Э. И. Бурман, Г. А. Артюхин, Б. Я. Зархин

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МАТРИЧНЫХ АЛГОРИТМОВ

и метода конечных элементов в инженерных расчетах

З.И.Бурман, Г.А.Артюхин, Б.Я.Зархин

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МАТРИЧНЫХ АЛГОРИТМОВ

и метода конечных элементов в инженерных расчетах

Москва
«МАШИНОСТРОЕНИЕ»
1988

ББК 32.974
Б91
УДК 519.85

Рецензент д-р техн. наук **Г. И. Пшеничнов**

Бурман З. И. и др.
Б91 Программное обеспечение матричных алгоритмов и метода конечных элементов в инженерных расчетах/З. И. Бурман, Г. А. Артюхин, Б. Я. Зархин.— М.: Машиностроение, 1988.— 256 с.: ил.

ISBN 5-217-00011-2

Изложена методология создания программного обеспечения для численной реализации на ЭВМ матричных алгоритмов решения разнообразных задач механики и физики и, в частности, матричных алгоритмов метода конечных элементов. Описываются структура и принципы функционирования системы матричного программного обеспечения (СМПО), а также методология создания на ее основе пакетов прикладных программ. Приведены примеры использования СМПО в различных областях техники.

Для инженерно-технических работников, занятых прочностными расчетами в различных отраслях промышленности и строительстве.

Б $\frac{2405000000-010}{038(01)-88}$ 10—88 **ББК 32.974**

ISBN 5-217-00011-2

Предисловие

Повышение качества машин и сооружений при существенном сокращении сроков их проектирования в значительной мере зависит от широкого внедрения в практику проектирования современных методов расчета и эффективного использования средств вычислительной техники.

Широкие возможности вычислительной техники стимулировали разработку математических формулировок инженерных задач и методов их решения, специально ориентированных на применение ЭВМ. В частности, были созданы матричные методы решения разнообразных задач математической физики, строительной механики и т. д. Особенно важной в прикладном отношении стала разработка метода конечных элементов, впервые позволившего проводить практические расчеты весьма сложных конструкций [41]. В числе первых отечественных работ подобного характера следует назвать работы [34, 37]. С тех пор как при решении задач механики впервые стали применяться цифровые ЭВМ, понятия «матричные методы расчета», а затем «метод конечных элементов» стали неотделимы от понятия «программное обеспечение» [33]. В настоящее время матричные методы, в том числе метод конечных элементов (МКЭ), наиболее часто используют для практического решения инженерных задач. Этому способствовали как разработка теории матричных методов и МКЭ и эффективных алгоритмов, так и создание развитого специализированного программного обеспечения (пакетов прикладных программ) для проведения расчетов. Такое программное обеспечение сделало матричные методы инструментом широкого круга исследователей и проектировщиков. Созданные пакеты прикладных программ ориентированы на различные классы задач, типы ЭВМ; многие из них учитывают квалификацию пользователей.

В книге изложена методология создания программного обеспечения для численной реализации на ЭВМ матричных алгоритмов решения разнообразных задач строительной механики, механики деформируемого твердого тела, гидромеханики, в частности, алгоритмов метода конечных элементов.

Значительная часть книги посвящена описанию структуры и принципов функционирования конкретной системы матричного программного обеспечения (СМПО), ориентированной на ЭВМ серии

ЕС и выполняющей основные функции базы данных, специфические для решения инженерных задач.

Система матричного программного обеспечения и основанный на ней пакет прикладных программ для реализации МКЭ разработаны коллективом отраслевой научно-исследовательской лаборатории автоматизированных систем проектирования и расчета на прочность пространственных конструкций Казанского инженерно-строительного института под руководством д-ра техн. наук З. И. Бурмана.

Архитектура и первоначальный базовый вариант СМПО для ЕС ЭВМ были разработаны И. М. Агнистиковым [2]. Основой этой работы послужила разработанная М. Т. Тимофеевым матричная административная система для машин второго поколения [10].

При создании книги были использованы программы и другие материалы, в разработке которых принимали участие: И. М. Агнистиков, Д. М. Кордончик, А. А. Абдюшев, Г. А. Десятник, В. И. Лукашенко, О. М. Аксенов, Я. З. Бурман, Н. В. Донецкий, Д. Р. Хафизова, М. Т. Тимофеев, З. А. Бикчентаева, И. А. Шаихов, Р. А. Шакирзянов, В. А. Шувалов, Н. А. Давлеткильдеева, Р. И. Хакимов, Р. Р. Фатхуллин, Б. Г. Ерунов, С. С. Соловьев, С. М. Бастраков, Ю. А. Денисов, Ю. Г. Попов, Р. К. Сафиуллин, Г. Н. Шмелев, Н. А. Пикулев, И. М. Сулейманович, Р. Р. Салахиев.

Предисловие, введение и гл. 1, 10 (кроме разд. 10.7) написаны З. И. Бурманом, гл. 2—4, 6 и разд. 10.7 — Б. Я. Зархиным и З. И. Бурманом, гл. 5, 7—9 — Б. Я. Зархиным, гл. 11 — Г. А. Артюхиным и З. И. Бурманом.

Введение

Решение задач с помощью матричных методов и метода конечных элементов сводится в конечном итоге к выполнению определенных действий над матрицами. Постановка задачи, характер и сложность рассчитываемых объектов определяют разнообразие матричных операций, порядок и свойства матриц. Объема оперативной памяти современных ЭВМ обычно недостаточно для размещения обрабатываемых матриц; решение практических задач возможно лишь при активном использовании внешней памяти. Поэтому наряду с выбором адекватного математического алгоритма необходимо определить соответствующую структуру данных, обеспечить учет специальных свойств матриц (симметрии, разреженности, положительной определенности), быстрый доступ к необходимым массивам данных, находящимся во внешней памяти, фиксацию часто используемой информации в оперативной памяти и т. д.

Функциональная полнота матричного программного обеспечения существенно зависит от сложности и многообразия алгоритмов предметной области. Так, например, алгоритмы матричного метода сил (и МКЭ в варианте метода сил) заметно сложнее и специфичнее алгоритмов метода перемещений, которые требуют соответственно меньшей номенклатуры операций; матричный смешанный метод может приводить к неположительно определенным матрицам, что несколько усложняет решение систем уравнений и алгебраической проблемы собственных значений и т. д.

Все существующие пакеты прикладных программ (ППП), реализующие матричные методы и МКЭ, в той или иной мере используют в качестве основы программы работы с матрицами. Однако часто жесткая ориентация на тот или иной метод расчета (например, МКЭ в варианте метода перемещений) не побуждает разработчиков пакетов к созданию или использованию развитого программного обеспечения для работы с матрицами, что ограничивает их возможности и отражается на эффективности программного обеспечения.

Примерами такого подхода являются разработанные в нашей стране ППП, реализующие МКЭ перемещений для расчета конструкций, ЛИРА [26], ФРОНТ, РИПАК, СПРИНТ, зарубежные пакеты SAP-IV, SESAM, ABAQUS, MARC [1] и т. д.

Использование специализированного матричного программного обеспечения для создания пакетов прикладных программ позволяет:

эффективно реализовать разнообразные матричные алгоритмы практически без ограничения их сложности;

разрабатывать пакеты программ, дающие возможность выполнять расчеты конструкций различными методами (сил, перемещений, смешанным), что практически важно для контроля достоверности результатов;

включать в пакеты интерфейс пользователя с языковыми средствами для так называемых «мощных операций», недостаток которых ощущается в языках высокого уровня [30]; наличие таких операций дает возможность пользователям заметно быстрее писать и отлаживать свои программы в среде пакета;

выполнять важнейшие функции специализированной базы данных (символическое обращение к информации, размещенной на внешних носителях, автоматическая настройка структур данных, иерархическая структура).

Архитектура взаимосвязи специализированного матричного обеспечения с построенным на его базе пакетом, использующим матричные методы, может быть различной. Так, например, может быть применен принцип многоуровневой библиотечной организации [58], когда нулевой уровень составляет библиотека подпрограмм для матричных операций, решения систем уравнений, нахождения собственных значений и т. п., первый уровень — библиотека проблемных программ для решения задач МКЭ с использованием библиотеки нулевого уровня. Также возможно использование специализированной подсистемы для матричных расчетов, снабженной специфической системой управления данными. Примером такого подхода могут служить пакет программ для решения задач строительной механики [44], основанный на матричной интерпретирующей системе; ППП NASTRAN [1], содержащий подсистему с языком DMAP, используемую для программирования алгоритмов МКЭ, пре- и постпроцессирования, создания специальных программ пользователя. ППП COSAR [25, 43, 45], реализующий МКЭ в варианте метода перемещений, включает в себя специализированную подсистему FEDAM для управления данными и матричных вычислений; в пакете ASKA [1, 51] имеется подсистема, обеспечивающая обработку блочных матриц.

При создании ППП или специализированной подсистемы для матричных вычислений важнейшей задачей является организация работы с матрицами произвольного размера и структуры. При этом целесообразно опираться на представление больших матриц в универсальном блочном формате [10]. В качестве средства реализации матричных алгоритмов может быть использован ППП СМПО, на базе которого создан ППП СУМРАК (суперэлементный метод расчета авиационных конструкций) [2, 35], предназначенный для применения различных вариантов МКЭ (суперэлементные методы сил, перемещений и смешанный) для расчета подкрепленных оболочек.

Автоматизируя выполнение матричных операций, СМПО предоставляет пользователю СУМРАКа экономичный и мощный входной язык.

СМПО может служить для:

решения разнообразных задач, записанных в матричной форме;

создания на ее основе пакетов прикладных программ решения задач динамики и прочности машиностроительных конструкций и сооружений.

В настоящей книге показаны оба направления использования средств СМПО, в частности, описано применение СМПО для создания пакета прикладных программ, реализующего различные варианты метода конечных элементов.

Постановкам задач, решаемым с применением СМПО, как и конкретным примерам расчетов, в работе уделено внимание лишь в той мере, в какой это необходимо для иллюстрации возможностей обсуждаемых программных средств.

Большое внимание уделено в книге проблеме эффективности матричного программного обеспечения: быстродействию работы программ и эффективности использования ресурсов различных видов памяти ЭВМ. В частности, показаны возможности «распараллеливания» вычислений на однопроцессорной ЭВМ, ускорения операций обмена между оперативной и дисковой памятью и т. д. Показано, что выбор современных методов решения задач линейной алгебры и адекватных им способов реализации позволяют достичь высокой эффективности по времени вычислений и памяти.

Глава 1. Формулировка проблемы программной реализации метода конечных элементов и матричных алгоритмов в инженерных расчетах

Рассматриваемые в книге программные средства предназначены для численной реализации матричных алгоритмов, которые встречаются в различных областях науки и техники при использовании численных методов, включая широко применяемый в инженерных расчетах метод конечных элементов (МКЭ). Каждый из вариантов МКЭ (суперэлементный метод сил, суперэлементный метод перемещений, суперэлементный смешанный метод и др.) имеет свою математическую формулировку и описывается своим матричным алгоритмом. Матричные формулировки задач механики деформируемого тела, строительной механики (в том числе, строительной механики тонкостенных авиаконструкций), ряда разделов физики (таких, как электронно-колебательные спектры многоатомных молекул и др.), аэроупругости и другие в том случае, когда используются иные численные методы (например, метод конечных сумм), привносят свой перечень матричных алгоритмов.

1.1. ОБЩАЯ ХАРАКТЕРИСТИКА ВЫЧИСЛИТЕЛЬНЫХ АСПЕКТОВ РАЗЛИЧНЫХ ВАРИАНТОВ МКЭ И МАТРИЧНЫХ АЛГОРИТМОВ

Поскольку книга в основном посвящена численной реализации матричных алгоритмов и метода конечных элементов, здесь уместно остановиться на их разновидностях и особенностях. В настоящее время в механике твердого деформируемого тела, строительной механике и в машиностроении применяются: конечно-элементный метод перемещений и его обобщенный аналог — суперэлементный метод перемещений, конечно-элементный метод сил и его обобщенный аналог — суперэлементный метод сил, конечно-элементный смешанный метод, суперэлементный смешанный метод, гибридный метод.

Наиболее проста численная реализация конечно-элементного метода перемещений [25, 26]. Все другие варианты метода конечных элементов по сравнению с ним намного сложнее. Так в гибридном методе [35] необходимо более сложным образом формировать матрицу жесткости (из-за задания внутри элемента равновесного поля напряжений), а в алгоритме суперэлементного метода сил [10, 12, 35] необходимо определять усилия от единичных

лишних неизвестных, учитывать вырезы, производить упругое сочленение суперэлементов и т. п. В случае использования суперэлементного смешанного метода [35] требуется более развитое программное обеспечение, так как применяются одновременно и метод сил, и метод перемещений.

Для тех инженерных расчетов, в которых используются матричные формулировки вне метода конечных элементов, чаще всего нужно формировать матрицы специального вида, а также выполнять разнообразные операции линейной алгебры при матрицах произвольного размера и структуры.

1.2. СУПЕРЭЛЕМЕНТНЫЙ МЕТОД СИЛ

Рассмотрим кратко алгоритм суперэлементного метода сил [10] при определении напряженно-деформированного состояния, в котором применяется физическая методика выбора основной системы и лишних неизвестных усилий. В этом случае суперэлементы рассчитываемой конструкции вначале регуляризуются и используется основная статически неопределимая система, затем производится учет вырезов. Формула для определения усилий в системе имеет вид

$$\mathbf{S}=\mathbf{S}_{\mathbf{R}}+\mathbf{S}_{\mathbf{W}}=\mathbf{b}_{\mathbf{R}}\mathbf{R}+\mathbf{b}_{\mathbf{W}}\mathbf{W}, \tag{1.1}$$

где $\mathbf{S}_{\mathbf{R}}=\mathbf{b}\mathbf{R}=\mathbf{B}_0\mathbf{R}+\mathbf{b}_1\mathbf{Ч}_{\mathbf{R}}$; $\mathbf{S}$ — вектор-столбец обобщенных усилий в элементах расчетной схемы; $\mathbf{R}$ — вектор внешних обобщенных сил; $\mathbf{S}_{\mathbf{R}}$ — обобщенные усилия от $\mathbf{R}$; $\mathbf{b}_{\mathbf{R}}$ — обобщенные усилия от единичных $\mathbf{R}$; $\mathbf{W}$ — третьестепенные лишние неизвестные (л. н.) взаимодействия суперэлементов; $\mathbf{S}_{\mathbf{W}}$ — усилия от $\mathbf{W}$; $\mathbf{b}_{\mathbf{W}}$ — усилия от единичных $\mathbf{W}$; $\mathbf{Ч}_{\mathbf{R}}$ — основные л. н.; $\mathbf{b}_1$ — усилия от единичных $\mathbf{Ч}_{\mathbf{R}}$; $\mathbf{B}_0$ — усилия, статически эквивалентные единичному внешнему нагружению.

Запишем выражения для усилий от внешнего нагружения в регуляризованной системе

$$\mathbf{S}=\mathbf{B}\mathbf{R}=(\mathbf{b}_{\mathbf{R}}-\mathbf{b}_{\mathbf{W}}\mathbf{D}_{\mathbf{WW}}^{-1}\mathbf{D}_{\mathbf{WR}})\mathbf{R}, \tag{1.2}$$

где $\mathbf{B}$ — усилия при единичных $\mathbf{R}$; $\mathbf{D}_{\mathbf{WW}}$ — разрешающая матрица системы канонических уравнений (СКУ) относительно $\mathbf{W}$; $\mathbf{D}_{\mathbf{WR}}$ — грузовая матрица СКУ относительно $\mathbf{W}$.

После определения усилий в регуляризованной конструкции (расчетной схеме) производится учет вырезов путем применения методики «наложения» начальных деформаций на ранее введенные фиктивные элементы, заполнившие вырезы [10]. Усилие в отдельном j-м суперэлементе с учетом вырезов имеет вид

$$\begin{aligned}\mathbf{S}_{cj}=\{\mathbf{b}_{\mathbf{R}j}-[(\mathbf{b}_{1j}\mathbf{D}_{11j}^{-1}\mathbf{b}_{1(h)j}^{\mathrm{T}}+\mathbf{b}_{2j}\mathbf{D}_{22j}^{-1}\mathbf{b}_{2(h)j}^{\mathrm{T}}\mathbf{D}_{hcj}^{-1}\mathbf{b}_{\mathbf{R}(h)j})]\}\mathbf{R}_j+\\+\{\mathbf{b}_{\mathbf{W}j}-[(\mathbf{b}_{1j}\mathbf{D}_{11}^{-1}\mathbf{b}_{1(h)j}^{\mathrm{T}}+\mathbf{b}_{2j}\mathbf{D}_{22}^{-1}\mathbf{b}_{2(h)j}^{\mathrm{T}})(\mathbf{D}_{hc}^{-1}\mathbf{D}_{\mathbf{W}(h)j})]\}\mathbf{W}_j=\\=\{\mathbf{b}_{\mathbf{R}j}-\Delta\mathbf{b}_{\mathbf{R}cj}\}\mathbf{R}_j+\{\mathbf{b}_{\mathbf{W}j}-\Delta\mathbf{b}_{\mathbf{W}_1cj}\}\mathbf{W}_j=\mathbf{b}_{\mathbf{R}cj}\mathbf{R}_j+\mathbf{b}_{\mathbf{W}cj}\mathbf{W}_j,\end{aligned} \tag{1.3}$$

где $\mathbf{D}_{11}$ — разрешающая матрица СКУ относительно **Ч**; $\mathbf{D}_{22}$ — разрешающая матрица СКУ относительно второстепенных л. н.; $\mathbf{b}_2$ — усилия от единичных второстепенных л. н.; $\mathbf{D}_{hc}$ — разрешающая матрица СКУ для нанесения вырезов. Индекс h показывает, что строки соответствуют вырезаемым элементам.

Выражения усилий, действующих в единой системе (составленной из суперэлементов с заранее учтенными вырезами), имеют вид

$$\mathbf{S}_c = (\mathbf{b}_{Rc} - \mathbf{b}_{Wc}\mathbf{D}_{WW}^{-1}\mathbf{D}_{WRc})\mathbf{R} = \mathbf{B}_c\mathbf{R}. \qquad (1.4)$$

При расчете методом сил нужно выполнять различные по характеру матричные операции; укажем хотя бы на необходимость формировать матрицы $\mathbf{b}_{1(h)}$, $\mathbf{b}_{2(h)}$ из линейно независимых строк матриц $\mathbf{b}_1$, $\mathbf{b}_2$ для обеспечения несингулярного процесса учета выреза и т. д.

Рассмотрим кратко особенности алгоритма конечно-элементного метода сил [49, 54, 55] с автоматизированным выбором основной системы и лишних неизвестных усилий. Для этого общего случая запишем два уравнения:

уравнение равновесия

$$\mathbf{EF} = \mathbf{L}, \qquad (1.5)$$

где **E** — матрица равновесия; **F** — объединенный вектор независимых обобщенных усилий в элементах (образованный из отдельных векторов усилий элементов); **L** — вектор внешней нагрузки;

уравнение неразрывности

$$\mathbf{DF} + \mathbf{d}_0 - \mathbf{E}^{\mathrm{T}}\mathbf{r} = 0, \qquad (1.6)$$

где **D** — объединенная матрица податливости элементов; $\mathbf{d}_0$ — вектор начальных деформаций; **E** — вектор внешних перемещений конструкции.

Для статически неопределимых конструкций матрица **E** является прямоугольной, потому уравнение (1.6) не может быть решено непосредственно; это означает, что усилия **F** должны быть разделены на основные **S** и лишние неизвестные усилия **X**, причем этот процесс разделения в отличие от предыдущего варианта метода сил в данном случае производится автоматически.

Предложено несколько методик реализации метода сил. Остановимся на методе сил с использованием метода ранга II [55—57], в котором сочетается процедура «взвешивания» (необходимая для того, чтобы выбираемая автоматически основная система была более «жесткой») и метод ранга для исследования преобразованной системы уравнений равновесия.

Исходные усилия **F** могут быть выражены через нагрузку **L** и преобразованные лишние неизвестные $\mathbf{X}_p$

$$\mathbf{F} = \mathbf{W}(\mathbf{B}_{ps}\mathbf{L} + \mathbf{B}_{px}\mathbf{X}_p), \qquad (1.7)$$

где **W** — весовая матрица, являющаяся функцией коэффициентов жесткости элементов; $\mathbf{B}_{ps}$ — матрица статически эквивалентных

усилий, соответствующих преобразованной нагрузке; $\mathbf{B}_{px}$ — матрица усилий, соответствующих преобразованным л. н.;

$$\mathbf{X}_p = -[\mathbf{V}_p]^{-1}\mathbf{B}_{px}^{\mathrm{T}}((\mathbf{W}^{\mathrm{T}}\mathbf{D}\mathbf{W})[\mathbf{B}_{ps}\mathbf{L}] + \mathbf{W}^{\mathrm{T}}\mathbf{d}), \tag{1.8}$$

а $\mathbf{V}_p = \mathbf{B}_{px}^{\mathrm{T}}(\mathbf{W}^{\mathrm{T}}\mathbf{D}\mathbf{W})\mathbf{B}_{px}$ — преобразованная матрица податливости статически неопределимой системы.

