для приводящей программы будут:

I) Параметры стандартных подпрограмм $K_1, \alpha_1; K_2, \alpha_2 \ldots; K_m, \alpha_m$ (Ячейки, с $\beta+1$ по $\beta+2m$). Эти параметры вводятся вручную в ячейки $\beta+1$ - $\beta+2m$ в виде некоторых приводящих чисел

0 0 α_i

0 0 K_i

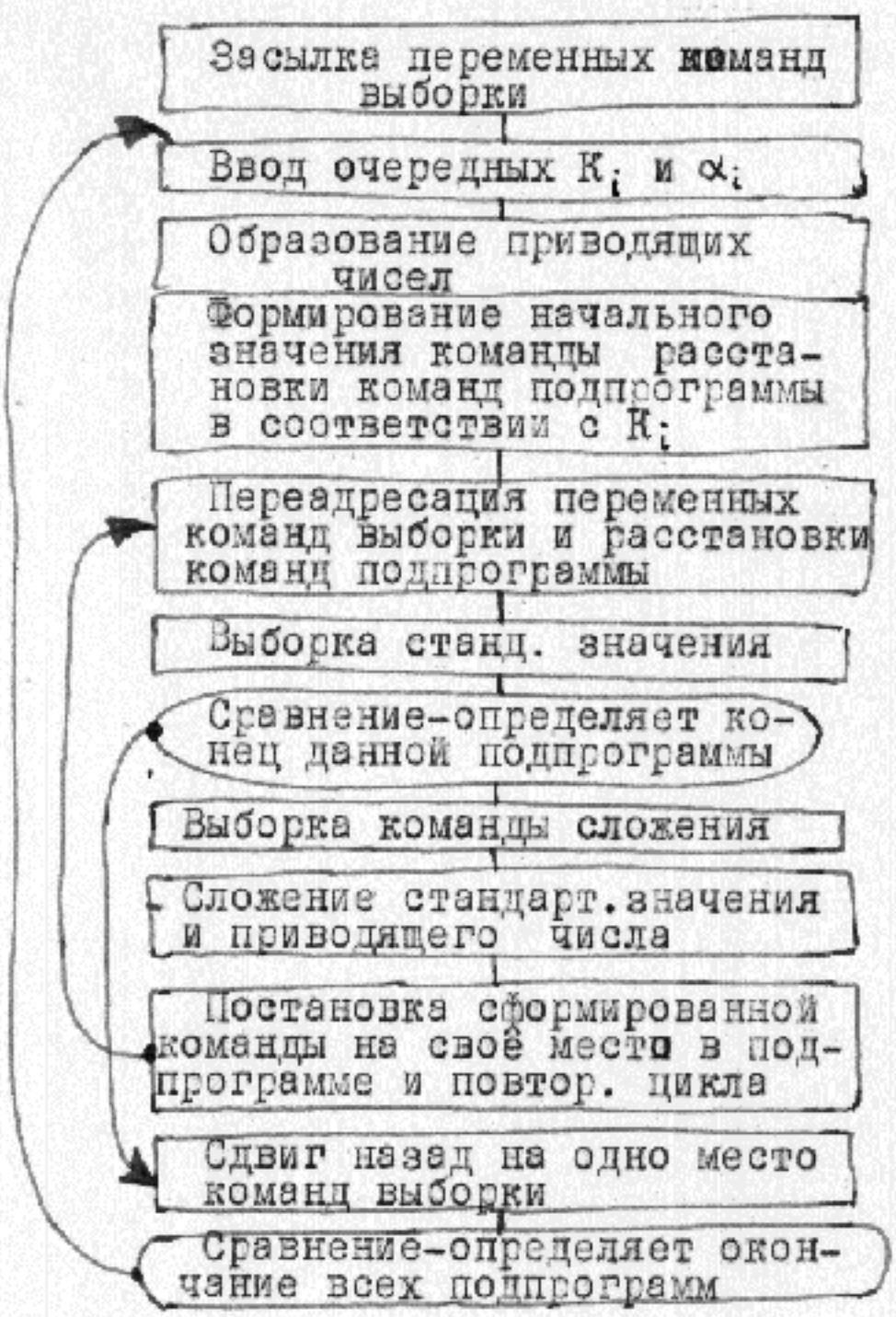
2) Число приводимых подпрограмм. Этот параметр вводится вручную на своё место в программе как константа сравнения, определяющеог, все ли подпрограммы введены. Т.к. переменной сравнения будет переменная команда выборки очередного K_i из ячеек группы , то этот параметр будет иметь вид

ПЕР $\beta+2m$ c+2

3) Начало расположения введённых подпрограмм в памяти (подпрограммы вводятся подряд и располагаются в памяти "впритык" в том же порядке, в каком расположены параметры K_i, α_i). Этот параметр можно тоже вводить в программу, но мы для простоты будем считать, что подпрограммы вводятся всегда в одно и то же место памяти.

Блок-схема программы изображена на следующей странице.

Замечания Сравнение, определяющее конец данной подпрограммы, стоит "в середине" потому, что, когда выбирается условное число, (оно выбирается командой выборки стандартного значения) оно не должно быть передано в приведённую подпрограмму, и команда сложения с приводящим числом не должна выполняться. Сдвиг назад на одно место команд выборки введён для того, чтобы сохранить поря-



Блок - схема приводящей программы

док следования стандартных значений команд и команд +' на стыке между программами(вслед за условным числом следует сразу стандартное значение 1-й команды новой программы).

Всей программы мы приводить не будем,т.к. более ясным изложение она не сделает,а составить её по приведённой блок-схеме несоставит никакого труда.Мы приведём,однако,программу образования приводящих чисел.Дело в том,что при составлении этой программы возникла любопытнаязадача - образовать приводящие числа

3-395

из чисел

0 0 α

0 0 K

3-8.49

с минимальным числом операций сдвига и сложения. Если заполнять каждый "адрес" приводящего числа независимо от других "адресов" то требуется около 40 операций. Удалось уменьшить эту программу до 21 операции, однако, нельзя быть уверенным, что это наименьшее число.

Мы считаем, что приводящие числа размещены в ячейках c+1 - c+15 так, как указывалось выше; числа K и α командами выборки переписываются сразу в c+13 и c+2. Получилась следующая программа: (ℓ - количество разрядов в одном адресе).

1	←'	c+3	2ℓ	c+7	
2	+'	c+7	c+3	c+5	K 0 K
3	Пер	c+5		c+12	
4	Пер	c+5		c+14	
5	←	c+3	ℓ	c+9	
6	Пер	c+9		c+11	
7	+'	c+2	c+9	c+13	0 K α
8	+'	c+14	c+9	c+14	K K K
9	←	c+2	2ℓ	c+6	
10	+'	c+2	c+6	c+4	α 0 α
11	+'	c+4	c+9	c+9	α K α
12	Пер	c+4		c+15	
13	+'	c+3	c+6	c+6	α 0 K
14	Пер	c+6		c+10	
15	+'	c+11	c+6	c+11	α K K
16	←	c+2	ℓ	c+8	
17	+'	c+8	c+12	c+12	K α K
18	+'	c+8	c+15	c+15	α α α
19	+'	c+8	c+10	c+10	α α K
20	+'	c+2	c+7	c+7	K 0 α
21	+'	c+7	c+8	c+8	K α α

Чтобы не загромождать изложения второстепенными деталями, мы изложили простейший вариант программы приведения. Немного усложнив её, можно было бы дополнительно сделать следующее:

1) Можно избавиться от ограничения, чтобы стандартные подпрограммы и главная программа работали обязательно на разных системах управления. Приводящую программу можно сделать и для машины с одной системой управления. В этом случае подпрограмма будет кончаться не командой возврата, а командой безусловного перехода. Адрес в этой команде можно заполнить, определив номер

3-394

3-850

команды главной программы, отсылающий к данной программе.

2) То, что в первоначальной форме записи подпрограммы за каждым начальным значением следует соответствующая команда сложения, сильно увеличивает объём библиотеки стандартных программ. Однако, т.к. нам по сути дела, чтобы сформировать эту команду, надо знать только I адрес, то можно сделать так: 3 или 4 начальных значения команд помещать подряд, а 4-ю или 5-ю ячейку, следующую за ними, заполнить сразу 3-мя или 4-мя адресами соответствующих чисел приведения. Программа приведения усложнится, но зато объём билиотеки стандартных подпрограмм может сократиться, примерно, на одну треть.

--------§§§§--------

ЛИТЕРАТУРА

[1] J. v. Neumann and H.H. Goldstine.
Numerical inverting of matrices of high order
Bull. Amer. Math. Soc. 53, N 11 (1947), 1021

см. также

УМН VI вып. 4 (1951), 123

[2] Тюринг. „Ошибки округления в матричных процессах".
УМН VI, вып 1 (1951)

[3] Фаддеева. „Вычислительные методы линейной алгебры".
М. 1951 г.

[4] Фокс, Хаски, Вилкинсон „Заметки о решении ~~алгебраических~~ систем алгебраических совместных линейных уравнений".
УМН V, вып 3 (1950)

[5] П.О. Гроссман. „К задаче численного решения систем совместных линейных алгебраических уравнений".
УМН V, вып 3 (1950), 87.