В рассматриваемом случае $\mathbf{W} = \mathbf{K}_L$ — нижняя треугольная матрица матрицы жесткости системы $\mathbf{K}$ (полученная применением процедуры Холецкого). Исходные усилия $\mathbf{F}$ вычисляют в этом случае по формуле

$$\mathbf{F} = \mathbf{W}(\mathbf{I} - \mathbf{B}_{px}\mathbf{V}_p^{-1}\mathbf{B}_{px}^{\mathrm{T}})\mathbf{B}_{px}\mathbf{L} - \mathbf{W}\mathbf{B}\mathbf{V}_p^{-1}\mathbf{B}_{px}\mathbf{W}^{\mathrm{T}}\mathbf{d}_0. \tag{1.9}$$

Процесс разделения усилий производится автоматически в процессе обращения прямоугольной матрицы $[\mathbf{E} \vdots \mathbf{L}]$. Число возможных псевдообратных матриц $[\mathbf{E} \vdots \mathbf{L}]^{-1}$ равно числу возможных основных систем. Обращение прямоугольной матрицы производится по алгоритму Жордана — Гаусса. Выбор базиса (основной системы) происходит при выборе главных элементов по алгоритму ранг III со «взвешиванием». В качестве критерия хорошей обусловленности основной системы принимается ее наименьшая деформативность.

Рассмотрим также алгоритм конечно-элементного метода сил, который назван процедурой «*LU* с обратным ходом» [49]. Благодаря применению этой процедуры к уравнению равновесия (1.5) удается получить ленточную матрицу податливости $\mathbf{B}^{\mathrm{T}}{}_x\mathbf{f}\mathbf{B}_x$, что улучшает и ускоряет процесс решения системы разрешающих уравнений. В отличие от вышеизложенного алгоритма в этом случае вместо схемы Жордана — Гаусса используется процедура исключения Гаусса (или процедура *LU*-разложения); затем применяется некоторая схема обратного хода, основанная на *LU*-процедуре, резко повышающей разреженность матрицы $\mathbf{B}_x$. Суть этой процедуры состоит в следующем. Предположим, что мы имеем факторизацию *LU*-матрицы равновесия системы. Обозначим через $\mathbf{B}_x^{LU}$ «самосопряженную» матрицу, которая образуется в ходе процедуры *LU*.

Применяя *LU*-процедуру к матрице равновесия со столбцами, расположенными в обратном порядке, получим матрицу $\mathbf{B}_x^{\mathrm{T}B}$, которая будет ленточной и даст тот же окончательный результат, что и матрица $\mathbf{B}_x^{LU}$, так как при этом опускают часть уравнений с линейно зависимыми строками.

В работе [10] описан конечно-элементный метод сил расчета на собственные колебания подкрепленных оболочек.

В работе [6] описан суперэлементный метод сил расчета на собственные колебания, т. е. решена задача динамического синтеза. Методика расчета на собственные и вынужденные колебания может быть рассмотрена как частный случай суперэлементного смешанного метода, описанного в разд. 1.4.

1.3. СУПЕРЭЛЕМЕНТНЫЙ ВАРИАНТ КОНЕЧНО-ЭЛЕМЕНТНОГО МЕТОДА ПЕРЕМЕЩЕНИЙ

Определение напряженно-деформированного состояния

При применении этого метода конструкцию разбивают на подконструкции (суперэлементы), границы которых могут быть в общем случае произвольными [54]. Напряженно-деформированное состояние каждой подконструкции анализируется отдельно в предположении, что она закреплена по границам со смежными подконструкциями. Затем из условия равновесия определяют действительные перемещения граничных узлов при их одновременном освобождении.

Рассмотрим условия равновесия r-й подконструкции

$$\begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ij} \\ \mathbf{K}_{ji} & \mathbf{K}_{jj} \end{bmatrix}_r \begin{bmatrix} \mathbf{U}_i \\ \mathbf{U}_j \end{bmatrix}_r = \begin{bmatrix} \mathbf{P}_i \\ \mathbf{P}_j \end{bmatrix}_r, \tag{1.10}$$

где индексы «i» и «j» относятся к внутренним и граничным перемещениям соответственно.

При условии, что границы подконструкции закреплены, т. е. $\mathbf{U}_j^r = 0$, из уравнения (1.10) имеем первое слагаемое уравнения

$$\mathbf{U}_i^r = (\mathbf{K}_{ji}^r)^{-1} \mathbf{P}_i^r \tag{1.11}$$

и

$$\mathbf{D}_j^r = \mathbf{K}_{ji}^r (\mathbf{K}_{ii}^r)^{-1} \mathbf{P}_i^r. \tag{1.12}$$

Матрица $\mathbf{K}_{ii}^r$ является невырожденной, так как границы r-й подконструкции закреплены. Освобождая границы подконструкции при $\mathbf{P}_i^r = 0$, из уравнения (1.10) имеем

$$\mathbf{U}_i^r = -\mathbf{K}_{ii}^{-1} \mathbf{K}_{ij} \mathbf{U}_j; \tag{1.13}$$

$$\mathbf{U}_j = \mathbf{K}_b^{-1} \mathbf{P}_b, \tag{1.14}$$

где

$$\mathbf{K}_b = \mathbf{K}_{jj} - \mathbf{K}_{ji} \mathbf{K}_{ii}^{-1} \mathbf{K}_{ij} \tag{1.15}$$

является матрицей жесткости границы.

Выделяя граничные узлы с учетом уравнения (1.13), получим для внешней нагрузки

$$\mathbf{P}_b^r = \mathbf{P}_j^r - \mathbf{K}_{ji}^r (\mathbf{K}_{ii}^r)^{-1} \mathbf{P}_i^r. \tag{1.16}$$

Уравнение равновесия полной конструкции можно записать через граничные перемещения и силы в виде

$$\mathbf{K}_b \mathbf{U}_b = \mathbf{P}_b, \tag{1.17}$$

где $\mathbf{K}_b$ и $\mathbf{P}_b$ получаются суммированием соответствующих матриц подконструкций так же, как это делается в прямом методе жесткостей при формировании матриц жесткости подконструкций.

Систему уравнений (1.17) можно решить относительно перемещений граничных узлов, если исключены перемещения конструкции как твердого тела и матрица $\mathbf{K}_b$ будет невырожденной.

Зная перемещения граничных узлов $\mathbf{U}_b$ из уравнения (1.10), можно определить перемещения внутренних узлов подконструкции от совместного действия сил $\mathbf{P}_i^r$ и граничных перемещений $\mathbf{U}_b^r$

$$\mathbf{U}_i^r = (\mathbf{K}_{ii}^r)^{-1}\mathbf{P}_i^r - (\mathbf{K}_{ii}^r)^{-1}\mathbf{K}_{ij}^r\mathbf{U}_b^r. \qquad (1.18)$$

Для определения перемещений граничных узлов $\mathbf{U}_b$ требуется решать систему уравнений (1.17), число неизвестных в которой много меньше, чем в уравнениях, требующихся для анализа полной конструкции. Таким образом, одну систему уравнений высокого порядка заменяют несколькими системами, порядок которых значительно ниже (см. уравнения (1.11) и (1.17)).

Расчет на собственные колебания и устойчивость

Пусть подконструкция имеет n^r степеней свободы. Выражения для кинетической и потенциальной энергии подконструкции можно записать в виде

$$\text{КЭ} = \frac{1}{2}\dot{\mathbf{U}}^{r\,\mathrm{T}}\mathbf{M}^r\dot{\mathbf{U}}^r; \qquad (1.19)$$

$$\text{ПЭ} = \frac{1}{2}\mathbf{U}^{r\,\mathrm{T}}\mathbf{K}^r\mathbf{U}^r, \qquad (1.20)$$

где $\mathbf{U}^r$ — вектор перемещений r-й подконструкции; $\mathbf{M}^r$ и $\mathbf{K}^r$ — соответственно матрицы массы и жесткости порядка $n^r \times n^r$.

Матрица $\mathbf{M}^r$ является действительной, симметричной и положительно определенной, матрица $\mathbf{K}^r$ — действительная, симметричная, положительно определенная или положительно полуопределенная.

Введем в рассмотрение сокращенную систему обобщенных координат ρ^r порядка $N^r (N^r \ll n^r)$ и запишем перемещения r-й подконструкции

$$\mathbf{U}^r = \mathbf{\Phi}^r\rho^r, \qquad (1.21)$$

где $\mathbf{\Phi}^r$ — матрица размером $n^r \times N^r$, столбцы которой являются допустимыми векторами. Допустимые векторы должны удовлетворять геометрическим граничным условиям подконструкции. Они должны быть линейно независимыми, а их линейные комбинации должны достаточно точно представлять низшие собственные векторы подконструкции.

В методе Рэлея — Ритца предполагается, что формы колебаний являются линейной комбинацией форм деформации.

После подстановки уравнения (1.21) в (1.19) и (1.20) получим выражения для энергии колеблющейся системы в виде

$$\text{ПЭ} = \frac{1}{2}\rho^{r\,\mathrm{T}}\mathbf{k}^r\rho^r; \qquad (1.22)$$

$$\text{КЭ} = \frac{1}{2}\dot{\rho}^{r\,\mathrm{T}}\mathbf{m}^r\dot{\rho}^r, \qquad (1.23)$$

где матрицы $\mathbf{m}^r = \mathbf{\Phi}^r \mathbf{M}^r \mathbf{\Phi}^r$ и $\mathbf{k}^r = \mathbf{\Phi}^{rT} \mathbf{k}^r \mathbf{\Phi}^r$ имеют размеры $N^r \times N^r$ и являются действительными и симметричными.

Рассмотрим конструкцию, состоящую из подконструкций. Выражения для кинетической и потенциальной энергии системы из независимо функционирующих подконструкций имеют вид

$$\text{КЭ} = \sum_{s=1}^{m} \text{КЭ}_s = \frac{1}{2} \dot{\rho}^T \mathbf{M} \dot{\rho}; \tag{1.24}$$

$$\text{ПЭ} = \sum_{s=1}^{m} \text{ПЭ}_s = \frac{1}{2} \rho^T \mathbf{K} \rho, \tag{1.25}$$

где $\rho = \{\rho_1, \rho_2, ..., \rho_m\}$ — вектор обобщенных перемещений расчлененной системы размерностью N; $\mathbf{M}$ и $\mathbf{K}$ — матрицы массы и жесткости нерасчлененной конструкции размером $N \times N$

$$\mathbf{M} = \lceil \mathbf{m}_1 \mathbf{m}_2 ... \mathbf{m}_m \rceil; \tag{1.26}$$

$$\mathbf{K} = \lceil \mathbf{k}_1 \mathbf{k}_2 ... \mathbf{k}_m \rceil. \tag{1.27}$$

Рассматривая свободные колебания для расчлененной конструкции, придем к отношению Рэлея

$$R = \frac{\rho^T \mathbf{K} \rho}{\rho^T \mathbf{M} \rho}. \tag{1.28}$$

В действительности подконструкции функционируют совместно, поэтому необходимо выполнение условия геометрической совместности, означающее, что величины перемещений и углов поворота точек, лежащих на общих границах подконструкций, должны быть одинаковыми. Так, для подконструкций a и b, имеющих общую границу j:

$$\mathbf{U}_{ja} = \mathbf{U}_{jb}. \tag{1.29}$$

Условие (1.29) можно рассматривать как уравнение связей, которые необходимо наложить на вектор обобщенных перемещений ρ расчлененной конструкции. Если мы имеем число уравнений связи, равное M_c, то вся конструкция имеет $n = N - M_c$ степеней свободы.

Перейдем к рассмотрению задачи синтеза подконструкций. Пусть $\mathbf{Q}$ является n-мерным вектором обобщенных форм колебаний всей конструкции, тогда

$$\rho = \mathbf{TQ}, \tag{1.30}$$

где $\mathbf{T}$ — матрица связи размером $N \times n$.

Подставим вектор обобщенных перемещений расчлененной системы ρ, выраженный уравнением (1.30), в (1.28) и получим отношение Рэлея для синтезированной конструкции

$$R_G = \frac{\mathbf{Q}^T \mathbf{K}_g \mathbf{Q}}{\mathbf{Q}^T \mathbf{M}_g \mathbf{Q}}, \tag{1.31}$$

где $\mathbf{K}_g = \mathbf{T}^{\mathrm{T}}\mathbf{K}\mathbf{T}$ и $\mathbf{M}_g = \mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{T}$ — действительные симметричные матрицы жесткости и массы синтезированной конструкции размером n. Из условия стационарности отношения Рэлея можно приближенно найти нормальные формы колебаний полной конструкции. Это соответствует решению собственной проблемы

$$\mathbf{K}_g\mathbf{Q} = \mathbf{M}_g\mathbf{Q}\Omega^2, \tag{1.32}$$

где Ω^2 определяет частоты собственных колебаний конструкции. Решение уравнения (1.32) дает обобщенные формы колебаний $\mathbf{Q}$, а действительные формы колебаний можно определить из уравнения (1.30). Формы колебаний всей конструкции синтезируются из форм колебаний отдельных подконструкций. Поэтому важно, чтобы последние возможно точнее аппроксимировали колебания полной конструкции. Каждая подконструкция, смежная с данной, оказывает на нее инерционное и упругое воздействия, которые передаются через границы подконструкции. Следовательно, в матрицах масс и жесткости подконструкций должны быть сделаны соответствующие уточнения.

Рассмотрим конструкцию, состоящую из двух подконструкций a и b. Потенциальная и кинетическая энергии полной конструкции равны

$$\text{ПЭ} = \frac{1}{2}\mathbf{U}_a^{\mathrm{T}}\mathbf{k}_a\mathbf{U}_a + \frac{1}{2}\mathbf{U}_b^{\mathrm{T}}\mathbf{k}_b\mathbf{U}_b; \tag{1.33}$$

$$\text{КЭ} = \frac{1}{2}\dot{\mathbf{U}}_a^{\mathrm{T}}\mathbf{m}_a\dot{\mathbf{U}}_a + \frac{1}{2}\dot{\mathbf{U}}_b^{\mathrm{T}}\mathbf{m}_b\dot{\mathbf{U}}_b. \tag{1.34}$$

Уравнение равновесия подконструкции b при установившихся колебаниях имеет вид

$$\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ij} \\ \mathbf{k}_{ji} & \mathbf{k}_{jj} \end{bmatrix}_b \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix}_b + \begin{bmatrix} \mathbf{m}_{ii} & \mathbf{m}_{ij} \\ \mathbf{m}_{ji} & \mathbf{m}_{jj} \end{bmatrix}_b \begin{bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_j \end{bmatrix}_b = \begin{bmatrix} \mathbf{F}_i \\ \mathbf{F}_j \end{bmatrix}_b, \tag{1.35}$$

где $\mathbf{u}$ — перемещения точек конструкции.

Индекс j, как и раньше, определяет члены, относящиеся к границе подконструкции, а индекс i — все остальные члены. В уравнении (1.35) следует положить $\mathbf{F}_i = 0$, так как перемещения точек подконструкции могут вызываться только силами, передаваемыми через границу, и никакие внешние нагрузки на конструкцию не действуют. Пренебрегая инерционными силами, запишем выражение (1.35) в виде

$$\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ij} \\ \mathbf{k}_{ji} & \mathbf{k}_{jj} \end{bmatrix}_b \begin{bmatrix} \mathbf{u}_j \\ \mathbf{u}_j \end{bmatrix}_b = \begin{bmatrix} 0 \\ \mathbf{F}_j \end{bmatrix}_b. \tag{1.36}$$

Из уравнения (1.36) имеем

$$\begin{aligned} \mathbf{u}_{jb} &= (-\mathbf{k}_{ii}^{-1}\mathbf{k}_{ij}\mathbf{u}_j)_b; \\ \mathbf{u}_b &= \begin{bmatrix} \mathbf{k}_{ii}^{-1}\mathbf{k}_{ij} \\ \mathbf{E} \end{bmatrix}_b \mathbf{u}_{jb}, \end{aligned} \tag{1.37}$$

где через $\mathbf{E}$ обозначена единичная матрица.

Подставим выражение (1.37) в уравнения (1.33) и (1.34). Выражения для потенциальной и кинетической энергии системы запишутся в виде

$$\text{ПЭ}=\frac{1}{2}\mathbf{u}_a^{\text{т}}\mathbf{k}_a\mathbf{u}_a+\frac{1}{2}\mathbf{u}_{jb}^{\text{т}}\mathbf{k}_b^*\mathbf{u}_{jb}, \tag{1.38}$$

$$\text{КЭ}=\frac{1}{2}\dot{\mathbf{u}}_a^{\text{т}}\mathbf{m}_a\dot{\mathbf{u}}_a+\frac{1}{2}\dot{\mathbf{u}}_{jb}^{\text{т}}\mathbf{m}_b^*\dot{\mathbf{u}}_{jb}, \tag{1.39}$$

где

$$\mathbf{k}_b^*=(\mathbf{k}_{jj}-\mathbf{k}_{ji}\mathbf{k}_{ii}^{-1}\mathbf{k}_{ij})_b; \tag{1.40}$$

$$\mathbf{m}_b^*=(\mathbf{m}_{jj}-\mathbf{m}_{ji}\mathbf{k}_{ii}^{-1}\mathbf{k}_{ij}-\mathbf{k}_{ji}\mathbf{k}_{ii}^{-1}\mathbf{m}_{ij}+\mathbf{k}_{ji}\mathbf{k}_{ii}^{-1}\mathbf{m}_{ii}\mathbf{k}_{ii}^{-1}\mathbf{k}_{ij})_b. \tag{1.41}$$

Из условия геометрической совместности подконструкций имеем $\mathbf{u}_{jb}=\mathbf{u}_{ja}$ или

$$\mathbf{u}_{jb}=L\mathbf{u}_a=[0\ \mathbf{E}]\begin{bmatrix}\mathbf{u}_i\\ \mathbf{u}_j\end{bmatrix}_a, \tag{1.42}$$

где $L=[0\mathbf{E}]$. С учетом уравнения (1.42) выражения для энергии полной конструкции примут вид

$$\text{ПЭ}=\frac{1}{2}\mathbf{u}_a^{\text{т}}\bar{\mathbf{k}}_a\mathbf{u}_a; \tag{1.43}$$

$$\text{КЭ}=\frac{1}{2}\dot{\mathbf{u}}_a^{\text{т}}\bar{\mathbf{m}}_a\dot{\mathbf{u}}_a, \tag{1.44}$$

где

$$\bar{\mathbf{k}}_a=\mathbf{k}_a+\mathbf{L}^{\text{т}}\mathbf{k}_b^*\mathbf{L}; \tag{1.45}$$

$$\bar{\mathbf{m}}_a=\mathbf{m}_a+\mathbf{L}^{\text{т}}\mathbf{m}_b^*\mathbf{L}. \tag{1.46}$$

Уточнения (модификация) матриц жесткости и массы подконструкции (1.45) и (1.46) соответствуют граничным степеням свободы подконструкции a. Эти дополнительные члены определяются выражениями (1.40) и (1.41) и соответствуют приведенным матрицам жесткости и масс, описанным в работе [15].

Уравнение свободных колебаний модифицированной подконструкции без учета демпфирования системы имеет вид

$$\bar{\mathbf{k}}\mathbf{u}=\omega^2\bar{\mathbf{m}}\mathbf{u}. \tag{1.47}$$

В случае действительных симметричных и положительно определенных матриц жесткости и массы (это соответствует закрепленной конструкции) все N корней уравнения (1.47) являются положительными и действительными.

Для определения ряда низших форм колебаний подконструкции из уравнения вида

$$\mathbf{KU}=\mathbf{MU}\Lambda \tag{1.48}$$

можно применить различные методы [5], однако наиболее распространенным является метод итерации подпространства, позволяющий эффективно отыскивать низшие частоты и соответствующие

им формы колебаний. Этот итерационный метод можно рассматривать как развитие улучшенной методики Рэлея — Ритца.

Пусть необходимо определить p собственных пар. Начинаем итерации с $q=\min\{2p, p+8\}$ линейно независимых пробных векторов $\mathbf{\Phi}^{(0)}$ по методу Ритца. Безразмерные улучшенные формы колебаний получаются в результате решения уравнений

$$\mathbf{D}^{(0)}=\mathbf{M}\mathbf{\Phi}^{(0)}; \tag{1.49}$$

$$\mathbf{K}\bar{\mathbf{\Phi}}^{(1)}=\mathbf{D}^{(0)}. \tag{1.50}$$

Применив метод Ритца при решении задачи о собственных значениях, получим матрицы жесткости и массы в обобщенных координатах

$$\mathbf{k}^{(1)}=\bar{\mathbf{\Phi}}^{(1)\mathrm{T}}\mathbf{K}\bar{\mathbf{\Phi}}^{(1)}=\bar{\mathbf{\Phi}}^{(1)\mathrm{T}}\mathbf{P}^{(0)}; \tag{1.51}$$

$$\mathbf{m}^{(1)}=\bar{\mathbf{\Phi}}^{(1)\mathrm{T}}\mathbf{M}\bar{\mathbf{\Phi}}^{(1)}=\bar{\mathbf{\Phi}}^{(1)\mathrm{T}}\mathbf{P}^{(1)}. \tag{1.52}$$

Индекс «(1)» относится к первому циклу итерационного процесса. Обобщенная задача о собственных значениях примет вид

$$\mathbf{k}^{(1)}\boldsymbol{\rho}^{(1)}=\mathbf{m}^{(1)}\boldsymbol{\rho}^{(1)}\Omega^{2(1)}. \tag{1.53}$$

Для решения уравнения (1.53) используется QL-алгоритм, позволяющий быстро и эффективно решать полную алгебраическую проблему собственных значений невысокого порядка для матриц, которые помещаются в оперативной памяти. В результате решения уравнения (1.53) получаем модальные векторы $\boldsymbol{\rho}^{(1)}$, нормированные относительно матрицы $\mathbf{m}^{(1)}$, так что

$$\rho^{1(\mathrm{T})}m^{(1)}\rho^{(1)}=E. \tag{1.54}$$

Улучшенные пробные векторы можно получить по формуле

$$\mathbf{\Phi}^{(1)}=\bar{\mathbf{\Phi}}^{(1)}\boldsymbol{\rho}^{(1)}. \tag{1.55}$$

Далее итерационный процесс можно продолжить, заменяя в выражении $\mathbf{P}^{(0)}$ на $\mathbf{P}^{(1)}$:

$$\mathbf{P}^{(1)}=\mathbf{M}\mathbf{\Phi}^{(1)}=\mathbf{M}\bar{\mathbf{\Phi}}^{(1)}\boldsymbol{\rho}^{(1)}=\bar{\mathbf{P}}^{(1)}\boldsymbol{\rho}^{(1)}. \tag{1.56}$$

В результате вычисления определяют истинные формы колебаний и их частоты

$$\mathbf{\Phi}^{(i)}\to\mathbf{U},\ \Omega^{2(i)}\to\Lambda \text{ при } i\to\infty.$$

Перейдем к синтезу форм колебаний полной конструкции.