3-851 3-395

	0	1	2	3	4	5	6	7	8	9	
0											0
1											1
2											2
3							$(n+1)$	$n+1$	1 1	n^2-1 n^2-1	3
4	1 1	n n	n^2 n^2	1	0 (12)	1 (13)	2 (27)	3 (26)	4 (14)	5 (59)	4
5	6 (28)	19 (29)	16 (14)	20 (29)	24 (60)	32 (28)	33 (29)	" 0 "	" 1 "		5
6	прим. без-раф	прим. без-раф									6

БЛАНК № 2.

3-396

3-8.52

	0	1	2	3	4	5	6	7	8	9	
0											0
1											1
2											2
3							(n+1) [redacted]	n+1	1 1	n^2-1 n^2-1	3
4	1	n n	n^2 n^2	1	0 (12)	1 (13)	2 (27)	3 (26)	4 (14)	5 (59)	4
5	6 (28)	17 (29)	16 (14)	20 (29)	24 (60)	32 (28)	33 (29)	„0"	„1"		5
6	пром. рез-тать [illegible]	пром. рез-таты									6

БЛАНК № 2

I	II	III	IV	V	VI$_1$	VI$_2$	VI$_3$	VII	
0		Засылка ком. прибавления 1 к диаг. элем в ш-й строке	З	Пер	44		12		
1		Засылка ком. постановки 1 в диаг. элемент в ш-м столбце	З	Пер	45		13	3-8 53	
2		Засылка ком. очистки ячейки	З	Пер	46		27		
3		Засылка ком. передачи числа	З	Пер	47		26		
4		Засылка команды деления	З	Пер	48		14		
5		Засылка [redacted] исполнительного диаг. перемещения сравнения	З	Пер	49		59		
6		Засылка команды умножения	З	Пер	50		28		
7		Переадресация ком. прибавления 1.	П	+'	12	36	12	(n+1)	
8		Переадресация ком. постановки 1.	П	+'	13	37	13		(n+1)
9		Перемещ сравнения для пропуска главного столбца	П	+'	59	38	59	(1)	(1)
10		Восстановление команды деления.	П	−'	14	39	14	n^2-1	n^2-1
11		Сдвиг ком. умножения на 1 столбец вправо	П	+'	28	43	28	(1)	
12		прибавление 1 к диаг. элементу в ш-й строке	*		(n+1)		[redacted]		
13		Постановка 1 в диаг. элемент в ш-м столбце	*				(n+1)		
14		Деление главного столбца	*		(n)		(n)		
15		переадресация деления	П	+'	14	41	14	(n)	(n)
16		Сравнение (со команде деления) на n повторений	НС	≤	14	52	14		
17		Засылка команды вычитания.	З	Пер	51		29		
18		Восстановление команды вычитания.	П	−'	29	42	29	(n^2)	(n^2)
19		Сдвиг ком. вычитания на 1 столбец вправо	П	+'	29	38	29	(1)	(1)
20		Сравнение для конца счёта по команде вычитания	НС	≤	53	29	35		
21		переадресация ком. передачи числа.	П	+'	26	43	26	(1)	
22		переадресация ком. очистки	П	+'	27	40	27		
23		Вычитание для продолжения перемещения сравнения		−	29	59	60		
24		Сравнение для пропуска главного столбца (по ком. вычитания)	НС	≤	60	54	19		
25		Восстановление команды умножения	П	−'	28	42	28	(n^2)	(n^2)
26		передача числа-скалярного произведения из ш-й строки.	*		(1)				
27		Очистка соответствующей ячейки матрицы.	*				(1)		
28		Команда умножения	*		(n)				
29		Команда вычитания	*		(n)		(n)		
30		Переадресация умножения	П	+'	28	41	28	(n)	(n)
31		переадресация вычитания	П	+'	29	41	29	(n)	(n)
32		Сравнение (по команде умножения) на n повторений	НС	≤	28	55	28		
33		сравнение конца цикла (по команде вычитания)	НС	≤	29	56	18		
34		повторение слёта перехода к следующему этапу		ПУ,			7		
35		стоп.		стоп					
36					n+1		[redacted]		
37							n+1		

									3-398
38					1		1		
39		Программа – 36			n^2-1		n^2-1		3-8.54
		перемещение команды – 8							
40		Константа сравнения 5					1		
		Константа переадресации 8							
41		"0", "1", ячейка для переменного сравнения – 3			n		n		
42		Всего 60 ячеек			n^2	-	n^2		
		60 и 61 – промежут. результаты.							
43					1				
44		Начало матрицы – 62		+	61-n	58	61		0 (12)
45				Пер	58		61-n		1 (13)
46				Пер	57		61		2 (27)
47				Пер	61		60		3 (26)
48				:	$61+n^2$	61	$61+n^2$		4 (14)
49				-	61	61	61		5 (59)
50				×	$61+n^2$	60	61		6 (28)
51				-	$61+n^2$	61	$61+n^2$		17 (29)
52				:	$62+n^2$	61	$62+n^2$		16 (14)
53				-	$62+n^2-n$	61	$62+n^2-n$		20 (29)
54					число "0" + 2-ичн				24 (60)
55				×	$61+n^2$	60	61		32 (28)
56				×	$62+n^2$	61	$62+n^2$		33 (29)
57					число "0"				
58					число "1"				
59									для переменного сравнения

Ш – количество разрядов в цифровом коде.