Рассмотрим, не теряя общности, конструкцию, разделенную на две подконструкции a и b. Полный вектор действительных перемещений можно представить в виде

$$\mathbf{U}=\{\mathbf{u}_{ia}\mathbf{u}_{ja}\mathbf{u}_{ib}\mathbf{u}_{jb}\}. \tag{1.57}$$

Матрицы жесткости и массы для всей конструкции имеют вид

$$\mathbf{K}=\begin{bmatrix}\mathbf{K}_a & \\ & \mathbf{K}_b\end{bmatrix},\quad \mathbf{M}=\begin{bmatrix}\mathbf{M}_a & \\ & \mathbf{M}_b\end{bmatrix}. \tag{1.58}$$

Из условия совместности действительных перемещений на общих границах следует выражение

$$\mathbf{U}=\{\mathbf{u}_{ia}\mathbf{u}_{ja}\mathbf{u}_{ib}\mathbf{u}_{jb}\}=L\{\mathbf{u}_{ia}\mathbf{u}_{ja}\mathbf{u}_{ib}\}=\mathbf{L}\mathbf{U}_s, \tag{1.59}$$

где $\mathbf{U}_s=\{\mathbf{u}_{ia}\mathbf{u}_{ja}\mathbf{u}_{ib}\}$; $\mathbf{L}$ — булева матрица.

Для каждой подконструкции определяется ряд низших форм колебаний. Предполагая, что действительные перемещения являются линейной комбинацией этих форм, запишем

$$\mathbf{U}_s=\mathbf{F}\mathbf{Q}, \tag{1.60}$$

где $\mathbf{Q}$ — вектор обобщенных перемещений полной конструкции;

$$\mathbf{F}=\begin{bmatrix}\mathbf{\Phi}_{ia} & & \\ & \mathbf{\Phi}_{ja} & \\ & & \mathbf{\Phi}_{ib}\end{bmatrix}. \tag{1.61}$$

Число обобщенных перемещений значительно меньше числа действительных перемещений, так как для определения форм колебаний полной конструкции используются лишь те формы колебаний подконструкций, которые соответствуют низшим частотам. При исследовании все подконструкции подразделяются на главные и второстепенные. Для второстепенных подконструкций определяют естественные формы колебаний при закрепленных границах, для главных подконструкций — при свободных границах, но матрицы жесткости и массы этих подконструкций модифицируются с помощью уравнений (1.45) и (1.46). Для главной подконструкции a определяют формы колебаний при свободных границах, т. е. $\mathbf{\Phi}_{ia}$ и $\mathbf{\Phi}_{ja}$, а для подконструкции b необходимо определить лишь формы $\mathbf{\Phi}_{ib}$ при закрепленной границе.

Подставим в уравнение (1.59) выражение (1.60) для $\mathbf{U}_s$

$$\mathbf{U}=\mathbf{L}\mathbf{F}\mathbf{Q}=\mathbf{T}\mathbf{Q}, \tag{1.62}$$

где $\mathbf{T}=\mathbf{L}\mathbf{F}$.

С учетом уравнения (1.62) уравнение движения полной конструкции при свободных колебаниях примет вид уравнения (1.48) т. е.

$$\mathbf{K}_g\mathbf{Q}=\mathbf{M}_g\mathbf{Q}\Omega^2, \tag{1.63}$$

где Ω^2 определяет частоты собственных колебаний конструкции; $\mathbf{K}_g$ и $\mathbf{M}_g$ — обобщенные матрицы масс и жесткости

$$\mathbf{K}_g=\mathbf{T}^{\mathrm{T}}\mathbf{K}\mathbf{T}; \tag{1.64}$$

$$\mathbf{M}_g=\mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{T}. \tag{1.65}$$

Уравнение (1.63) может быть решено для обобщенных форм колебаний $\mathbf{Q}$, а действительные формы колебаний определяются из уравнения (1.62).

Описанный метод покомпонентного синтеза дает приближенные значения частот и форм колебаний, так как основывается на использовании только ряда низших форм подконструкций.

Другим методом решения проблемы собственных значений (1.48) является соединение метода итерации подпространства с методом подконструкций. При использовании метода итерации подпространства на каждом шаге итерационного процесса необходимо решать систему уравнений (1.50). Решение может быть получено с использованием только матриц жесткости и массы подконструкций [40]. Рассмотрим сначала вычисление $\mathbf{P}^{(0)}$. Пусть l — число подконструкций, $\mathbf{M}^r$ — матрица масс для r-й подконструкции; $\mathbf{L}^r$ — булева матрица, позволяющая включить матрицу $\mathbf{M}^r$ в полную матрицу масс $\mathbf{M}$. Тогда можно записать

$$\mathbf{P}^{(0)}=\left(\sum_{r=1}^{l}\mathbf{L}^{r\mathrm{T}}\mathbf{M}^r\mathbf{L}^r\right)\mathbf{\Phi}^{(0)}=\sum_{r=1}^{l}\{(\mathbf{L}^{r\mathrm{T}}\mathbf{M}^r\mathbf{L}^r)\mathbf{\Phi}^{(0)}\}, \qquad (1.66)$$

т. е. матрица $\mathbf{P}^{(0)}$ вычисляется с использованием только матриц масс подконструкций.

Для вычисления $\overline{\mathbf{\Phi}}^{(1)}$ из уравнения (1.50) достаточно иметь только матрицы жесткости подконструкций. Уравнение (1.50) можно рассматривать как уравнение равновесия конструкции под действием псевдостатической нагрузки $\mathbf{P}^{(0)}$, при решении которого можно воспользоваться методом подконструкций.

Обычно при проектировании решаются как задача статики, так и задача о собственных колебаниях. Поэтому при решении уравнения (1.50) используются матрицы $\mathbf{K}_l$, $(\mathbf{K}_{ii}^{-1}\mathbf{K}_{ij})$ и другие, которые уже получены при решении задачи статики.

Метод является альтернативным методу покомпонентного синтеза и позволяет определить точно ρ низших форм колебаний и частот с использованием только матриц жесткости и масс подконструкций.

Рассмотрим алгоритм расчета на устойчивость.

Метод перемещений можно распространить на задачи анализа устойчивости [5, 54] введением геометрических жесткостей, которые добавляются к упругим жесткостям для формирования полной матрицы жесткости. Геометрические матрицы жесткости линейно зависят от начального распределения напряжений, поэтому уравнение устойчивости записывается в виде

$$\mathbf{K}_G\mathbf{U}\mathbf{\Lambda}=\mathbf{K}\mathbf{U}, \qquad (1.67)$$

где $\mathbf{K}_G$ — геометрическая матрица жесткости, а $\mathbf{\Lambda}$ определяют критическую конфигурацию нагрузки, при которой происходит потеря устойчивости конструкции.

При анализе реальных сооружений матрицы $\mathbf{K}$ и $\mathbf{K}_G$ имеют большие размеры, что приводит к необходимости решения собственной проблемы высокого порядка (порядка n). Для того чтобы использовать преимущества решения собственной проблемы малых размеров при описании конструкции с большим числом степеней свободы, применяют сокращенную систему обобщенных координат аналогично уравнению (1.21). Векторы $\mathbf{\Phi}^r$ должны давать сравни-

тельно полное описание формы потери устойчивости и могут быть найдены из решения уравнения

$$\mathbf{k}\mathbf{\Phi}^r = \mathbf{H}, \tag{1.68}$$

где столбцы вектора **H** представляют собой систему независимых векторов статической нагрузки.

Векторы **H** выбирают так, чтобы решение уравнения **Φ** аппроксимировало основные формы потери устойчивости. Решение уравнения (1.68) может быть найдено параллельно с определением напряженного состояния под действием внешней нагрузки.

Подставляя уравнение (1.21) в уравнение (1.67) и умножая это выражение слева на $\mathbf{\Phi}^{\mathrm{T}}$, получим проблему собственных значений сокращенных размеров

$$\mathbf{K}_g\rho = \mathbf{K}_{G_g}\rho\Omega^2, \tag{1.69}$$

где $\mathbf{K}_{G_g} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{K}_G\mathbf{\Phi}$ и $\mathbf{K}_g = \mathbf{\Phi}^{\mathrm{T}}\mathbf{K}\mathbf{\Phi}$ — соответственно обобщенные матрицы жесткости и геометрической жесткости.

Из уравнения (1.69) определяем приближенные формы потери устойчивости

$$\mathbf{U} = \mathbf{\Phi}\rho. \tag{1.70}$$

Полученные из решения уравнения (1.70) формы **U** можно использовать в качестве первого приближения в методе одновременных итераций.

Изложенную выше методику, основанную на использовании форм колебаний отдельных подконструкций для анализа форм колебаний полной конструкции, можно обобщить на случай анализа устойчивости, заменив во всех выражениях матрицу масс **M** на геометрическую матрицу жесткости $\mathbf{K}_G$.

1.4. СМЕШАННЫЙ СУПЕРЭЛЕМЕНТНЫЙ МЕТОД

Определение напряженно-деформированного состояния

В некоторых случаях является целесообразным построение схемы МКЭ, основанной на так называемой комбинированной или смешанной вариационной постановке задачи, при которой в отдельных частях тела определены разные функционалы [32]. Такая постановка задачи позволяет найти решение наиболее экономичным методом [10, 35].

Следуя [64], предположим для определенности, что область V, занятая телом, разбита поверхностью Ω на две части V_1 и V_2. Область V_1 ограничена поверхностями S_1 и Ω, область V_2 — поверхностями S_2 и Ω. В области V_1 определен функционал Лагранжа, неизвестными являются перемещения, в области V_2 определен функционал Кастильяно, неизвестными являются напряжения. На границе Ω должны выполняться условия сопряжения — равенство перемещений и напряжений со стороны V_1 и V_2. Согласно [32] при-

ходим к задаче на одновременный экстремум двух функционалов

$$\min \Pi_1(\mathbf{u}^*, \bar{\sigma});$$
$$\max \Pi_2(\bar{\sigma}, \mathbf{u}^*). \tag{1.71}$$

Здесь введены обозначения

$$\Pi_1(\mathbf{u}^*, \bar{\sigma}) = \frac{1}{2}\int_{V_1} (\mathbf{A}\mathbf{u}^*)^{\mathrm{T}}\mathbf{D}\mathbf{A}\mathbf{u}^* dV - \int_{V_1} (\mathbf{u}^*)^{\mathrm{T}} \rho dV -$$
$$- \int_{S_{\sigma_1}} (\mathbf{u}^*) \mathbf{g}_S dS - \int_{\Omega} (\mathbf{u}^*)^{\mathrm{T}} \mathbf{A}_{\Omega}^{\mathrm{T}} \bar{\sigma} d\Omega \text{ — функционал Лагранжа;} \tag{1.72}$$

$$\Pi_2(\bar{\sigma}, \mathbf{u}^*) = -\frac{1}{2}\int_{V_2} (\bar{\sigma})^{\mathrm{T}} \mathbf{D}^{-1}\sigma dV +$$
$$+ \int_{S_{u_2}} \mathbf{u}_S^{\mathrm{T}} \mathbf{A}_S^{\mathrm{T}} \bar{\sigma} dS - \int_{\Omega} (\mathbf{u}^*)^{\mathrm{T}} \mathbf{A}_{\Omega}^{\mathrm{T}} \bar{\sigma} d\Omega \tag{1.73}$$

— функционал Кастильяно; $\mathbf{u}^*$ — вектор кинематически возможных перемещений, определенный в области V_1; $\bar{\sigma}$ — вектор статически возможных напряжений, определенный в области V_2; $\mathbf{A}$ — матрица операции дифференцирования; $\mathbf{D}$ — матрица упругости; ρ — плотность тела в данной точке; g_S — заданный вектор поверхностных сил на области $S_{\sigma_1} \subset S_1$; $\mathbf{A}_{\Omega}(\mathbf{A}_S)$ — матрица направляющих косинусов к поверхности $\Omega(S)$; $\mathbf{u}_S$ — заданный вектор перемещений на $S_{u_2} \subset S_2$.

Заметим, что в функционале $\Pi_1(\mathbf{u}^*, \bar{\sigma})$ вариации подлежит вектор $\mathbf{u}^*$, а $\bar{\sigma}$ является параметром, в $\Pi_2(\bar{\sigma}, \mathbf{u}^*)$ — наоборот. На поверхности Ω положительной считается нормаль, внешняя к области V_1, поэтому интеграл по Ω в Π_2 взят со знаком минус.

Разобьем тело на конечные элементы так, чтобы поверхность Ω проходила по межэлементным границам и не возникало разрывов между отдельными элементами. Введем аппроксимацию перемещений в области V_1 и напряжений в области V_2

$$\mathbf{u}_i^* = N\mathbf{q}_i \text{ по области } V_{1i}, \tag{1.74}$$

$$\bar{\sigma}_k = \mathbf{P}\beta_k + \mathbf{P}_{FK} \text{ по области } V_{2i}, \tag{1.75}$$

где $\mathbf{q}_i$ — узловые перемещения КЭ; N — функция формы; $\mathbf{P}$ — матрица аппроксимации напряжений, удовлетворяющая однородным уравнениям равновесия и статическим граничным условиям; β_k — параметры напряжений; $\mathbf{P}_{FK}$ — вектор напряжений, удовлетворяющий неоднородным уравнениям равновесия и статическим граничным условиям (частное решение неоднородных уравнений). Предположим для простоты, что на поверхности S_{u_2} кинематические условия однородны

$$\mathbf{u}_s = 0 \text{ по области } S_{u_2}. \tag{1.76}$$

Тогда соответствующий интеграл по S_{u_2} (1.73) пропадает. После подстановки соотношений (1.74)—(1.76) в выражения (1.72) и (1.73) последние принимают вид

$$\Pi_1(\mathbf{q}_i, \boldsymbol{\beta}_l) = \sum_i \left(\frac{1}{2}\mathbf{q}_i^{\mathrm{T}}\mathbf{K}_i\mathbf{q}_i - \mathbf{q}_i^{\mathrm{T}}\mathbf{F}_i\right) - \sum_j \mathbf{q}_j^{\mathrm{T}}\mathbf{T}_j - \sum_l \mathbf{q}_l^{\mathrm{T}}\mathbf{M}_l\boldsymbol{\beta}_l, \quad (1.77)$$

$$\Pi_2(\boldsymbol{\beta}_k, \mathbf{q}) = -\sum_k \left(\frac{1}{2}\boldsymbol{\beta}_k^{\mathrm{T}}\mathbf{H}_k\boldsymbol{\beta}_k + \frac{1}{2}\mathbf{R}_k + \boldsymbol{\beta}_k^{\mathrm{T}}\mathbf{Q}_k\right) - \sum_l \mathbf{q}_l^{\mathrm{T}}\mathbf{M}_l\boldsymbol{\beta}_l, \quad (1.78)$$

где $\mathbf{K}_i = \int\limits_{V_{1i}} (\mathbf{A}N)^{\mathrm{T}} D(\mathbf{A}N)\,dV$; $\mathbf{F}_i = \int\limits_{V_{1i}} N^{\mathrm{T}}\rho\,dV$;

$$\mathbf{T}_j = \int\limits_{S_{\sigma_{1j}}} N^{\mathrm{T}} g_s\,dS; \quad \mathbf{H}_k = \int\limits_{V_{2k}} \mathbf{P}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{P}\,dV; \quad (1.79)$$

$$\mathbf{Q}_k = \int\limits_{V_{2k}} \mathbf{P}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{P}_{FK}\,dV; \quad \mathbf{M}_l = \int\limits_{\Omega_l} N^{\mathrm{T}}\mathbf{A}_\Omega^{\mathrm{T}}\mathbf{P}\,d\Omega;$$

$\mathbf{K}_i$ — матрица жесткости i-го КЭ; $\mathbf{H}_k$ — матрица податливости k-го КЭ; $\mathbf{F}_i$ и $\mathbf{T}_j$ — векторы узловой нагрузки; $\mathbf{Q}_k$ — вектор обобщенных перемещений, соответствующих параметрам $\boldsymbol{\beta}_k$; $\mathbf{M}_l$ — матрица связи обобщенных параметров напряжений с узловыми усилиями на границе Ω. Отметим, что суммирование по l ведется по межэлементным границам, принадлежащим поверхности раздела Ω, при этом $\mathbf{q}_l$ — перемещения узлов, лежащих на Ω; $\boldsymbol{\beta}_l$ — те параметры напряжений в КЭ, примыкающих к поверхности Ω со стороны области V_2, которые определяют напряжения на этой поверхности.

Поле перемещений в области V_1 и поле напряжений в области V_2 должны быть непрерывны, откуда следует, что аппроксимация внутри отдельных КЭ (см. формулы (1.74), (1.75)) не является независимой. Можно записать

$$\mathbf{q}_i = \mathbf{C}_i\mathbf{q}; \quad (1.80)$$

$$\boldsymbol{\beta}_k = \mathbf{B}_k\boldsymbol{\beta}, \quad (1.81)$$

где $\mathbf{C}_i$, $\mathbf{B}_k$ — матрицы инциденций, состоящие из нулей и единиц; $\mathbf{q}$ — вектор узловых перемещений для области V_1; $\boldsymbol{\beta}$ — вектор параметров напряжений в области V_2 (размеры векторов равны соответственно степеням кинематической и статической неопределимости для областей V_1 и V_2). Очевидно, в их число входят перемещения граничных узлов на Ω и силовые связи вдоль этой же границы. Обозначим их соответственно $\mathbf{q}_\Omega$ и $\boldsymbol{\beta}_\Omega$. Можно записать

$$\mathbf{q}_\Omega = \mathbf{G}_1 q; \quad \boldsymbol{\beta}_\Omega = \mathbf{G}_2\boldsymbol{\beta}, \quad (1.82)$$

где $\mathbf{G}_1$ и $\mathbf{G}_2$ — матрицы инциденций.

Аналогично (1.80) представим вектор перемещений j-го КЭ, выходящего на поверхность S_{σ_1}:

$$\mathbf{q}_j = \tilde{\mathbf{C}}_j\mathbf{q}. \quad (1.83)$$

Для поверхности Ω можно записать

$$\mathbf{q}_l = \mathbf{S}_{1l}\mathbf{q}_\Omega, \quad \boldsymbol{\beta}_l = \mathbf{S}_{2l}\boldsymbol{\beta}_\Omega. \tag{1.84}$$

Матрицы $\mathbf{S}_{1l}$, $\mathbf{S}_{2l}$ имеют смысл, аналогичный матрицам $\mathbf{C}_j$, $\mathbf{B}_k$.

Введем следующие матрицы:

$$\begin{aligned}
&\mathbf{C}=\{\mathbf{C}_1\mathbf{C}_2\ldots\mathbf{C}_i\}, \quad \mathbf{B}=\{\mathbf{B}_1\mathbf{B}_2\ldots\mathbf{B}_k\ldots\},\\
&\mathbf{F}=\{\mathbf{F}_1\ldots\mathbf{F}_i\ldots\}; \quad \mathbf{T}=\{\mathbf{T}_1\mathbf{T}_2\ldots\mathbf{T}_j\},\\
&\mathbf{Q}=\{\mathbf{Q}_1\ldots\mathbf{Q}_k\ldots\}, \quad \mathbf{R}=\Sigma\mathbf{R}_k;\\
&\mathbf{K}=\lfloor\mathbf{K}_1\mathbf{K}_2\ldots\mathbf{K}_i\ldots\rfloor, \quad \mathbf{H}=\lfloor\mathbf{H}_1\mathbf{H}_2\ldots\mathbf{H}_k\ldots\rfloor;\\
&\mathbf{M}=\lfloor\mathbf{M}_1\mathbf{M}_2\ldots\mathbf{M}_l\ldots\rfloor, \quad \tilde{\mathbf{C}}=\{\tilde{\mathbf{C}}_1\tilde{\mathbf{C}}_2\ldots\tilde{\mathbf{C}}_j\ldots\};\\
&\mathbf{S}_1=\{\mathbf{S}_{11}\mathbf{S}_{12}\ldots\mathbf{S}_{1l}\ldots\}, \quad \mathbf{S}_2=\{\mathbf{S}_{21}\mathbf{S}_{22}\ldots\mathbf{S}_{2l}\ldots\}.
\end{aligned} \tag{1.85}$$

Здесь $\{\ldots\}$ — вектор-столбец, $\lfloor\ \rfloor$ — диагональная матрица.

С учетом введенных обозначений имеем

$$\Pi_1(\mathbf{q}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{q}^\mathrm{T}\bar{\mathbf{K}}\mathbf{q} - \mathbf{q}^\mathrm{T}\mathbf{L}\boldsymbol{\beta} - \mathbf{q}^\mathrm{T}\mathbf{F}_*; \tag{1.86}$$

$$\Pi_2(\boldsymbol{\beta}, \mathbf{q}) = -\frac{1}{2}\boldsymbol{\beta}^\mathrm{T}\mathbf{f}\boldsymbol{\beta} - \frac{1}{2}\mathbf{R} - \boldsymbol{\beta}^\mathrm{T}\mathbf{Q}_* - \mathbf{q}^\mathrm{T}\mathbf{L}\boldsymbol{\beta}. \tag{1.87}$$

Здесь дополнительно введены матрицы

$$\begin{aligned}
&\bar{\mathbf{K}}=\mathbf{C}^\mathrm{T}\mathbf{K}\mathbf{C}, \quad \mathbf{f}=\mathbf{B}^\mathrm{T}\mathbf{H}\mathbf{B}, \quad \mathbf{F}_*=\mathbf{C}^\mathrm{T}\mathbf{F}+\tilde{\mathbf{C}}^\mathrm{T}\mathbf{T};\\
&\mathbf{Q}_*=\mathbf{B}^\mathrm{T}\mathbf{Q}, \quad \mathbf{L}=\mathbf{G}_1^\mathrm{T}\mathbf{S}_1^\mathrm{T}\mathbf{M}\mathbf{S}_2\mathbf{G}_2=\mathbf{G}_1^\mathrm{T}\mathbf{L}_{12}\mathbf{G}_2.
\end{aligned} \tag{1.88}$$

Матрицы $\bar{\mathbf{K}}$ и $\mathbf{f}$ являются соответственно матрицей жесткости первой области (подконструкции) и матрицей податливости второй области (подконструкции).

Условиями стационарности функционалов Π_1 и Π_2 являются следующие уравнения:

$$\frac{\partial\Pi_1(\mathbf{q}, \boldsymbol{\beta})}{\partial\mathbf{q}} = \bar{\mathbf{K}}\mathbf{q} - \mathbf{L}\boldsymbol{\beta} - \mathbf{F}_* = 0; \tag{1.89}$$

$$\frac{\partial\Pi_2(\boldsymbol{\beta}, q)}{\partial\boldsymbol{\beta}} = \mathbf{f}\boldsymbol{\beta} - \mathbf{L}^\mathrm{T}\mathbf{q} - \mathbf{Q}_* = 0 \tag{1.90}$$

или в матричной форме

$$\begin{bmatrix} \mathbf{K} & -\mathbf{L} \\ \mathbf{L}^\mathrm{T} & \mathbf{f} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_* \\ -\mathbf{Q}_* \end{bmatrix}. \tag{1.91}$$

Матрица системы (1.91) кососимметрична.

Неизвестные векторы $\mathbf{q}$ и $\boldsymbol{\beta}$ отнесены к разным частям тела, взаимосвязь областей осуществляется по сравнительно небольшому числу элементов, благодаря чему матрица $\mathbf{L}$ является редкозаполненной.

Пронумеруем неизвестные так, что сначала перечисляются внутренние, а затем граничные неизвестные

$$\mathbf{q}=\{\mathbf{q}_b\mathbf{q}_\Omega\},\ \boldsymbol{\beta}=\{\boldsymbol{\beta}_b,\ \boldsymbol{\beta}_\Omega\}. \tag{1.92}$$

При этом матрицы $\mathbf{G}_1$ и $\mathbf{G}_2$ уравнений (1.82) имеют следующую структуру:

$$\mathbf{G}_1=\|0\ \mathbf{E}_1\|,\quad \mathbf{G}_2=\|0\ \mathbf{E}_2\|, \tag{1.93}$$

где $\mathbf{E}_1$, $\mathbf{E}_2$ — единичные матрицы; 0 — нулевые прямоугольные матрицы.

Матрица $\mathbf{L}$ с учетом (1.88), (1.93) примет вид

$$\mathbf{L}=[0\ \mathbf{E}_1]^{\mathrm{T}}\mathbf{L}_{12}[0\ \mathbf{E}_2]=\begin{bmatrix}0 & 0\\ 0 & \mathbf{L}_{12}\end{bmatrix}. \tag{1.94}$$

В соответствии с делением неизвестных на внутренние и граничные перепишем уравнение (1.91) в форме

$$\begin{bmatrix}\overline{\mathbf{K}}_{bb} & \overline{\mathbf{K}}_{b\Omega} & 0 & 0\\ \overline{\mathbf{K}}^{\mathrm{T}}_{b\Omega} & \overline{\mathbf{K}}_{\Omega\Omega} & 0 & -\mathbf{L}_{12}\\ 0 & 0 & \mathbf{f}_{bb} & \mathbf{f}_{b\Omega}\\ 0 & \mathbf{L}^{\mathrm{T}}_{12} & \mathbf{f}^{\mathrm{T}}_{b\Omega} & \mathbf{f}_{\Omega\Omega}\end{bmatrix}\begin{bmatrix}\mathbf{q}_b\\ \mathbf{q}_\Omega\\ \boldsymbol{\beta}_b\\ \boldsymbol{\beta}_\Omega\end{bmatrix}=\begin{bmatrix}\mathbf{F}_{*b}\\ \mathbf{F}_{*\Omega}\\ -\mathbf{Q}_{*b}\\ -\mathbf{Q}_{*b}\end{bmatrix}. \tag{1.95}$$

Исключим внутренние переменные из системы

$$\mathbf{q}_b=\overline{\mathbf{K}}^{-1}_{bb}\mathbf{F}_{*b}-\overline{\mathbf{K}}^{-1}_{bb}\overline{\mathbf{K}}_{b\Omega}\mathbf{q}_\Omega; \tag{1.96}$$

$$\boldsymbol{\beta}_b=-\mathbf{f}^{-1}_{bb}\mathbf{Q}_{*b}-\mathbf{f}^{-1}_{bb}\mathbf{f}_{b\Omega}\boldsymbol{\beta}_\Omega. \tag{1.97}$$

Здесь обращаемые матрицы являются симметричными, положительно определенными ленточными матрицами.

Подставляя (1.96), (1.97) в оставшиеся уравнения, из (1.95) имеем

$$\begin{aligned}&\overline{\mathbf{K}}^{*}_{\Omega\Omega}\mathbf{q}_\Omega-\mathbf{L}_{12}\boldsymbol{\beta}_\Omega=\mathbf{F}_{*\Omega}-\overline{\mathbf{K}}^{\mathrm{T}}_{b\Omega}\overline{\mathbf{K}}^{-1}_{bb}\mathbf{F}_{*b};\\ &\mathbf{L}^{\mathrm{T}}_{12}\mathbf{q}_\Omega+\mathbf{f}^{*}_{\Omega\Omega}\boldsymbol{\beta}_\Omega=-\mathbf{Q}_{*\Omega}+\mathbf{f}^{\mathrm{T}}_{b\Omega}\mathbf{f}^{-1}_{bb}\mathbf{Q}_{*b},\end{aligned} \tag{1.98}$$

где

$$\begin{aligned}&\overline{\mathbf{K}}^{*}_{\Omega\Omega}=\overline{\mathbf{K}}_{\Omega\Omega}-\overline{\mathbf{K}}^{\mathrm{T}}_{b\Omega}\overline{K}^{-1}_{bb}\overline{\mathbf{K}}_{b\Omega};\\ &\mathbf{f}^{*}_{\Omega\Omega}=\mathbf{f}_{\Omega\Omega}-\mathbf{f}^{\mathrm{T}}_{b\Omega}\mathbf{f}^{-1}_{bb}\mathbf{f}_{b\Omega}.\end{aligned} \tag{1.99}$$

Назовем матрицу $\overline{\mathbf{K}}^{*}_{\Omega\Omega}$ матрицей жесткости первого суперэлемента (СЭ), а $\mathbf{f}^{**}_{\Omega\Omega}$ — матрицей податливости второго СЭ. Блочное исключение неизвестных согласно формулам (1.96)—(1.98) по физическому смыслу может трактоваться как определение напряженно-деформированного состояния суперэлементным методом. При этом один из СЭ рассчитывается методом перемещений, другой — методом сил. Естественно назвать такой подход смешанным суперэлементным методом.

Матрицы (1.99) вычисляют независимо для каждого СЭ. Граничные перемещения и усилия определяют путем решения системы (1.98) относительно невысокого порядка. Тем не менее и здесь удобнее применить блочное исключение. В зависимости от того, какие переменные — перемещения или усилия — исключаются в первую очередь, приходим к сочленению соответственно либо методом сил, либо методом перемещений.

Исключим из первого матричного уравнения (1.98) вектор

$$\mathbf{q}_{\Omega}=(\overline{\mathbf{K}}^{*}_{\Omega\Omega})^{-1}(\mathbf{F}_{*\Omega}-\overline{\mathbf{K}}^{\mathrm{T}}_{b\Omega}\mathbf{F}_{*b})+(\overline{\mathbf{K}}^{**}_{\Omega\Omega})^{-1}\mathbf{L}_{12}\boldsymbol{\beta}_{\Omega}. \qquad (1.100)$$

Подставив найденное выражение во второе матричное уравнение (1.98), получим

$$[\mathbf{f}^{*}_{\Omega\Omega}+\mathbf{L}^{\mathrm{T}}_{12}(\overline{\mathbf{K}}^{**}_{\Omega\Omega})^{-1}\mathbf{L}_{12}]\boldsymbol{\beta}_{\Omega}=-\mathbf{Q}_{*\Omega}+\mathbf{f}^{\mathrm{T}}_{b\Omega}\mathbf{f}^{-1}_{bb}\mathbf{Q}_{*b}-$$
$$-\mathbf{L}^{\mathrm{T}}_{12}(\overline{\mathbf{K}}^{*}_{\Omega\Omega})^{-1}(\mathbf{F}_{*\Omega}-\overline{\mathbf{K}}^{\mathrm{T}}_{b\Omega}\mathbf{K}^{-1}_{bb}\mathbf{F}_{*b}). \qquad (1.101)$$

Из этого уравнения определяются усилия взаимодействия СЭ. По смыслу оно является уравнением совместности деформаций СЭ.

Общая схема решения в этом случае выглядит следующим образом. Рассчитываются изолированные СЭ. При этом $\boldsymbol{\beta}_{\Omega}=0$. Перемещения первого СЭ определяются по формулам

$$\begin{aligned}\overline{\mathbf{q}}_{b}&=\overline{\mathbf{K}}^{-1}_{bb}\mathbf{F}_{*b}-\overline{\mathbf{K}}^{-1}_{bb}\overline{\mathbf{K}}_{b\Omega}\overline{\mathbf{q}}_{\Omega};\\ \mathbf{q}_{\Omega}&=(\overline{\mathbf{K}}^{*}_{\Omega\Omega})^{-1}(\mathbf{F}_{*\Omega}-\overline{\mathbf{K}}^{\mathrm{T}}_{b\Omega}\overline{\mathbf{K}}^{-1}_{bb}\mathbf{F}_{*b}).\end{aligned} \qquad (1.102)$$

Впрочем, здесь необязательно разделять перемещения на внутренние и граничные. Перемещения могут быть найдены из решения системы уравнений

$$\overline{\mathbf{K}}_{\overline{\mathbf{q}}}=\mathbf{F}_{*}. \qquad (1.103)$$

Матрица жесткости $\overline{\mathbf{K}}$ должна быть невырождена (подконструкция закреплена).

Во втором СЭ параметры напряжений определяются по формулам

$$\boldsymbol{\beta}_{b}=-\mathbf{f}^{-1}_{bb}\mathbf{Q}_{*b};\quad \overline{\boldsymbol{\beta}}_{\Omega}=0. \qquad (1.104)$$

Усилия сочленения $\boldsymbol{\beta}$ находятся из уравнений (1.104).

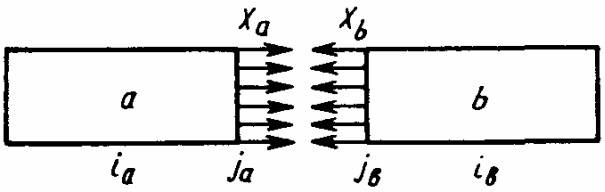
Дополнительные перемещения и усилия в суперэлементах с учетом их взаимодействия определяются по следующим формулам:

$$\begin{aligned}&\mathbf{q}_{\Omega}=(\overline{\mathbf{K}}^{**}_{\Omega\Omega})^{-1}\mathbf{L}_{12}\overline{\overline{\boldsymbol{\beta}}}_{\Omega},\quad \mathbf{q}=-\overline{\mathbf{K}}^{-1}_{bb}\overline{\mathbf{K}}_{b\Omega}\overline{\overline{\mathbf{q}}}_{\Omega};\\ &\overline{\overline{\boldsymbol{\beta}}}_{b}=-\mathbf{f}^{-1}_{bb}\mathbf{f}_{b\Omega}\overline{\overline{\boldsymbol{\beta}}}_{\Omega}.\end{aligned} \qquad (1.105)$$

Окончательное решение является суперпозицией двух найденных:

$$\mathbf{q}=\overline{\mathbf{q}}+\overline{\overline{\mathbf{q}}},\quad \boldsymbol{\beta}=\overline{\boldsymbol{\beta}}+\overline{\overline{\boldsymbol{\beta}}}. \qquad (1.106)$$

Заметим, что произведение матрицы $(\mathbf{K}^{*}_{\Omega\Omega})^{-1}$ слева на некоторую матрицу может быть получено с помощью решения системы урав-



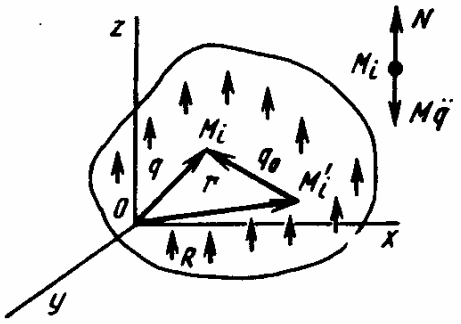


Рис. 1.1. Объект расчета при двух суперэлементах

Рис. 1.2. Построение вектора полного перемещения точек

нений (1.103) при соответствующей правой части и последующей операции перестроения (выборки строк). Группировка неизвестных на внутренние и граничные является условной и на практике не всегда выгодной. Аналогично сочленение СЭ может быть осуществлено по методу перемещений.

Расчет на собственные и вынужденные колебания

Расчленим конструкцию, находящуюся в состоянии свободных колебаний, на два суперэлемента (рис. 1.1). Внутренние узлы суперэлементов обозначим индексом i, а все направления перемещений узлов границы — j. Каждый суперэлемент рассмотрим в состоянии вынужденных колебаний под действием гармонических лишних неизвестных $\mathbf{X}$, приложенных к его границам. Поэтому вектор внешних сил имеет вид

$$\mathbf{N}=\begin{vmatrix}\mathbf{N}_i\\ \mathbf{N}_j\end{vmatrix}=\begin{vmatrix}0\\ \mathbf{X}\end{vmatrix}. \qquad (1.107)$$

Для объединения суперэлементов в единую конструкцию на границах суперэлементов потребуем выполнения условий

$$\mathbf{q}_{ja}=\mathbf{q}_{jb}; \qquad (1.108)$$

$$\mathbf{X}_a=-\mathbf{X}_b. \qquad (1.109)$$

Кроме того, должно выполняться условие равенства частот вынужденных колебаний суперэлементов частоте собственных колебаний конструкции

$$\mathbf{P}_a=\mathbf{P}_b=\mathbf{P}. \qquad (1.110)$$

Прежде всего остановимся на вопросе вынужденных колебаний суперэлементов. Рассмотрим незакрепленный суперэлемент, рассчитываемый методом сил, перемещающийся прямолинейно в связанной с ним инерциальной системе отсчета (рис. 1.2). Полные перемещения точек системы запишем в виде

$$\mathbf{q}=\mathbf{r}+\mathbf{q}_0, \qquad (1.111)$$

где $\mathbf{r}$ — вектор упругих перемещений; $\mathbf{q}_0$ — вектор перемещений конструкции как твердого тела.

Запишем

$$\mathbf{q}=\rho\mathbf{z}, \qquad (1.112)$$

где ϱ — матрица форм колебаний конструкции; $\mathbf{z}$ — обобщенные координаты, которые в случае ортонормированности форм колебаний (т. е. при $\varrho^{\mathrm{T}}\mathbf{M}\varrho=\mathbf{E}$, где $\mathbf{E}$ — единичная матрица) представляют из себя нормальные координаты. Пусть

$$\mathbf{q}=\rho_e\mathbf{z}_e+\rho_0\mathbf{z}_0, \tag{1.113}$$

где ϱ_e — матрица упругих форм колебаний, а ϱ_0 — матрица форм перемещений как твердого тела. Опуская ряд промежуточных вычислений [7], запишем выражение для ускорений

$$\ddot{\mathbf{q}}=\ddot{\mathbf{r}}+\mathbf{A}_0\ddot{\eta} \tag{1.114}$$

и

$$\mathbf{R}=\mathbf{N}-\mathbf{M}\ddot{\mathbf{q}}, \tag{1.115}$$

где $\mathbf{A}_0$ — матрица статического равновесия конструкции; η — обобщенные координаты.

Получим также следующие выражения [7]:

$$\mathbf{r}=\mathbf{F}(\mathbf{N}-\mathbf{M}\ddot{\mathbf{q}}), \tag{1.116}$$

$$\ddot{\eta}=\mathbf{M}_0^{-1}\mathbf{A}_0^{\mathrm{T}}(\mathbf{N}-\mathbf{M}\ddot{\mathbf{r}}), \tag{1.117}$$

где $\mathbf{F}$ — матрица податливости конструкции во введенной инерциальной системе координат, а $\mathbf{M}_0=\mathbf{A}_0^{\mathrm{T}}\mathbf{M}\mathbf{A}_0$ — обобщенная матрица инерции свободной конструкции.

Из уравнения (1.114) получим

$$\ddot{\mathbf{q}}=\mathbf{C}\ddot{\mathbf{r}}+\mathbf{W}_0\mathbf{N}, \tag{1.118}$$

где

$$\mathbf{W}_0=\mathbf{A}_0(\mathbf{A}_0^{\mathrm{T}}\mathbf{M}\mathbf{A}_0)^{-1}\mathbf{A}_0^{\mathrm{T}}=\mathbf{A}_0\mathbf{M}^{-1}\mathbf{A}_0^{\mathrm{T}}; \tag{1.119}$$

$$\mathbf{C}=\mathbf{E}-\mathbf{A}_0(\mathbf{A}_0^{\mathrm{T}}\mathbf{M}\mathbf{A}_0)^{-1}\mathbf{A}_0^{\mathrm{T}}\mathbf{M}=\mathbf{E}-\mathbf{W}_0\mathbf{M}. \tag{1.120}$$

При нулевых начальных условиях $(\mathbf{q}(0)=\dot{\mathbf{q}}(0)=0)$ и отсутствии $\mathbf{N}$ имеем

$$\mathbf{q}=\mathbf{C}\mathbf{r}, \tag{1.121}$$

что свидетельствует о том, что матрица $\mathbf{C}$ преобразует перемещения закрепленной конструкции в упругие перемещения свободной конструкции (можно показать, что для нее выполняется тождество $\mathbf{C}^n=\mathbf{C}$, где $n=1, 2, \ldots$ [6]).