БЛАНК №1

I	II	III	IV	V	VI₁	VI₂	VI₃	VII	
0		Засылка ком. прибавл. 1 к диал. числу в ш-й строке	3	Пер	44		12	3-399	
1		Засылка ком. остановки 1 в диал. чис в m-м столбце	3	Пер	45		13		
2		Засылка ком. очистки ячейки	3	Пер	46		27	3-855	
3		Засылка ком. передачи числа	3	Пер	47		26		
4		Засылка ком. деления	3	Пер	48		14		
5		Засылка нал. знач. перем. сравн.	3	Пер	49		59		
6		Засылка ком. умножения	3	Пер	50		28		
7		Переадресация ком. прибавления 1	П	+'	12	36	12	(n+1)	[redacted]
8		переадресация ком. остановки 1	П	+'	13	37	13		(n+1)
9		перемена сравнения в ком. пропуска главн. столбца	П	+'	59	38	59	(1)	(1)
10		Восстановление ком. деления	П	−'	14	39	14	(n^2-1)	(n^2-1)
11		Сдвиг ком. умножения на 1 столбец вправо	П	+'	28	43	28	(1)	
12		прибавление 1 к диал. числу в m-й строке	*		(n+1)		[redacted]		
13		остановка 1 в диал. числ в m-м столбце	*				(n+1)		
14		Деление	*		(n)		(n)		
15		Переадресация деления	П	+'	14	41	14	(n)	(n)
16		Сравнение (по команде деления) на n повторений	НС	≤	14	52	14		
17		Засылка команды вычитания	З	Пер,	51		29		
18		Восстановление ком. вычитания	П	−	29	42	29	(n^2)	(n^2)
19		Сдвиг ком вычитания на 1 столбец вправо	П	+'	29	38	29	(1)	(1)
20		Сравнение для конца счёта по команде вычитания	НС	≤	53	29	35		
21		Переадресация ком. передачи числа	П	+'	26	43	26	(1)	
22		Переадресация ком. очистки	П	+'	27	40	27		(1)
23		Вычитание для образования переменной сравнения		−	29	59	60		
24		Сравнение для пропуска главного столбца (по ком. вычитания)	НС	≤	60	54	19		
25		Восстановление команды умножения	П	−'	28	42	28	(n^2)	(n^2)
26		Передача числа - скалярного произвед. из m-й строки	*	−	(1)				
27		Очистка соответствующей ячейки матрицы	*				(1)		
28		команда умножения	*		(n)				
29		Команда вычитания	*		(n)		(n)		
30		переадресация умножения	П	+'	28	41	28	(n)	(n)
31		переадресация вычитания	П	+'	29	41	29	(n)	(n)
32		Сравнение (по команде умножения) на n повторений	НС	≤	28	55	28		
33		Сравнение конца цикла (по команде вычитания	НС	≤	29	56	18		
34		повторение счёта, переход к следующему этапу.	I	$ПУ_1$			7		
35		СТОП		стоп					
36					n+1		[redacted]		
37							n+1		
38					1		1		
39					n^2-1		n^2-1		

40								1	3-410
41		Программа - 36			n			n	
42		Перемещение команд - 8 Констант сравнения - 5			n^2			n^2	3-256
43		Констант переадресации - 8 "0", "1", ячейка для переменной сравнения - 3			1				
44		Всего - 60 ячеек. Со и 61 - промежут. результаты.		+	61-n	58		61	0 (12)
45		Начало матрицы 62		Пер	58			61-n	1 (13)
46				Пер	57			61	2 (27)
47				Пер	61			60	3 (26)
48				:	$61+n^2$	61		$61+n^2$	4 (14)
49				-	61	61		61	5 (59)
50				×	$61+n^2$	60		61	6 (28)
51				-	$61+n^2$	61		$61+n^2$	17 (29)
52				:	$62+n^2$	61		$62+n^2$	16 (14)
53				-	$62+n^2-n$	61		$62+n^2-n$	20 (29)
54		m кол. во разрядов в цифровой части.				$0+2^{-m}$			24 (60)
55				×	$61+n^2$	60		61	32 (28)
56				-	$62+n^2$	61		$62+n^2$	33 (29)
57					Число "0"				
58					Число "1"				
59									переменная сравнения

Бланк № 1