Из принципа возможных перемещений получим

$$\mathbf{R}=\mathbf{C}^{\mathrm{T}}\mathbf{Q}, \tag{1.122}$$

где $\mathbf{Q}$ — силы, связанные с $\mathbf{q}$ зависимостью

$$\mathbf{q}=\mathcal{F}\mathbf{Q}, \tag{1.123}$$

а $\mathbf{F}$ — матрица податливости незакрепленной конструкции. Используя уравнения (1.121), (1.122) и учитывая, что $\mathbf{r}=\mathbf{F}\mathbf{R}$, получим

$$\mathcal{F}=\mathbf{C}\mathbf{F}\mathbf{C}^{\mathrm{T}}. \tag{1.124}$$

Рассмотрим уравнение (1.118). Умножая его слева на **FM**, получим

$$\mathbf{FM}\ddot{\mathbf{q}}=\mathbf{FMC}\ddot{\mathbf{r}}+\mathbf{FMW}_0\mathbf{N}. \tag{1.125}$$

Матрица **MC** — симметричная, поэтому

$$\mathbf{MC}=\mathbf{C}^{\mathrm{T}}\mathbf{M}, \tag{1.126}$$

а уравнение (1.125) примет вид

$$\mathbf{FM}\ddot{\mathbf{q}}=\mathbf{FC}^{\mathrm{T}}\mathbf{M}\ddot{\mathbf{r}}+\mathbf{FMW}_0\mathbf{N}. \tag{1.127}$$

Умножая уравнения (1.115) слева на **F**, получим

$$\mathbf{FM}\ddot{\mathbf{q}}=\mathbf{FN}-\mathbf{FR}. \tag{1.128}$$

Приравнивая правые части уравнений (1.127), (1.128), после несложных преобразований получим дифференциальное уравнение вынужденных колебаний незакрепленной конструкции относительно перемещений **r** [7]

$$\mathbf{FC}^{\mathrm{T}}\mathbf{M}\ddot{\mathbf{r}}+\mathbf{r}=\mathbf{FC}^{\mathrm{T}}\mathbf{N}, \tag{1.129}$$

которому можно придать вид

$$\mathcal{F}\mathbf{M}\ddot{\mathbf{r}}+\mathbf{Cr}=\mathcal{F}\mathbf{N}. \tag{1.130}$$

Уравнение (1.130) с учетом того, что $\mathbf{r}=\boldsymbol{\varrho}_e\mathbf{z}_e$, запишем в виде

$$\mathcal{F}\mathbf{M}\boldsymbol{\rho}_e\ddot{\mathbf{z}}_e+\mathbf{C}\boldsymbol{\rho}_e\mathbf{z}_e=\mathcal{F}\mathbf{N}. \tag{1.131}$$

При $\mathbf{N}=0$ и условии $\mathbf{r}=\bar{\mathbf{r}}e^{i\omega t}$ получаем задачу на собственные значения

$$(\mathcal{F}\mathbf{M}-\lambda\mathbf{C})\,\boldsymbol{\rho}_e\bar{\mathbf{z}}_e=0. \tag{1.132}$$

Из него следует тождество

$$\mathcal{F}\mathbf{M}\boldsymbol{\rho}_e=\mathbf{C}\boldsymbol{\rho}_e\boldsymbol{\Lambda} \text{ или } \mathcal{F}\mathbf{M}\boldsymbol{\rho}_e=\boldsymbol{\rho}_e\boldsymbol{\Lambda}. \tag{1.133}$$

Из (1.131) с учетом (1.133) получим

$$\boldsymbol{\Lambda}\ddot{\mathbf{z}}_e+\mathbf{z}_e=\boldsymbol{\Lambda}\boldsymbol{\rho}_e^{\mathrm{T}}\mathbf{N}, \tag{1.134}$$

или

$$\ddot{\mathbf{z}}_e+\boldsymbol{\Lambda}^{-1}\mathbf{z}_e=\boldsymbol{\rho}_e^{\mathrm{T}}\mathbf{N}, \tag{1.135}$$

являющееся несвязанной системой дифференциальных уравнений, записанных относительно нормальных координат $\mathbf{z}_e$. Приведем дифференциальное уравнение, описывающее перемещения незакрепленной конструкции в обобщенных координатах $\mathbf{z}_0$:

$$\ddot{\mathbf{z}}_0=\boldsymbol{\rho}_0^{\mathrm{T}}\mathbf{N}. \tag{1.136}$$

Из последних двух уравнений определяется частота вынужденных колебаний и неизвестные обобщенные координаты $\mathbf{z}_e$ и $\mathbf{z}_0$. Тогда полные перемещения конструкции будут определяться из уравнения (1.113).

Обсудим решение неполной собственной проблемы; это соответствует требованиям практики, так как обычно требуется знать ограниченное число форм. Для этого вектор перемещений представим в виде суммы

$$\mathbf{q}=\rho\mathbf{z}=\rho_U\mathbf{z}_U+\rho_V\mathbf{z}_V, \tag{1.137}$$

где индекс U — соответствует учитываемым, а V — отбрасываемым формам собственных колебаний.

В матрицу учитываемых форм ρ_U включим и формы перемещений как твердого тела ρ_0.

Если внешняя сила $\mathbf{N}$ действует по гармоническому закону

$$\mathbf{N}=\overline{\mathbf{N}}e^{ipt}, \tag{1.138}$$

то при установившихся колебаниях из уравнения (1.135) находим

$$\bar{\mathbf{z}}_s=\frac{\rho_s^{\mathrm{T}}\overline{\mathbf{N}}}{\omega_s^2-p^2}, \quad s=\overline{1,m}.$$

Тогда

$$\rho_V\bar{\mathbf{z}}_V\cong\sum_{s=u+1}^{m}\rho_s\frac{\rho_s^{\mathrm{T}}\overline{\mathbf{N}}}{\omega_s^2-p^2}. \tag{1.139}$$

Учитывая, что $\omega_s^2-p^2\cong\omega_s^2$ $(s=\overline{u+1,m})$, запишем

$$\rho_V\bar{\mathbf{z}}_V\cong\sum_{s=u+1}^{m}\rho_s\frac{\rho_s^{\mathrm{T}}\overline{\mathbf{N}}}{\omega_s^2}=(\rho_V\Lambda_V\rho_V^{\mathrm{T}})\overline{\mathbf{N}}, \tag{1.140}$$

где

$$\Lambda_V=\left[\frac{1}{\omega_{u+1}^2}\ \frac{1}{\omega_{u+2}^2}\cdots\frac{1}{\omega_m^2}\right]. \tag{1.141}$$

Таким образом, вклад отброшенных форм колебаний приближенно выражается соотношением (1.140). Чтобы не определять матрицу ρ_V, выразим вклад этих форм через удерживаемые. Умножив (1.133) справа на ρ_e^{T}, получим

$$\mathcal{F}\mathbf{M}\rho_e\rho_e^{\mathrm{T}}=\rho_e\Lambda\rho_e^{\mathrm{T}}, \tag{1.142}$$

из чего следует [7], что

$$\mathcal{F}=\rho_e\Lambda\rho_e^{\mathrm{T}}=\rho_U\Lambda_U\rho_U^{\mathrm{T}}+\rho_V\Lambda_V\rho_V^{\mathrm{T}}. \tag{1.143}$$

Поэтому

$$\rho_V\bar{\mathbf{z}}_V\cong(\rho_V\Lambda_V\rho_V^{\mathrm{T}})\overline{\mathbf{N}}=\mathcal{F}_V\overline{\mathbf{N}}, \tag{1.144}$$

где

$$\mathcal{F}_V=\mathcal{F}-\rho_U\Lambda_U\rho_U^{\mathrm{T}} \tag{1.145}$$

— остаточная податливость для форм колебаний, не удерживаемых явным образом.

Тогда

$$\mathbf{q}=\rho_U\bar{\mathbf{z}}_U+\mathcal{F}_V\bar{\mathbf{N}}, \tag{1.146}$$

а из уравнения (1.135) получим уравнение сокращенного размера

$$(\Lambda_U^{-1}-p^2\mathbf{E})\,\bar{\mathbf{z}}_U=\rho_U^{\mathrm{T}}\mathbf{N}. \tag{1.147}$$

Для суперэлемента, рассчитываемого методом перемещений, уравнение вынужденных колебаний имеет вид

$$\mathbf{K}\mathbf{q}+\mathbf{M}\ddot{\mathbf{q}}=\mathbf{N}. \tag{1.148}$$

С учетом (1.112) и условия ортонормированности, оно сводится к уравнениям, полностью совпадающим с (1.135), (1.136). При решении неполной собственной проблемы учет отброшенных высших форм осуществляется так же, как и в методе сил, по формуле (1.146). Это позволяет вести стыковку суперэлементов независимо от того, каким методом они рассчитаны. Отметим, что в методе перемещений для пользования формулой (1.145) нужно знать только некоторые столбцы матрицы податливости.

Перейдем к стыковке суперэлементов. С учетом (1.107) уравнения (1.146) и (1.147) для граничных узлов суперэлемента принимают вид

$$\bar{\mathbf{q}}_j=\rho_{Uj}\bar{\mathbf{z}}_U+\mathcal{F}_{Vjj}\bar{\mathbf{N}}_j; \tag{1.149}$$

$$(\Lambda_U^{-1}-p^2\mathbf{E})\,\bar{\mathbf{z}}_U=\rho_{Uj}^{\mathrm{T}}\bar{\mathbf{N}}_j. \tag{1.150}$$

Для полных перемещений уравнение (1.146) запишем в виде

$$\bar{\mathbf{q}}=\rho_U\bar{\mathbf{z}}_U+\mathcal{F}_{Vj}\bar{\mathbf{N}}_j. \tag{1.151}$$

В связи с этим условие (1.108) будет иметь вид

$$\rho_{Uja}\bar{\mathbf{z}}_{Ua}+\mathcal{F}_{Ujja}\bar{\mathbf{N}}_{ja}=\rho_{Ujb}\bar{\mathbf{z}}_{Ub}+\mathcal{F}_{Ujjb}\bar{\mathbf{N}}_{jb}, \tag{1.152}$$

откуда с учетом (1.109) находим

$$\bar{\mathbf{N}}_{ja}=\mathbf{K}_{jj}(\rho_{Ujb}-\rho_{Uja}\bar{\mathbf{z}}_{Ua}), \tag{1.153}$$

где

$$\mathbf{K}_{jj}=(\mathcal{F}_{Vjja}+\mathcal{F}_{Vjjb})^{-1}. \tag{1.154}$$

Тогда уравнения колебаний суперэлементов типа (1.150) запишутся так

$$\begin{aligned}(\Lambda_{Ua}^{-1}-p^2\mathbf{E})\,\bar{\mathbf{z}}_{Ua}&=\rho_{Uja}^{\mathrm{T}}\mathbf{K}_{jj}(\rho_{Ujb}\bar{\mathbf{z}}_{Ub}-\rho_{Uja}\bar{\mathbf{z}}_{Ua});\\(\Lambda_{Ub}^{-1}-p^2\mathbf{E})\,\bar{\mathbf{z}}_{Ub}&=\rho_{Ujb}^{\mathrm{T}}\mathbf{K}_{jj}(\rho_{Uja}\bar{\mathbf{z}}_{Ua}-\rho_{Ujb}\bar{\mathbf{z}}_{Ub}).\end{aligned} \tag{1.155}$$

Представим эти уравнения в матричной форме

$$[(\Lambda^{-1}+\mathbf{S}^{\mathrm{T}}\mathbf{K}_{jj}\mathbf{S})-p^2\mathbf{E}]\,\bar{\mathbf{z}}=0, \tag{1.156}$$

где

$$\Lambda^{-1}=\lfloor\Lambda_{Ua}^{-1}\Lambda_{Ub}^{-1}\rfloor,\quad \mathbf{S}=\lfloor-\rho_{Uja}\rho_{Ujb}\rfloor,\quad \bar{\mathbf{z}}=\{\bar{\mathbf{z}}_{Ua}\bar{\mathbf{z}}_{Ub}\}. \tag{1.157}$$

Из уравнения (1.156) определяются частоты колебаний p и обобщенные координаты $\mathbf{z}$. Тогда соотношение (1.153) с учетом введен-

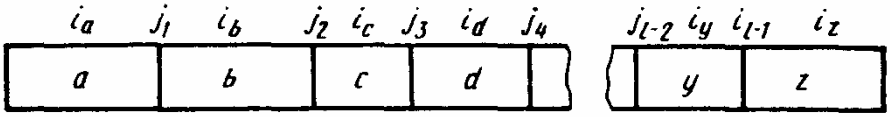


Рис. 1.3. Объект расчета при произвольном числе суперэлементов

ных обозначений принимает вид

$$\bar{\mathbf{N}}_j = \mathbf{K}_{jj}\mathbf{S}\bar{\mathbf{z}}. \tag{1.158}$$

При этом уравнения (1.151) для каждого суперэлемента в блочном виде будут следующими:

$$\begin{aligned} \bar{\mathbf{q}}_a &= \begin{bmatrix} \mathbf{q}_{ia} \\ \mathbf{q}_{ja} \end{bmatrix} = \begin{bmatrix} \rho_{Uia} & 0 \\ \rho_{Uja} & 0 \end{bmatrix} \bar{\mathbf{z}} + \begin{bmatrix} \mathcal{F}_{Vija} \\ \mathcal{F}_{Vjja} \end{bmatrix} \mathbf{K}_{jj}\mathbf{S}\bar{\mathbf{z}}; \\ \bar{\mathbf{q}}_b &= \begin{bmatrix} \bar{\mathbf{q}}_{ib} \\ \bar{\mathbf{q}}_{jb} \end{bmatrix} = \begin{bmatrix} 0 & \rho_{Uib} \\ 0 & \rho_{Ujb} \end{bmatrix} \bar{\mathbf{z}} - \begin{bmatrix} \mathcal{F}_{Vijb} \\ \mathcal{F}_{Vjjb} \end{bmatrix} \mathbf{K}_{jj}\mathbf{S}\bar{\mathbf{z}}. \end{aligned} \tag{1.159}$$

Для граничных точек суперэлементов соблюдаются условия (1.108). Поэтому при определении окончательных значений перемещений конструкции в граничных точках их значения можно взять из любого уравнения системы (1.159), например, из первого. Тогда

$$\bar{\mathbf{q}} = \begin{bmatrix} \bar{\mathbf{q}}_{ia} \\ \bar{\mathbf{q}}_{ja} \\ \bar{\mathbf{q}}_{ib} \end{bmatrix} = \left(\begin{bmatrix} \rho_{Uia} & 0 \\ \rho_{Uja} & 0 \\ 0 & \rho_{Uib} \end{bmatrix} + \begin{bmatrix} \mathcal{F}_{Vija} \\ \mathcal{F}_{Vjja} \\ -\mathcal{F}_{Vijb} \end{bmatrix} \mathbf{K}_{jj}\mathbf{S} \right) \bar{\mathbf{z}}. \tag{1.160}$$

Формы собственных колебаний конструкции определим нормированием столбцов матрицы перемещений

$$\rho = \| \bar{\mathbf{q}} \| . \tag{1.161}$$

Если конструкция разбита на l суперэлементов (рис. 1.3), то на их границах выполняются условия

$$\bar{\mathbf{q}}^a_{j_1} = \bar{\mathbf{q}}^b_{j_1},\ \bar{\mathbf{q}}^b_{j_2} = \bar{\mathbf{q}}^c_{j_2}, \ldots, \bar{\mathbf{q}}^y_{j_{l-1}} = \bar{\mathbf{q}}^z_{j_{l-1}}; \tag{1.162}$$

$$\mathbf{N}^a_{j_1} = -\bar{\mathbf{N}}^b_{j_1},\ \mathbf{N}^b_{j_2} = -\mathbf{N}^c_{j_2}, \ldots, \bar{\mathbf{N}}^y_{j_{l-1}} = -\bar{\mathbf{N}}^z_{j_{l-1}}. \tag{1.163}$$

Из этих уравнений найдем вектор возмущающих сил

$$\bar{\mathbf{N}}_j = \begin{bmatrix} \bar{\mathbf{N}}^a_{j_1} \\ \bar{\mathbf{N}}^b_{j_2} \\ \bar{\mathbf{N}}^c_{j_3} \\ \vdots \\ \bar{\mathbf{N}}^y_{j_{l-1}} \end{bmatrix} = \mathbf{K}_{jj}\mathbf{S}\bar{\mathbf{z}}, \tag{1.164}$$

где

$$\mathbf{K}_{jj}=$$

$$=\begin{bmatrix} \mathcal{F}^a_{Vj_1,j_1}+\mathcal{F}^b_{Vj_1,j_1} & -\mathcal{F}^b_{Vj_1,j_2} & \dots & 0 \\ -\mathcal{F}^b_{Vj_2,j_1} & \mathcal{F}^b_{Vj_2,j_2}+\mathcal{F}^c_{Vj_2,j_2} & \dots & 0 \\ 0 & -\mathcal{F}^c_{Vj_3,j_2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \mathcal{F}^y_{Vj_{l-1},j_{l-1}}+\mathcal{F}^z_{Vj_{l-1},j_{l-1}} \end{bmatrix}; \tag{1.165}$$

$$\mathbf{S}=\begin{bmatrix} -\rho^a_{Uj_1} & \rho^b_{Uj_1} & 0 & \dots & 0 & 0 \\ 0 & -\rho^b_{Uj_2} & \rho^c_{Uj_2} & \dots & 0 & 0 \\ 0 & 0 & -\rho^c_{Uj_3} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -\rho^y_{Uj_{l-1}} & \rho^z_{Uj_{l-1}} \end{bmatrix}; \tag{1.166}$$

$$\mathbf{z}=\{\bar{z}_{Ua}\bar{z}_{Ub}\dots\bar{z}_{Uy}\bar{z}_{Uz}\}. \tag{1.167}$$

Уравнение (1.150) запишем для всех суперэлементов. После некоторых преобразований представим эти уравнения в матричной форме

$$[(\Lambda^{-1}+\mathbf{S}^{\mathrm{T}}\mathbf{K}_{jj}S)-p^2\mathbf{E}]\bar{\mathbf{z}}=0, \tag{1.168}$$

где

$$\Lambda^{-1}=[\Lambda^{-1}_{Ua}\Lambda^{-1}_{Ub}\dots\Lambda^{-1}_{Uz}]. \tag{1.169}$$

Для перемещений конструкции в целом получаем

$$\bar{\mathbf{q}}=(\mathbf{\Phi}+\mathcal{F}\mathbf{K}_{jj}\mathbf{S})\bar{\mathbf{z}}, \tag{1.170}$$

где

$$\bar{\mathbf{q}}=\{\bar{q}_{ia}\bar{q}_{j1a}\bar{q}_{ib}\bar{q}_{j2b}\bar{q}_{ic}\bar{q}_{j2c}\dots\bar{q}_{j_{l-1}y}\bar{q}_{iz}\}; \tag{1.171}$$

$$\mathbf{\Phi}=\begin{bmatrix} \rho^a_{Ui} & 0 & \dots & 0 \\ \rho^a_{Uj1} & 0 & \dots & 0 \\ 0 & \rho^b_{Ui} & \dots & 0 \\ 0 & \rho_{Uj2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \rho^z_{Ui} \end{bmatrix}; \tag{1.172}$$

$$\mathbf{F}=\begin{bmatrix} \mathcal{F}^a_{Vi,j_1} & 0 & 0 & \dots & 0 & 0 \\ \mathcal{F}^a_{Vj_1,j_1} & 0 & 0 & \dots & 0 & 0 \\ -\mathcal{F}^b_{Vj,j_1} & \mathcal{F}^b_{Vi,j_2} & 0 & \dots & 0 & 0 \\ -\mathcal{F}^b_{Vj_2,j_1} & \mathcal{F}_{Vj_2,j_2} & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & -\mathcal{F}^y_{V_i,j_{l-2}} & \mathcal{F}^y_{Vi,j_{l-1}} \\ 0 & 0 & 0 & \dots & -\mathcal{F}^y_{Vj_{l-1},j_{l-2}} & \mathcal{F}^y_{Vi_{l-1},j_{l-1}} \\ 0 & 0 & 0 & \dots & 0 & -\mathcal{F}^z_{Vi,j_{l-1}} \end{bmatrix}. \tag{1.173}$$

Окончательные значения форм колебаний конструкции определяются нормированием столбцов матрицы перемещений

$$\rho = \|(\Phi + \mathcal{F} K_{jj} S)\bar{z}\| . \qquad (1.174)$$

1.5. О МАТРИЧНЫХ АЛГОРИТМАХ, ИСПОЛЬЗУЕМЫХ В ИНЖЕНЕРНЫХ РАСЧЕТАХ

В настоящее время матричные представления стали одним из основных способов формулировки научно-технических задач и одновременно методом приведения их к форме, пригодной для решения на ЭВМ.

Назовем лишь здесь некоторые типы задач, для решения которых часто используются матричные представления и алгоритмы.

1. Разнообразные задачи механики твердого тела, строительной механики, вычислительной гидромеханики, небесной механики.

2. Физические задачи дифракции и рассеяния частиц, определения колебательных спектров молекул, физики плазмы, теплофизики.

3. Задачи математической статистики, регрессионный анализ, обработка экспериментальных данных.

4. Задачи вычислительной, в частности, квантовой химии.

5. Расчет электро- и радиосхем (на основе законов Кирхгофа), электронных фильтров, антенных устройств, электромагнитных полей приборов.

6. Расчет тепло- и массопереноса, определение тепловых полей элементов конструкций и т. д.

Матричные обозначения придают формулировкам задач компактный вид, обозримость, вследствие чего упрощается и унифицируется программная реализация, так как язык матричной алгебры идеально приспособлен к расчетам на ЭВМ.

Несмотря на принадлежность названных задач к различным предметным областям и различия в принятых способах описания явлений и терминологии, методы решения их на ЭВМ близки. Необходимыми обычно являются алгоритмы формирования матриц специального вида (желательно с учетом их свойств и структуры, в частности, блочности, симметрии, разреженности), объединения, расчленения, перестроения матриц и выполнения ряда алгебраических операций, в частности решения систем линейных уравнений, определения собственных значений и векторов.

Общность требований к выполняемым матричным процедурам дает возможность построения единых комплексов программ для решения разнообразных технических задач, представленных в матричной форме. Система матричного программного обеспечения (СМПО), представленная в настоящей книге, является примером такого комплекса программ.

Наличие средств выполнения операций над матрицами произвольного размера и структуры, а также простота пополнения СМПО новыми программами, необходимыми для учета специфики разнообразных алгоритмов, дает возможность применять его к решению широкого спектра научно-технических задач.

В гл. 10 помещены примеры применения СМПО к решению задач строительной механики вантовых систем и гидромеханики, представленных в матричной форме.

1.6. ОСНОВНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ И МАТРИЧНЫХ АЛГОРИТМОВ

Проблема обеспечения наибольшего удобства для пользователей, решающих различные научно-технические задачи, записанные в матричной форме, надежности и эффективности работы, определяет ряд требований к прикладному программному обеспечению, наиболее важными из которых являются:

1) модульность. Модульная структура позволяет проводить модификации и добавления, облегчает сопровождение;

2) иерархическая структура;

3) программная или аппаратная реализация механизма виртуальной памяти;

4) надежная и эффективная реализация основных матричных операций (перестроение, поэлементные операции, умножение, решение систем линейных алгебраических уравнений с учетом их структуры и разреженности, решение полной и частичной проблемы собственных значений);

5) высокое качество документации;

6) открытость, позволяющая интегрировать математическое обеспечение с другими программными средствами.

Реализация алгоритмов МКЭ налагает на программное обеспечение дополнительные требования. Должны быть обеспечены:

1) наличие нескольких подсистем, основанных на различных вариационных принципах (методе сил, методе перемещений);

2) возможность применения суперэлементного метода;

3) высокое качество и разнообразие используемых конечных элементов;

4) эффективность формирования матрицы жесткости и матрицы податливости (наличие гибких схем управления памятью);

5) развитые подсистемы ввода-вывода и контроля конечно-элементной информации (быстрота и качество подготовки исходных данных, графическая визуализация данных и результатов, использование справочных библиотек, диагностирование алгоритмических отказов с указанием причины ошибок и способа их исправления).

Описываемый в книге пакет прикладных программ удовлетворяет всем перечисленным требованиям.

Глава 2. Архитектура программного обеспечения МКЭ и матричных алгоритмов

К архитектуре программного обеспечения относятся: организация общей структуры системы и связей между ее отдельными компонентами; создание, хранение и использование информации; сопряжение программного обеспечения с различными аппаратурными средствами ЭВМ.

2.1. СТРУКТУРА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ МАТРИЧНЫХ АЛГОРИТМОВ

Основой для создания программного обеспечения для вычислительной реализации матричных алгоритмов является схожесть и простота матричных операций.

В данном разделе рассмотрим вопрос структуры математического обеспечения общего назначения, не затрагивая метод конечных элементов, который также формулируется в терминах матричной алгебры, но имеет еще и свои особенности.

Структура программного матричного обеспечения определяется предъявляемыми к нему требованиями, которые были перечислены выше.

По-видимому, в соответствии с этими требованиями должно быть создано программное обеспечение или пакет прикладных программ (ППП) сложной структуры. Основными компонентами ППП являются: входной язык пакета, монитор, библиотека стандартных программных единиц (СПЕ), библиотека процедур для связи ППП с операционной системой. ППП назовем процедурно-ориентированным и будем считать, что он является самостоятельным при решении различных задач, записанных в матричной форме, и в то же время базовым при создании и функционировании программных комплексов, ориентированных на решение различных научно-технических задач (например, задач прочности авиаконструкций, машин, сооружений и т. п.), за счет присоединения к нему проблемно-ориентированной программной части.

В целях экономии оперативной памяти целесообразно применять принцип интерпретации, в соответствии с которым вся информация, необходимая для осуществления конкретной матричной операции, находится в оперативной памяти вычислительной машины только во время выполнения этой операции. Поэтому монитор пакета осуществляет действия, связанные с реализацией этого основного принципа.

Информационной единицей монитора является матрица. В данном случае целесообразно ввести следующее определение [35]: блочной матрицей считается структура данных, состоящая из совокупности обычных матриц, рассматриваемых в качестве блоков, и метаматрицы, содержащей информацию о размерах и местонахождении матриц.

Входной язык матричного программного обеспечения представляет собой набор операторов. Удобное размещение матриц достигается следующим образом: с целью сокращения числа операций ввода-вывода, осуществляемых монитором с рабочим файлом (РФ), ранее определенные матрицы запоминают в области данных; матрицы размещают в управляемой и автоматической памяти (управляемая память дает возможность программисту участвовать в распределении оперативной памяти при выполнении программы, а автоматическая память полностью изолирует его от управления памятью); средствами реализации рассматриваемых видов динамического распределения являются структуры данных в виде стека и очереди. ППП позволяет включать в него отдельные подпрограммы и целые модули, написанные на языках высокого уровня (например, фортране), в целях расширения круга пользователей, а также использования готовых разработок.

2.2. СТРУКТУРА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ РЕШЕНИЯ ЗАДАЧ МКЭ

Современный ППП, используемый в системе автоматизированного проектирования (САПР), должен включать в себя различные варианты МКЭ и иметь автоматизированные подсистемы подготовки данных (препроцессор) и обработки информации и визуализации (постпроцессор).

На основе опыта следует, по-видимому, рекомендовать следующую структуру математического обеспечения для решения задач МКЭ. Программный комплекс состоит из процедурно-ориентированной и проблемно-ориентированной частей. Процедурно-ориентированная часть носит общецелевой независимый характер, выполняет функции матричного программного обеспечения и не имеет явных признаков метода конечных элементов; проблемно-ориентированная часть ППП полностью определяется характером научно-технической проблемы и базируется на первой.

Проблемно-ориентированная часть комплекса подразделяется на три компоненты: препроцессор, процессор, постпроцессор, каждая из которых представляется набором подсистем. Хотя основной вычислительной компонентой является процессор, препроцессорные части пакета должны быть автоматизированы, чтобы подготовка данных (с контролем и диагностикой) и обработка полученных результатов и их визуализация по производительности были адекватны процессорной части ППП. Предполагаемая привязка проблемно-ориентированной части ППП к процедурно-ориентированной позволит решить ряд важных задач автоматизации расчетов, например, поручить функции базы данных процедурно-ориентированной части ППП. В этом случае отпадает необходимость использования таких универсальных и не всегда эффективных средств, как системы управления базами данных (СУБД). Важно также предусмотреть модульный принцип построения ППП, предполагающий создание многоуровневых программ, что в сочетании с методами

генерации данных позволит писать программы с использованием всего лишь нескольких операторов языка высокого уровня.

Для выживаемости комплекс должен быть открытым и иметь возможность «воспринимать» модули на языках высокого уровня. Удовлетворение требований, указанных в разд. 1.6, упрощается при наличии процедурно-ориентированной части комплекса, какой в данном случае является ППП СМПО.

2.3. ЭВОЛЮЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МКЭ

Учитывая историю развития методов, можно считать, что программное обеспечение МКЭ опирается на программное обеспечение матричных алгоритмов.

Поскольку первые языки высокого уровня (алгол-60, фортран, кобол и др.) не были ориентированы непосредственно на матричные вычисления, то шел поиск новых языков; одним из первых языков нового поколения является язык «альфа-системы автоматизации программирования». Однако альфа-система не позволяла выполнять операции над блочными матрицами, так как они в альфа-системе носят лишь вспомогательный характер [35].

С учетом этого было создано матричное программное обеспечение для ЭВМ второго поколения [10, 12]. Для этого был применен один из наиболее доступных методов автоматизации программирования с использованием специализированной библиотеки стандартных программ в системах ИС-2 и ССП-2 [35]. Стандартные программы позволили запрограммировать матричные алгоритмы, не прибегая к ручному программированию в кодах машины, так как программист, по существу имеет дело с новой ЭВМ, команды которой расширены за счет операций, выполняемых по библиотечным программам. Впервые была предложена метаматрица, которая представляет собой условную матрицу, содержащую информацию о размерах и структуре реальных блочных матриц, хранящихся на внешних носителях; этим решались основные проблемы работы с матрицами.

В дальнейшем описанное выше матричное программное обеспечение совершенствовалось в направлении повышения уровня автоматизации (была создана матричная административная система, которая обеспечивала динамическое распределение внешней памяти, работу с данными, способствовала отладке программ и т. п.). Эта идеология в значительной мере сохранилась в программном обеспечении машин третьего поколения (ЕС ЭВМ) [35].

Следует отметить матричный язык ПМЗ (MATLAN), применяемый для ЕС ЭВМ. Пакет написан в рамках операционной системы ОС/ЕС с использованием языков ассемблер (управляющая часть, транслятор) и фортран (программные модули); о его особенностях будет сказано ниже.

Уровень разработок программного матричного обеспечения в значительной мере определил состояние разработок программных комплексов для метода конечных элементов.

Программное обеспечение COSAR используют для прочностных расчетов методом конечных элементов (в варианте метода перемещений) при решении задач механики твердого деформируемого тела [25]. Система COSAR является открытой. Хотя подпрограммы системы преимущественно выполнены на языках высокого уровня, в ней предусмотрены возможности использования в будущем новых технических средств и математического обеспечения (например, матричных модулей, виртуальной памяти, параллельной обработки). Для системы разработана специализированная подсистема управления данными FEDAM, приспособленная к особенностям МКЭ и операций над матрицами высокого порядка; в ее функции входят динамическое управление оперативной и внешней памятью, организация и проведение автоматического обмена данными между оперативной и внешней памятью; сегментирование больших матриц на субматрицы. В каждой гиперматрице имеется матрица отображения, в которой описаны адреса всех субматриц на внешнем запоминающем устройстве. Для больших субматриц, требующих большого объема памяти, может также потребоваться сегментирование с построением соответствующих матриц отображения; таким способом могут быть обработаны матрицы любой величины. В системе (с учетом того, что большая часть счетного времени расходуется на выполнение относительно коротких частей программ) предусматривается дополнительная оптимизация этих частей с соответствующей заменой их на стандартные программы на языке ассемблер. Система COSAR в дальнейшем была существенно расширена. При разработке первого варианта системы ориентировались на ЭВМ БЭСМ-6, которая по функциональной структуре соответствует вычислительным машинам третьего поколения.

В системах NASTRAN, FORMAT, COSAR имеются подсистемы для выполнения матричных операций, которые могут быть использованы не только для расчета конструкций, например, матричная подсистема DMAP (из пакета NASTRAN) применяется для численной реализации матричных алгоритмов из других областей механики.

Таким образом, при создании вышеназванных программных комплексов уделено определенное внимание вопросу выбора средств программирования для численной реализации матричных алгоритмов как основы конечно-элементных расчетов (особенно при ориентации на различные варианты МКЭ).

Достаточно близкой по архитектуре к программному обеспечению МКЭ, описанному в данной книге, следует считать систему COSAR, которая имеет следующие характеристики. Входящая в COSAR подсистема FEDAM обеспечивает размещение и организацию данных. Сегментация данных — постраничная; информация организована в оперативной памяти (ОП) по схеме очереди (первой освобождается наиболее долго находящаяся в ОП страница), в наборе прямого доступа — по схеме кучи. При выборе размера страницы возникает противоречие между характеристиками устройства прямого доступа и характером данных.

В отличие от подсистемы FEDAM в СМПО применяется поблочная сегментация, причем размер блока не является постоянным. Большие матрицы представляются в форме гиперматриц, организация которых близка к принятой в СМПО концепции блочных матриц. Однако субматрицы FEDAM в отличие от блоков СМПО имеют одинаковые размеры, они, как правило, квадратные, что, разумеется, не всегда удобно. Кроме того, подсистема FEDAM системы COSAR обеспечивает только манипуляцию данными. Функции СМПО значительно шире и, кроме управления данными, заключаются в выполнении разнообразных матричных операций.

2.4. СРАВНИТЕЛЬНЫЙ АНАЛИЗ СУЩЕСТВУЮЩИХ СРЕДСТВ ПРОГРАММНОЙ РЕАЛИЗАЦИИ МАТРИЧНЫХ АЛГОРИТМОВ И МКЭ

В настоящее время можно говорить о трех поколениях прикладных программ: одиночные программы, ППП с простой структурой, ППП со сложной структурой. Пакеты с простой структурой представляют собой наборы функционально совместимых между собой программ, расширяющих библиотеку стандартных подпрограмм того языка программирования, на котором они написаны, например, пакеты научных подпрограмм на языке фортран и ПЛ/1.

Использование подпрограмм из этих пакетов в проблемно-ориентированном математическом обеспечении упрощает реализацию только отдельных матричных операций. Организация же данных и вычислительного процесса зависит полностью от пользователя и возможностей языка, выбранного для решения задачи.

Можно говорить о том, что структура и эффективность проблемно-ориентированных ППП зависят и определяются базовыми процедурно-ориентированными программными средствами. С увеличением сложности решаемых задач становится очевидным, что использование универсальных средств и ППП с простой структурой может привести к неоптимальному использованию ресурсов вычислительной машины, к значительному усложнению процессов программирования и решения задач. В такой ситуации необходима разработка и использование ППП со сложной структурой. Следует отметить, что принципы построения таких пакетов во многом аналогичны принципам построения языков программирования высокого уровня [2].

Пакет со сложной структурой включает в себя следующие компоненты: управляющую программу (монитор); транслятор с входного языка; библиотеку модулей пакета; сервисные программы; средства генерации и адаптации пакета; системные файлы пакета [2]. Монитор работает под непосредственным контролем операционной системы и управляет общим ходом вычислительного процесса в пакете. Транслятор с входного языка пакета обрабатывает исходный модуль и строит вычислительную схему процесса решения задачи. В качестве транслятора могут использоваться макропроцессоры базового языка программирования [7]. В этом случае пакет должен содержать достаточно полный набор макрооперато-

ров. Библиотека пакета содержит модули, предназначенные для решения задач предметной области. Сервисные программы выполняют функции, связанные с действиями монитора, подготовкой исходных данных и обработкой результатов работы пакета. Средства генерации и адаптации пакета позволяют наилучшим образом приспособить пакет к конкретной операционной системе и конфигурации машины. Системные файлы пакета содержат библиотеки пакета и данные, необходимые для вычислительного процесса.

К пакетам со сложной структурой относится ПМЗ (MATLAN) — ППП для решения матричных задач [47]. Этот пакет написан в рамках операционной системы ОС/ЕС с использованием языков ассемблер (управляющая часть, транслятор) и фортран (программные модули). Пакет позволяет обрабатывать матрицы различной структуры и размерности. Однако при решении больших задач, характерной особенностью которых является наличие большого числа матриц и обрабатывающих их операторов, выявляется ряд существенных недостатков ПМЗ, главными из которых являются: большой объем необходимой оперативной памяти и внешней памяти на магнитных дисках; невысокая скорость выполнения матричных операций; ограниченный набор операторов.

Так, например, на ЭВМ ЕС-1033 СМПО выполнила умножение матриц **A** [100×200] и **B** [200×100] за 1 мин 37 с, ПМЗ — за 4 мин 2 с, т. е. в 2,5 раза медленнее (размеры рабочих областей были по 70 Кбайт), причем основное время в ПМЗ было потрачено на операции обмена, что говорит о нерациональной организации вычислительного процесса, в который пользователь ПМЗ не имеет возможности вмешаться.

Однако самый большой недостаток ПМЗ — невозможность пополнения так называемых сегментных алгоритмов, т. е. алгоритмов, обрабатывающих большие матрицы, не помещающиеся целиком в ОП. Если набор алгоритмов для обработки малых матриц можно пополнять, используя для этого фортран, то при обработке больших матриц необходимость считывания их отдельными кусками без знания структуры хранения матриц может привести к катастрофическому росту времени выполнения. Мало этого. Если программа пользователя достаточно велика и требуется применение оверлейной структуры (в ПМЗ нет динамической связи программ), то невозможно непосредственно передать матрицу с выхода подпрограммы ПМЗ на вход следующей: обязателен предварительный вывод ее на диск. В составе ПМЗ отсутствует, например, такая важная матричная операция, как наложение малой матрицы со сложением в большую. Эта операция является основной при формировании матрицы жесткости в методе перемещений. Для ее реализации требуется сначала выбрать подматрицу из большой матрицы, затем выполнить сложение и записать результат. При этом затраты на ввод-вывод почти удваиваются. В ПМЗ отсутствуют также современные методы решения собственной проблемы и систем линейных алгебраических уравнений.

Резюмируя вышесказанное, можно сделать вывод, что ПМЗ может быть применен только для малых задач, которые можно решать средствами, имеющимися в ПМЗ.

2.5. ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ СМПО — СРЕДСТВО ДЛЯ РЕАЛИЗАЦИИ МАТРИЧНЫХ АЛГОРИТМОВ МКЭ

СМПО — это пакет со сложной структурой. Он включает в себя следующие компоненты: программу управления данными (монитор пакета), транслятор с входного языка, библиотеку модулей пакета, средства генерации и адаптации пакета (рис. 2.1). Эти компоненты ППП тесно связаны с системой наборов данных пакета. Пакет базируется на операционной системе ОС ЕС. Имеется также вариант пакета, функционирующий в среде ДОС ЕС; оба варианта пакета совместимы на уровне входного языка. Монитор ППП СМПО работает под непосредственным контролем операционной системы и управляет потоками информации и ходом вычислительного процесса в пакете. Транслятор с входного языка обрабатывает исходный модуль и строит вычислительную схему процесса решения задачи. Транслятором в ППП СМПО служит Макроассемблер ЕС ЭВМ; в процессе трансляции активно используется макробиблиотека пакета, содержащая макроопределения операторов пакета. Библиотека модулей пакета содержит готовые к выполнению загрузочные модули, каждый из которых решает соответствующую задачу линейной алгебры для матрицы некоторой структуры и разреженности. Сервисные программы выполняют служебные функции, связанные с действиями монитора, подготовкой исходных данных и обработкой результатов работы пакета. Средства генерации и адаптации ППП СМПО, в качестве которых выступает библиотека процедур пакета, позволяют в короткие сроки наилучшим образом приспособить пакет к конкретной операционной системе и конфигурации ЭВМ.

Для долговременного хранения информации в ППП СМПО используются наборы данных на магнитных лентах и дисках. Информация хранится на дисках в виде библиотечного набора, на лентах — в специальном формате. ППП СМПО при операциях с матричными данными обеспечивает важнейшие специфические функции базы данных [24]: доступ к матрицам осуществляется по их именам, избыточность информации минимальна, плотность хранимой информации велика, переход от одного типа носителя к другому несложен.

В основе функционирования пакета лежит принцип интерпретации, в соответствии с которым матрицы и программы размещаются в ОП только на время выполнения операции, причем все операции ввода-вывода выполняются автоматически. Динамическое распределение оперативной и внешней памяти, осуществляемое пакетом, позволяет обрабатывать большое число матриц, размеры которых могут во много раз превышать выделенный под задачу объем оперативной памяти. Матрицы большой размерности пред-

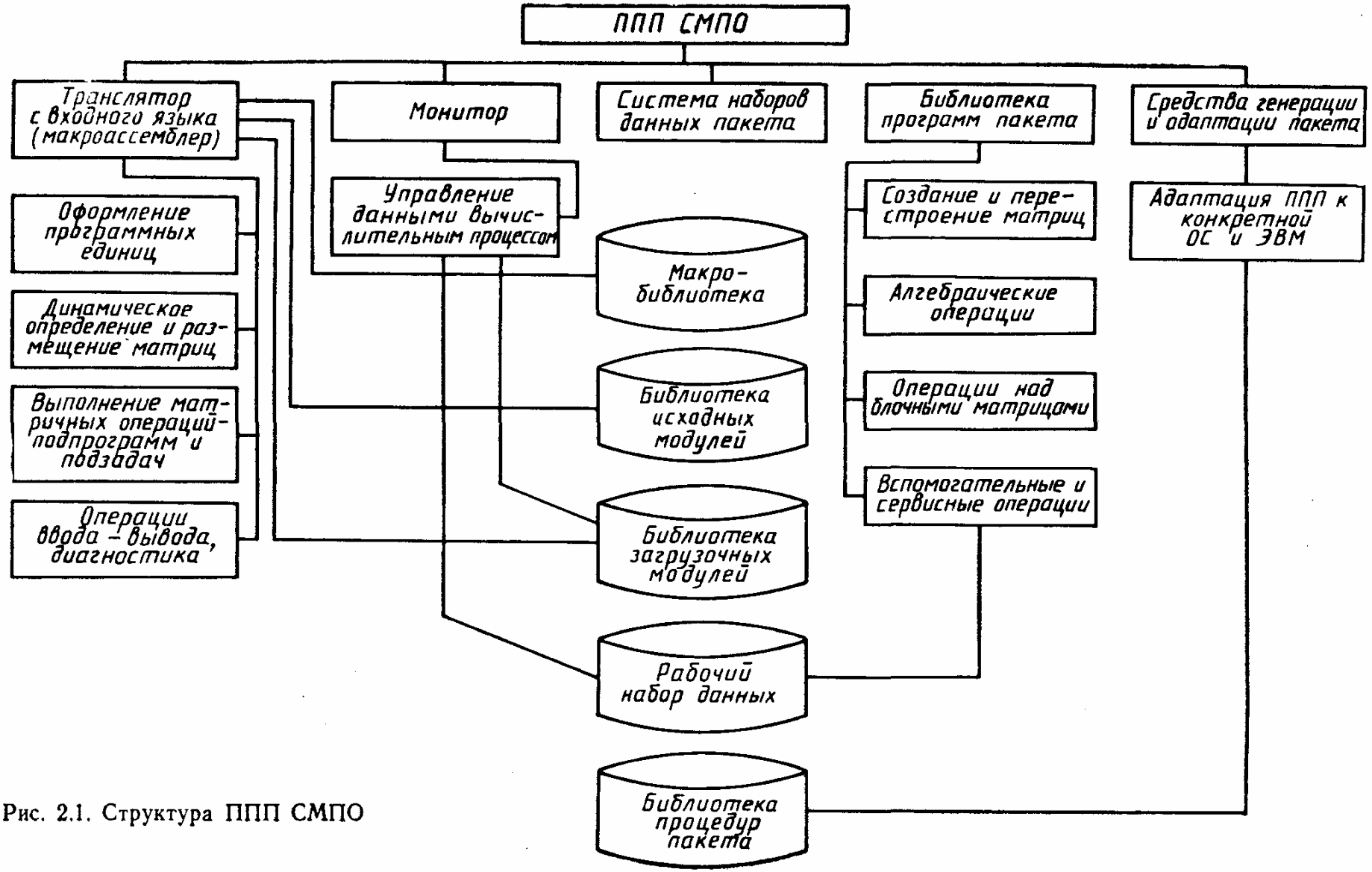


Рис. 2.1. Структура ППП СМПО

ставляют в СМПО в универсальном блочном формате. Многие традиционные алгоритмы линейной алгебры применимы к таким матрицам, так как элементами их являются подматрицы-блоки. Это существенно облегчает разработку программ и дает возможность просто учитывать разреженность матриц на уровне блоков.

Для повышения производительности пакета в дополнение к принципу интерпретации, не требующему от пользователя пакета никаких усилий по размещению матриц в ОП, но не всегда эффективному из-за большого числа операций обмена, у пользователя имеется возможность применения принципа управляемого или автоматического распределения ОП [2]. В управляемом режиме пользователь явно указывает те матрицы, которые размещаются в ОП, и сам указывает на необходимость освобождения памяти. В автоматическом режиме блоки блочных матриц сохраняются в ОП, насколько это возможно; их замена происходит автоматически, если для размещения требуемых для выполнения операции блоков недостаточно места.

Возможна комбинация всех трех принципов распределения памяти.

Процедурно-ориентированный входной язык пакета является расширением ассемблера и позволяет создавать хорошо структурированные программы. Программы пакета также написаны на этом языке.

Операторы пакета представляют собой макрокоманды. Благодаря этому удалось достигнуть высокой производительности программ и заметной экономии ОП на их размерах: даже лучшие оптимизирующие компиляторы фортрана с неплохой производительностью полученного кода сильно проигрывают по требуемой ими памяти (не считая модулей, подключаемых при редактировании).

Для машин среднего класса (ЕС-1022, ЕС-1033) экономия памяти на размерах программ очень существенна; для высокопроизводительных ЭВМ (ЕС-1045, ЕС-1060), особенно при наличии виртуальной памяти, на первый план выступает экономия времени выполнения. Это связано с тем, что даже в лучшем случае фортран-циклы содержат на несколько команд больше, чем ассемблер-циклы. Если команды с плавающей точкой (особенно умножение) выполняются гораздо дольше других команд, что характерно для машин среднего класса, то несколько лишних команд, связанных с настройкой адресов, не оказывают заметного влияния на временные затраты; если же разница во времени выполнения различных команд невелика, что является отличительной чертой высокопроизводительных ЭВМ, то эти дополнительные команды уже нельзя игнорировать. Следствием этого является то обстоятельство, что при переходе с ЭВМ ЕС-1033 на ЭВМ ЕС-1045 время выполнения фортран-программы уменьшается в 3—3,5 раза, а выполнения СМПО-программ — в 4,5—7 раз.

В пакете СМПО предусмотрена возможность подсоединения фортран-программ.

Глава 3. Организация данных в ППП СМПО

В большинстве языковых средств программирования предусмотрена возможность непосредственного определения и обработки лишь ограниченного набора типов данных. К ним чаще всего относятся целые и вещественные числа, символьные и логические переменные. Из типов данных могут организовываться структуры данных: строки, массивы, таблицы, деревья. Однако ограничения на размеры структур и способы обращения к ним делают целесообразным их использование только при условии полного размещения нужной структуры в основной памяти ЭВМ. Даже наличие аппарата виртуальной памяти при игнорировании особенностей решаемых задач может привести к необоснованным издержкам и не позволит решить сложную задачу в условиях ограниченных возможностей ЭВМ. При решении задач, связанных в основном с матричными операциями, желательно иметь возможность обращения к матрице только по ее идентификатору, не задумываясь над проблемами ее представления и размещения в памяти ЭВМ.

3.1. ТИПЫ ДАННЫХ ППП СМПО

Базовым языком программирования в ППП СМПО является ассемблер ЕС ЭВМ [16, 28]. Кроме этого для написания отдельных подпрограмм может быть использован фортран.

Основными собственными типами (структурами) данных в ППП СМПО являются матрицы и списки.

Матрица — двумерный массив, элементы которого располагаются в памяти ЭВМ по строкам и идентифицируются одним именем.

Идентификатор матрицы по правилам ассемблера представляет собой последовательность не более восьми буквенно-цифровых символов, начинающуюся с буквы. Элементами матрицы являются вещественные числа с плавающей точкой двойной точности, которые в ассемблере имеют формат D. В начале матрицы, перед ее первым элементом, располагаются размеры — число строк и число столбцов матрицы. Каждый размер является натуральным числом в формате полуслова H. Таким образом, матрица A размером $m \times n$ занимает в памяти объем, равный $(8mn+4)$ байта. Адрес элемента a_{ij} относительно начала матрицы равен $8[n(i-1)+j]-4$.

Размещение размеров матрицы вместе с ее элементами позволяет освободить пользователя от необходимости указывать размеры при выполнении матричных операций и сократить число параметров при обращении к подпрограммам. Для сравнения укажем, что при использовании пакета научных подпрограмм на языке фортран приходится указывать фактические размеры используемых матриц [16]. Это усложняет программирование и является источником возможных ошибок.

Размещение матрицы в ППП СМПО может быть статическим и динамическим. Статически размещенная матрица полностью задается непосредственно в тексте программы с помощью оператора

SMATR. Этот оператор резервирует поле памяти, идентифицирует элементы матрицы и устанавливает их значение.

Поскольку статические матрицы располагаются в тексте программы, их размеры не могут быть большими и должны задаваться до этапа трансляции программы.

Динамически размещенная матрица создается в процессе выполнения программы с помощью операторов OPR, MATR, ZAPOM. Поле с элементами матрицы не размещается в самой программе. Программа содержит только дескриптор (описатель) матрицы.

Дескриптор является восьмибайтовым поименованным полем. Имена матрицы и ее дескриптора совпадают. Поле дескриптора заполняется при создании матрицы и содержит следующую информацию:

A ‹ПМ, M1, N, M2, ПС, D›

где A — имя матрицы размером M×N; ПМ — признак матрицы (1 байт); M1 — младший байт первого размера (1 байт); N — второй размер матрицы (2 байта); M2 — старшие полбайта первого размера (0,5 байта); ПС — признак состояния матрицы (0,5 байта); D — адрес расположения матрицы (3 байта).

Монитор пакета распознает дескриптор матрицы по ПМ, который всегда равен шестнадцатеричной константе X'80'. Первый размер матрицы — число строк — равен M2M1 и не может превышать числа 4095. Второй размер — число столбцов N — не превышает числа 32767. Признак состояния (ПС) указывает на то, где в данный момент находится матрица. Адрес расположения матрицы D может быть в зависимости от ПС либо дисковым адресом, либо адресом основной памяти (ОП). Сама матрица располагается в памяти обычным образом. После уничтожения динамически размещенной матрицы ПС и D становятся равными нулю.

Кроме матриц, содержащих действительные числа, в ППП СМПО имеются специальные структуры данных, которые называются списками (шкалами) и предназначены для хранения целочисленной информации.

Списки могут быть по своей структуре простыми или сложными, а по размещению — статическими или динамическими.

Простые и сложные статические списки задаются с помощью операторов ADSP, CSP. Элементы простого списка S, представляющего собой вектор с N компонентами, располагаются в ОП последовательно. Перед первым элементом списка (S_1) указан его размер — N. Как размер, так и сами элементы S_I являются целыми числами в формате полуслова H:

S ‹N, S_1, S_2,..., S_N›

Сложные статические списки имеют заголовок, который содержит адреса простых подсписков в формате A. Заголовок заканчивается нулевым адресом. Имя списка присваивается полю, содержащему адрес заголовка (рис. 3.1).

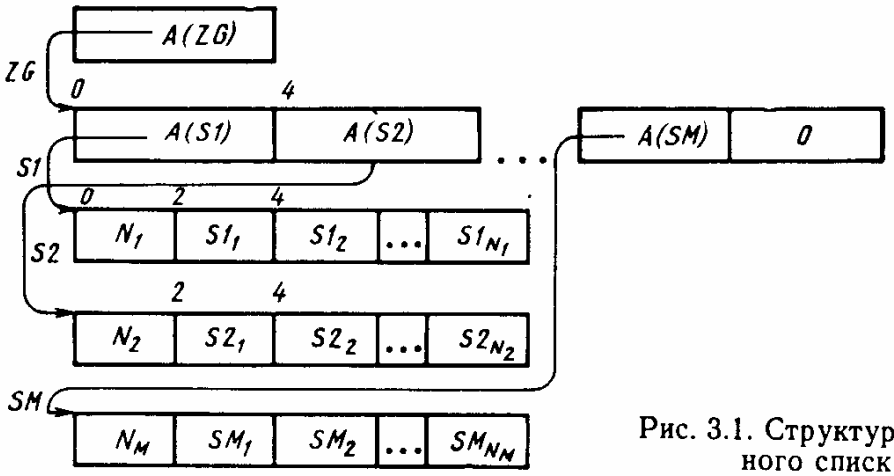


Рис. 3.1. Структура сложного списка

Если простой список упаковать в матрицу, получится простой динамический список. Он имеет следующий вид:

$S\langle M, N, K, L, S_1, S_2, \ldots, S_{K\times L}\rangle$

Здесь $K\times L$ — размер списка, т. е. число его элементов; M и N — размеры матрицы, в которую упакован список.

Поскольку элемент матрицы занимает 8 байтов, а элемент списка — 2 байта, должно выполняться неравенство $4MN \geqslant KL+4$, где дополнительные 4 байта требуются для размещения чисел K и L; в остальном величины M и N произвольны.

Как и всякая матрица в ППП СМПО, матрица, содержащая список, имеет дескриптор обычного формата и может располагаться на магнитном диске или в ОП.

Простые и сложные динамические списки создаются путем ввода элементов списков операторами VVS и VVSS соответственно. Сложные динамические списки после создания имеют такую же структуру, как и сложные статические списки.

Некоторые программы ППП СМПО используют так называемые смешанные статические списки, имеющие следующую структуру:

S $\langle A(S1)\rangle$

S1 $\langle K, L, S1_1, S1_2, \ldots, S1_{KL}\rangle$

Здесь A(S1) — адрес подсписка S1.

Поскольку специальных средств для создания таких списков в ППП СМПО нет, их можно формировать операторами языка ассемблер.

Рассмотрим пример кодирования смешанного статического списка. Пусть нужно сформировать список размерами 2×4, содержащий элементы 1, 3, 3, 4, 4, 7, 8, 3:

```
S     DC    A(S1)
S1    DC    H'2, 4', H'1, 3, 3, 4, 4, 7, 8, 3'
```

или короче:

```
S     DC    A(*+4), H' 2, 4, 1, 3, 3, 4, 4, 7, 8, 3'
```

Здесь константа A(*+4) содержит адрес информации, расположенной непосредственно за ней самой, т. е. необходимый в соответствии со структурой адрес подшкалы S1.

3.2. ОБЫЧНЫЕ И БЛОЧНЫЕ МАТРИЦЫ

Матрицы, рассмотренные в разд. 3.1, являются обычными матрицами. Если размер матрицы превышает выделенный объем ОП, ее обработка в виде обычной матрицы становится невозможной. В этом случае следует осуществлять переход к блочной матрице.

Блочной матрицей называется структура данных, состоящая из совокупности обычных матриц, рассматриваемых в качестве блоков, и метаматрицы, содержащей информацию о размерах и местонахождении блоков. Размеры блоков и метаматрицы ограничены величиной ОП, а размеры всей блочной матрицы зависят в основном от емкости внешних накопителей на магнитных дисках, где хранятся блоки и метаматрицы. При обработке блочной матрицы в ОП постоянно находится только ее метаматрица. Блоки вызываются в ОП по мере необходимости.

Все подпрограммы, выполняющие алгебраические и логические операции над блочными матрицами, написаны с учетом их структуры. Операции над нулевыми блоками не выполняются. Для симметричной матрицы возможно получение и хранение только одной из ее симметричных половин.

Выбор размера блока зависит от специфики задачи. Иногда такое разбиение имеет физический смысл (например, в конечно-элементном методе сил каждый блок матрицы податливости необъединенных элементов соответствует элементу конструкции [35]), в других случаях физический смысл найти не удается. Размер блока в этом случае определяют исходя из компромисса между стремлением к экономии памяти при уменьшении размера блока благодаря тому, что нулевые блоки в ней не хранятся, и стремлением к уменьшению временны́х затрат на ввод-вывод при укрупнении блоков. При этом надо помнить, что чрезмерное уменьшение размера блока невыгодно также из-за роста метаматрицы, а чрезмерное увеличение — из-за увеличения времени на обработку нулевых элементов внутри блоков. Практика показывает, что в тех случаях, когда основными по временны́м затратам являются операции умножения и решения систем уравнений, для машин невысокой производительности (ЕС-1022, ЕС-1033) эффективен размер блока (50×50)... (60×60), а для высокопроизводительных ЭВМ (ЕС-1045, ЕС-1060) — (60×60)...(90×90).

При обработке блочных матриц пользователь ППП СМПО должен обеспечить соответствие их структур. Например, при сложении блочных матриц метаматрицы слагаемых и метаматрица суммы должны иметь одинаковые размеры. Блоки с одинаковыми координатами должны иметь одинаковые размеры. Число суперстрок в матрице блочной системы уравнений должно равняться числу суперстрок в правой части, а суперстроки в этих двух матрицах, находящиеся на одном уровне, должны иметь одинаковое число

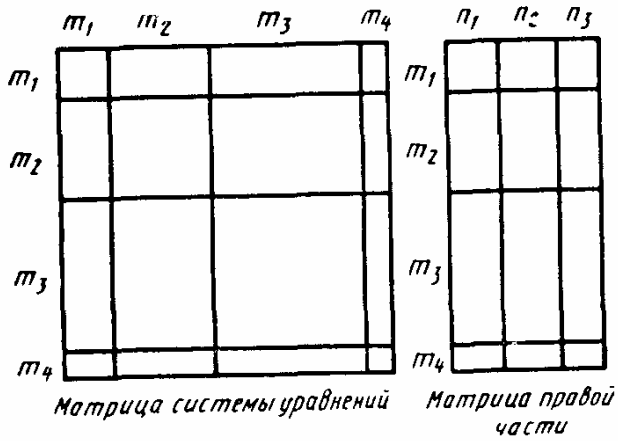


Рис. 3.2. Пример блочной структуры матриц при решении системы уравнений

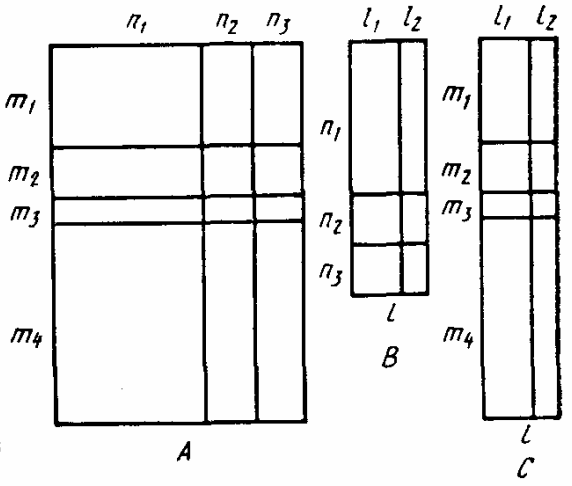


Рис. 3.3. Пример блочной структуры при умножении блочных матриц (C=AB)

строк, т. е. если правую часть мысленно приписать к матрице системы уравнений, должна получиться блочная матрица регулярной структуры (рис. 3.2).

При умножении блочных матриц размеры метаматриц и блоков должны быть такими, чтобы можно было применить обычный алгоритм умножения матриц (элементами этих матриц будут блоки) (рис. 3.3).

При выполнении блочных операций ППП СМПО всегда проверяет соответствие структуры операндов.

Дескрипторы в метаматрице могут указывать не только на обычные, но и на блочные матрицы. В этом случае получаются более сложные иерархические формы связи и хранения матриц — блочные суперматрицы, у которых блоками являются другие блочные матрицы. Создание таких матриц не представляет большого труда. Обратное преобразование суперматрицы к более простой метаматрице возможно с помощью операции

```
&FC      SPFOR (&C)
BSUM     SASP 1,&A,&FC
```

Здесь метаматрица &C размером K×L образуется из метаматрицы A размером M×N, имеющей в качестве блоков метаматрицы A_{ij} размером $M_{ij}\times N_{ij}$. Размеры матрицы C вычисляются по формулам

$$K=\sum_{i=1}^{M} M_{i1};\quad L=\sum_{j=1}^{N} N_{1j}.$$

Каждая матрица A_{ij} записывается в качестве суперблока в метаматрицу &C согласно порядку их расположения в метаматрице A.

Следует отметить, что над прямоугольными метаматрицами можно производить некоторые операции, характерные для обычных

матриц. Это связано с одинаковым форматом представления матриц.

При обработке блочных матриц бывает нужно выполнять необходимые операции над ее блоками. Для этого из метаматрицы выбирают дескриптор нужного блока и пересылают его в заранее зарезервированное поименованное поле длиной 8 байт. Такая выборка может быть выполнена как оператором VBL, так и программами GVBL или VEL или непосредственно с помощью машинных команд (в зависимости от того, что для пользователя удобнее сделать в конкретной ситуации). После этого с блоком можно обращаться, как с обычной матрицей, используя в качестве ее имени имя поля, куда переслан дескриптор. Если после обработки дескриптор блока не изменился, даже если элементы блока изменили свое значение, то записывать дескриптор в метаматрицу не обязательно. Дескриптор блока изменяется только тогда, когда блок надо заменить блоком другого размера или изменить его местоположение.

При создании блочной матрицы сначала создается метаматрица, если она не создана заранее, а потом ее блоки. Создание блоков обычно выполняется в цикле. Цикл должен содержать определение блока с помощью оператора OPR, сжатие стека дисковой памяти с помощью оператора STEK и запись блока в метаматрицу оператором ZBL. Если никакой дополнительной обработки создаваемых блоков не выполняется, то эти блоки будут содержать только нулевые элементы.

3.3. ОРГАНИЗАЦИЯ РАЗМЕЩЕНИЯ МАТРИЦ

Основным местом расположения матриц является рабочий набор данных, который размещается на магнитных дисках. В процессе обработки матрица может загружаться в ОП либо на время выполнения одной операции, либо на более длительный срок. Кроме того, в ППП СМПО могут создаваться и обрабатываться рабочие матрицы, которые временно размещаются в ОП, не занимая места в рабочем наборе данных.

В процессе работы программы может оказаться, что некоторые ранее определенные матрицы больше пользователю не нужны. Чтобы избежать фрагментации, при распределении памяти в рабочем наборе используется принцип стека, в соответствии с которым матрицы, определенные последними, уничтожаются первыми. Определение матриц выполняется с помощью оператора OPR, уничтожение — оператора KBL.

Рассмотрим пример. Пусть имеются матрицы A размером $M \times N$, $B(N \times K)$ и $C(K \times L)$. Необходимо найти их произведение $D = ABC$. Для вычисления AB потребуется промежуточная матрица RAB. Вычисления будем производить по следующей схеме:

1. Определим матрицу D размером $M \times L$.
2. Определим матрицу RAB размером $M \times C$.
3. Вычислим $RAB = A \times B$.
4. Вычислим $D = RAB \times L$.

5. Уничтожим матрицу RAB.

Если бы мы сначала определили матрицу RAB, а потом — D, то не смогли бы уничтожить RAB, сохранив при этом D.

Большинство традиционных алгоритмов линейной алгебры легко приспосабливаются к стековому принципу распределения памяти. Все алгоритмы СМПО по мере возможности определяют сначала результирующие матрицы, а затем рабочие, чтобы последние легко можно было бы уничтожить. Комбинируя этот метод с размещением небольших рабочих матриц в ОП, можно написать эффективные программы, не оставляющие после себя ставшие ненужными рабочие матрицы.

В процессе выполнения задачи пользователя заранее указанного в процедуре SMPOTRV или SMPOV размера рабочего набора может оказаться недостаточно. В этом случае выполняется дополнительное распределение дисковой памяти. Такое распределение может быть выполнено до 15 раз, если на диске есть свободное место и рабочий набор данных состоит не более чем из 16 непрерывных участков. Поскольку потребность в дисковой памяти не всегда просто определить, при нормальном завершении задачи пользователя ППП СМПО сообщает размер использованного участка рабочего набора данных. Эту же информацию можно получить и в процессе выполнения задачи с помощью оператора STAT.

В соответствии со стековым принципом распределения памяти каждый оператор определения матриц открывает один элемент стека дисковой памяти независимо от того, сколько матриц определяется одновременно. При освобождении дисковой памяти число элементов стека уменьшается и память, описываемая несколькими верхними элементами стека, освобождается. Иногда появляется необходимость присоединить несколько верхних элементов стека к предыдущему, чтобы уменьшить число использованных элементов стека и избежать его переполнения. Для этого используется оператор STEK. Однако после его применения нельзя будет уничтожить только матрицы, определенные последним оператором OPR, потому что одновременно с ними будут уничтожены все матрицы, объединенные в один блок оператором STEK.

Размер стека дисковой памяти указывается в программе пользователя в операторе PROG и, следовательно, может меняться от задачи к задаче. Поэтому оператор STEK особенно часто используется при обработке блочных матриц: когда в цикле создаются новые блоки, при этом автоматически формируются новые элементы стека, и обязательно нужно эти новые элементы присоединить к уже существующим, иначе стек дисковой памяти переполнится, сколько бы места под него не было зарезервировано.

При размещении матриц в области данных ОП по умолчанию, т. е. если не указано противное, в ППП СМПО предполагается использовать принцип интерпретации, в соответствии с которым любая матрица размещается в ОП лишь на время выполнения той операции, в которой она участвует. Этот принцип легко реализовать, для этого не требуется большого объема ОП для размещения

матриц, но он приводит к большим затратам времени на ввод-вывод.

С целью сокращения числа операций ввода-вывода, связанных с обменом информацией между ОП и рабочим набором данных, матрицы могут быть размещены на определенный срок в ОП. Размещение может быть осуществлено с помощью операторов, реализующих управляемое или автоматическое распределение памяти. При управляемом распределении памяти используется стековая структура данных, при автоматическом — структура данных в виде очереди.

3.4. УПРАВЛЯЕМОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

Управляемое распределение памяти дает возможность пользователю участвовать в распределении памяти во время выполнения программы. Область действия управляемой памяти локализована в блоке запоминания. Открытие блока запоминания производится оператором ZAPOM, содержащим список матриц, определенных ранее операторами OPR и MATR. Закрытие блока осуществляется оператором UDAL, в котором указываются число закрываемых блоков и имена матриц, выводимых в рабочий набор данных. Указанные в операторе ZAPOM матрицы будут размещены в ОП и останутся в ней до тех пор, пока не будут оттуда удалены с помощью оператора UDAL.

Рассмотрим пример. Пусть имеются матрицы A и B. Необходимо вычислить их сумму, умножить ее поэлементно на A и результат послать в матрицу M. С учетом необходимости уменьшения числа операций ввода-вывода вычисления будем производить по следующей схеме:

1. Определим матрицу M того же размера, что и матрица A.
2. Запомним в ОП матрицы A и M.
3. Вычислим $M=A+B$.
4. Вычислим $M=M\odot A$, где $\odot$ — знак поэлементного умножения.
5. Удалим A и M из ОП; при этом выведем M в рабочий набор.

При выполнении операции 3 матрицы A и M уже находятся в ОП. Поэтому монитор пакета при планировании этой операции выделяет ОП лишь для матрицы B, а затем вводит ее. После выполнения операции память, занимаемая матрицей B, освобождается. На диск она не выводится, поскольку является входным параметром операции и не изменяется. Для выполнения операции 4 не требуется выделять ОП, потому что все операнды уже запомнены, т. е. находятся в ОП. На шаге 5 мы освобождаем ОП, занимаемую матрицами A и M, причем должны указать, что матрица M изменилась, и ее нужно вывести в рабочий набор, чтобы сохранить полученное значение. Поскольку матрица A не изменяется, выводить ее на диск не нужно.

В управляемом режиме область данных ППП СМПО имеет вид, показанный на рис. 3.4. В начале работы она пустая и начинает заполняться с конца. При запоминании матриц занятая часть области данных увеличивается и соответственно сдвигается вверх граница между свободной и занятой частями, при удалении матриц граница опускается вниз. Матрицы, размещаемые в ОП лишь на время операции, располагаются в конце свободной части области данных. В промежутках между выполнением операций свободная часть области данных действительно является свободной и не содержит полезной информации.

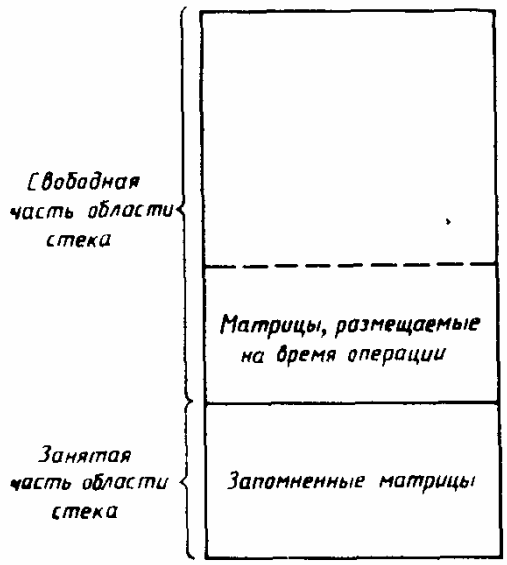


Рис. 3.4. Структура области данных при управляемом распределении памяти

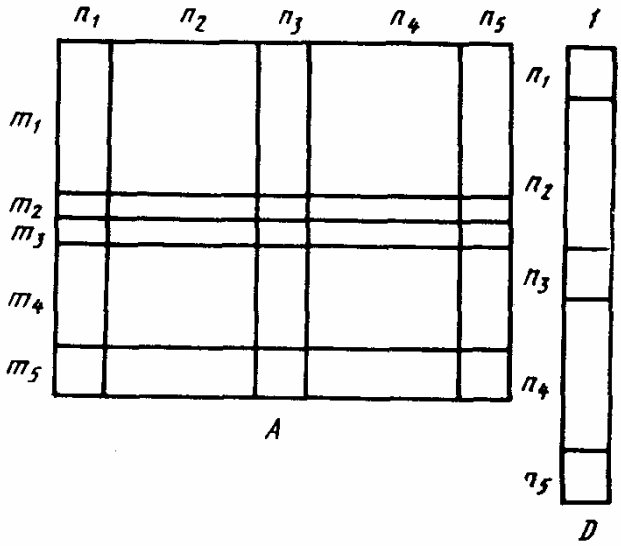


Рис. 3.5. Пример блочной структуры при умножении матрицы A на диагональную матрицу D

При выполнении конкретной задачи может оказаться, что выделенной области памяти недостаточно для запоминания какой-либо матрицы или для временного ее размещения в области данных. В этом случае выполнение задачи пользователя будет прекращено, а на печать будет выведено сообщение ОБЛАСТЬ ДАННЫХ ЗАПОЛНЕНА. Пользователь должен будет или увеличить область данных, или уменьшить размерность решаемой задачи, или перейти от простых матриц к блочным, или уменьшить размеры блоков блочных матриц. Вообще говоря, иногда оценить требуемый размер области данных бывает нелегко, поэтому в конце выполнения задачи пользователя ППП СМПО сообщает размер использованной части области данных. Ту же информацию пользователь может получить с помощью оператора STAT.

Для иллюстрации применения метода управляемого распределения памяти составим алгоритм умножения блочной матрицы на блочную диагональную матрицу. Пусть на вход алгоритма подаются две матрицы: блочная A и диагональная D, которая хранится в виде блочного столбца. Предполагаем, что блочная структура этих матриц соответствует выполняемой операции (рис. 3.5).

Результат получаем на месте исходной матрицы A.

1. Найдем M и N — размеры метаматрицы A. Проверим, равен ли первый размер метаматрицы D величине N, а второй — единице. Если условие не выполняется, готовим соответствующее сообщение пользователю и прекратим работу.

2. Запомним метаматрицы A и D в ОП. Запоминание метаматриц при обработке блочных матриц резко сокращает время решения задачи благодаря уменьшению числа операций ввода-вывода, так как метаматрица постоянно нужна для выборки дескрипторов блоков. Все программы СМПО при обработке блочных матриц запоминают их метаматрицы.

3. Для I от 1 до N выполним пункты с 4 по 12.

4. Выберем из метаматрицы D дескриптор блока, расположенного по координатам (I, 1), и поместим его в заранее заготовленное 8-байтовое поле BLOKD. Теперь с этим блоком можно обращаться как с обычной матрицей.

5. Запомним матрицу BLOKD.

6. Для J от 1 до M выполним пункты с 7 по 10.

7. Выберем из метаматрицы A дескриптор блока, расположенного по координатам (J, I), и поместим его в 8-байтовое поле BLOKD.

8. Если BLOKA — нулевой блок, идем к п. 10.

9. Умножаем матрицу BLOKA на диагональную матрицу BLOKD.

Результат записываем на место BLOKA. Поскольку дескриптор этого блока не изменяется, нет необходимости переписывать этот дескриптор в блочную матрицу, так как в программе умножения на диагональную матрицу предусмотрена проверка соответствия размеров операндов, проверять их перед обращением к этой программе не нужно.

10. Конец цикла по J.

11. Удаляем из оперативной памяти матрицу BLOKD. Поскольку эта матрица в процессе работы не изменялась, ее не нужно выводить на диск. Достаточно просто освободить занимаемую ею память.

12. Конец цикла по I.

13. Удаляем из ОП метаматрицы A и D. Поскольку их содержимое не менялось, не выводим их на диск.

Как видно из этого примера, стековая структура управляемого способа распределения памяти дает возможность просто и эффективно программировать многие алгебраические алгоритмы при обработке блочных матриц.

3.5. АВТОМАТИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

Автоматический режим распределения памяти (режим очереди) применяется в ППП СМПО тогда, когда управляемый режим неудобен или неэффективен. Автоматический режим действует только на блоки блочных матриц и не распространяется на простые матрицы и метаматрицы. При автоматическом режиме применяется принцип очереди: блоки, размещенные в области данных в ОП раньше других, освобождают ее первыми.

Использование автоматического режима распределения памяти не исключает действия управляемого режима. Пользователь и при автоматическом режиме имеет возможность явного запоминания матриц и блоков в области стека.

Режим очереди включается оператором NOCH. Область данных разбивается на две части: область стека (ОСТ) и область очереди (ОЧ) (рис. 3.6), причем в операторе NOCH пользователь указывает размер «свободной» части области стека. Эта область размещается вплотную к области, где расположены запомненные ранее матрицы и метаматрицы. Остальная часть области данных отдается под ОЧ.

Дескрипторы матриц и блоков, обрабатываемых монитором пакета, работающим в режиме очереди, состоят не из двух слов, как в режиме стека, а из трех. Первые два слова имеют обычный формат, а третье используется при организации очереди. Используя это слово, программист указывает монитору, где нужно размещать матрицу: в ОСТ или в ОЧ. Если матрица должна размещаться в ОСТ, третье слово ее дескриптора равняется нулю. Лишь при использовании операторов ZAPOM и UDAL, а также при обработке запомненных матриц дескриптор может состоять из двух слов, т. е.

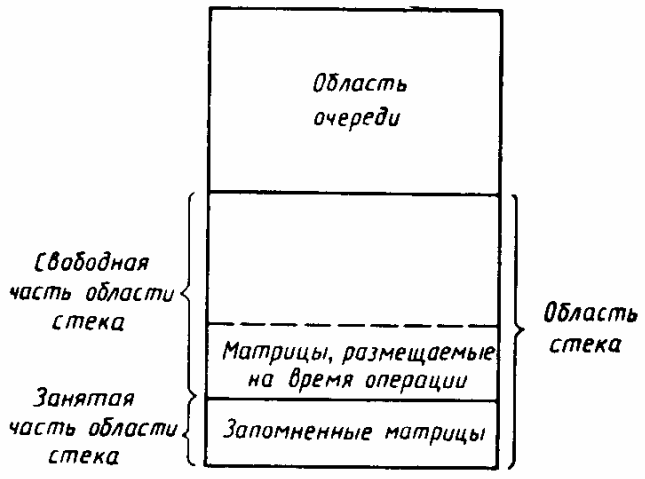


Рис. 3.6. Структура области данных при автоматическом распределении памяти

третье слово приравнивать нулю не обязательно. Если программисту требуется, чтобы нужный блок размещался в ОЧ, должно быть обеспечено следующее: а) метаматрица, содержащая дескриптор данного блока, должна быть запомнена, и ее можно удалять из ОП только после выключения режима очереди; б) третье слово дескриптора должно содержать адрес расположения дескриптора блока в метаматрице. Эти требования будут удовлетворяться автоматически, если перед включением очереди будут запомнены метаматрицы всех блочных матриц, которые будут обрабатываться в автоматическом режиме (удалить их можно будет после отмены режима), а выборка дескрипторов блоков из метаматриц будет выполняться специальными форматами оператора VBL или программы GVBL.

При включенном режиме очереди граница между свободной и занятой частями области стека может передвигаться из-за запоминания и удаления матриц, но граница между ОЧ и ОСТ остается неподвижной.

Область очереди начинает заполняться сверху. Если для размещения очередного блока имеющегося свободного места недостаточно, из ОЧ исключается несколько блоков. Монитор ППП СМПО располагает информацией о том, какие из блоков изменили свое значение, а какие — нет. Первые выводятся на диск, вторые просто исключаются, освобождая ОП.

Режим автоматического распределения памяти выключается оператором КОСН. При этом ОЧ освобождается, и все блоки, изменившие свое значение, выводятся в рабочий набор данных. Структура области данных снова принимает вид, показанный на рис. 3.4.

Рассмотрим пример использования автоматического распределения ОП. При решении симметричной положительно определенной системы уравнений методом Холецкого [36] после факторизации матрицы системы уравнений требуется решить две треугольные системы. Если матрицы системы уравнений и правых частей имеют большой размер (несколько тысяч или десятков тысяч переменных), то их можно хранить только в блочном виде. При решении треугольной системы каждый блок множителя Холецкого используется только один раз, а блоки правой части используются многократно. Поэтому целесообразно накапливать блоки множителя Холецкого в области стека, а блоки правой части — в области очере-

ди. Для этого требуется лишь вычислить размер наибольшего блока множителя, чтобы при организации очереди оставить достаточно места для свободной части ОСТ. Это легко можно сделать, анализируя метаматрицу множителя. Если бы не использовался автоматический режим распределения ОП, то затраты времени на ввод-вывод были бы больше. Аналогичная картина наблюдается, если накапливать в ОЧ как блоки множителя, так и правой части: блоки множителя больше по размеру, и именно они оккупировали бы бóльшую часть ОЧ, что привело бы к частым ее реорганизациям и, как следствие, к увеличению обмена. Нецелесообразно также разделение правых частей на несколько отдельно обрабатываемых столбцов: даже если каждый из них полностью помещается в ОП, для каждого столбца придется заново загружать в память фактор Холецкого, а это еще больше увеличит затраты на обмен. Таким образом, мы приходим к выводу, что предлагаемый здесь способ решения системы уравнений наиболее оптимален.

Принцип очереди также активно используется в алгоритмах метода перемещений ППП СУМРАК на этапе формирования глобальной матрицы жесткости, которая, как известно, формируется путем суммирования матриц жесткости отдельных конечных элементов, причем они могут рассылаться сразу в несколько блоков. Вычислительные трудности при реализации этого процесса связаны с тем, что порядок использования различных блоков матрицы жесткости является очень сложным, даже в какой-то степени хаотичным, особенно при необходимости минимизации ширины ленты, поэтому не представляется возможным написать эффективный алгоритм рассылки, основанный на принципе стека. Здесь применение автоматического распределения памяти уменьшает время формирования матрицы жесткости в 5—10 раз по сравнению с алгоритмом, который при каждой рассылке заново загружает в ОП нужные блоки матрицы жесткости.

Аналогичная ситуация складывается после решения системы уравнений методом перемещений на этапе определения напряжений (усилий) в элементах. Здесь нужно выполнять разборку векторов решения системы уравнений (мы по-прежнему считаем, что матрицы настолько велики, что целиком в ОП не помещаются).

При использовании автоматического режима распределения ОП пользователь в конце работы программы уже не получает информации о размере использованной части области данных: вся выделенная им память будет использована. Если в ОЧ недостаточно места для выполнения очередной операции, выполнение задачи будет прекращено, а на печать выводится сообщение НЕДОСТАТОЧНЫЙ РАЗМЕР ПАМЯТИ ДЛЯ ОЧЕРЕДИ! Эффективность автоматического распределения памяти прямо зависит от размера ОП, выделяемой под очередь: чем больше размер ОП ЭВМ, тем больший объем памяти можно выделить под область очереди и тем быстрее будет решаться задача благодаря минимизации числа операций ввода-вывода.

Глава 4. Организация и управление программными единицами в ППП СМПО

В настоящее время возникла необходимость компьютеризации всех областей деятельности человека в науке и производстве. Это выдвигает повышенные требования к базовым программным средствам, используемым при создании проблемно-ориентированных программных систем. Одним из таких требований является наличие средств, позволяющих использовать идеи структурного программирования на всех этапах проектирования систем [20].

4.1. СТРУКТУРИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

При разработке систем на этапах проектирования и кодирования программ обычно используются два основных подхода: нисходящий и восходящий [20, 23].

При нисходящем, или пошаговом, проектировании задачи и программы, их реализующие, иерархически структурируются и разрабатываются путем последовательного уточнения.

При восходящем проектировании вначале разрабатываются программы низшего уровня. Такой подход наиболее характерен для систем, работающих в реальном масштабе времени, когда окончательные характеристики всей системы прямо зависят от свойств аппаратных средств и программ, управляющих их работой.

На практике чаще всего используются оба подхода. Большинство систем проектируются сверху вниз, но кодирование и тестирование программ осуществляются снизу вверх. Сначала осуществляется автономная отладка отдельных программ, затем комплексов программ и, наконец, всей системы в целом.

ППП СМПО предназначен в основном для нисходящего проектирования систем. С этой целью в него введены три типа программных единиц: основная программа (ОСП), стандартный алгоритм (СА), стандартная подпрограмма (СТП). Если принять следующую структуризацию системы: функция, задача, операция, то ОСП служит для реализации функции, СА — задачи, СТП — операции.

ОСП позволяет записать алгоритм решения проблемы на функциональном, самом крупном, уровне. Алгоритм решения обычно записывается путем перечисления СА, необходимых для реализации конкретной функции. В свою очередь программирование СА сводится к обращению к другим СА или СТП. На низшем уровне находится СТП, которая не содержит вызовов каких-либо других программных единиц.

ОСП и СА могут использовать практически все операторы входного языка ППП СПО (включая команды языка ассемблера). СТП могут кодироваться на фортране или на ассемблере с использованием при этом только тех операторов ППП СМПО, которые не передают управление с возвратом в другие программы.

Все СА и СТП имеют только один вход и один выход. Они являются повторно используемыми [28], что позволяет сократить временные затраты на передачу управления тем СА и СТП, которые вызываются неоднократно. Это достигается благодаря тому, что монитор ППП СМПО сохраняет в ОП копии уже использованных программ.

Особенностью СА является возможность его неоднократного выполнения с различными фактическими параметрами путем только одного обращения к нему. Число выполнений явно указывается в списке фактических параметров.

В существующих программах пакета реализована концепция защитного программирования [23] путем контроля характеристик входных параметров — размеров матриц и списков, значений координат и элементов списков.

Автоматизация операций ввода-вывода, наличие различных типов программных единиц, развитые средства контроля и слежения за ходом выполнения программ, богатая библиотека СА и СТП, возможность пополнения этой библиотеки путем разработки собственных программ делает ППП СМПО мощным инструментальным средством создания сложных программных систем, примером которой может являться ППП СУМРАК.

4.2. ОСНОВНАЯ ПРОГРАММА

ОСП, составленная пользователем ППП СМПО, обычно выполняется в качестве главной программы пункта задания. Она получает управление от операционной системы (ОС) в начале работы и возвращает управление ей по окончании работы.

Основная программа должна начинаться оператором PROG. В операторе PROG пользователь может указать размер стека дисковой памяти, если его не устраивает принятое по умолчанию значение 50. Если рабочий набор данных размещается на двух магнитных дисках, пользователь должен проинформировать об этом ППП СМПО, сообщив в операторе PROG размер в дорожках той части набора, который размещается на первом диске. В операторе может указываться размер области ОП, выделяемой для размещения СА и СТП. По умолчанию ее размер равняется 40 Кбайт.

Исполняемая часть основной программы завершается оператором KPROG, который заканчивает выполнение программы пользователя и возвращает управление операционной системе.

В неисполняемой части основной программы располагаются статические поля, константы и списки, необходимые для вызова и работы СА и СТП.

Текст основной программы должен завершаться оператором END.

Рассмотрим, как происходит выполнение основной программы.

ОС передает управление основной программе. Начинает выполняться оператор PROG, который выполняет следующие действия. Инициализируются базовые регистры 10, 11, 12. Используя инфор-

мацию о разделе ОП, переданную через поле PARM оператора EXEC, оператор PROG с помощью макрокоманды GETMAIN [28] выделяет ОП для области данных. Выделенная ОП обнуляется. Формируется системная область сохранения. С помощью макрокоманды ОС LOAD загружается в ОП программа управления данными (монитор ППП СМПО). Открываются наборы данных для ввода, печати и рабочий набор данных. Подключается блок обработки программных прерываний. Печатается время начала выполнения основной программы и текст ОСНОВНАЯ ПРОГРАММА «имя-прог.», где «имя-прог.» — название основной программы. В регистр 15 загружается адрес монитора, а в регистр 13 — адрес системной области сохранения. Выполнение оператора PROG заканчивается и ППП СМПО готова к выполнению задач пользователя.

После выполнения задач пользователя срабатывает оператор KPROG. Он печатает текст КОНЕЦ ОСНОВНОЙ ПРОГРАММЫ «имя-прог.», текущее время и статистическую информацию, обеспечиваемую оператором STAT.

Рассмотрим структуру системной области сохранения. Она содержит информацию, необходимую для связей между различными компонентами СМПО. Адрес начала этой области всегда находится в регистре 13, поэтому содержимое ее доступно любой программе в ППП СМПО. Системная область сохранения содержит следующие байты информации:

0 — системные переключатели, устанавливаемые операторами SLED и KONTR;

1—3 — адрес монитора;

4—11 — два адреса, связывающие в цепочку области сохранения;

12—71 — область для сохранения общих регистров 14, 15, 0, 1, ..., 12;

72—103 — область для сохранения регистров с плавающей точкой 0, 2, 4, 6;

104—107 — адрес области данных;

108—111 — адрес блока управления данными (DCB) для наборов данных на магнитной ленте;

112—123 — адреса DCB для рабочего набора данных, входного набора перфокарт, выходного набора для печати;

124—127 — адрес начала стека дисковой памяти;

128—131 — адрес области для накопления статистической информации о прохождении задачи пользователя;

132—135 — адрес области, содержащей информацию монитора о распределении области данных на области стека и очереди;

136—139 — поле, используемое для организации возврата из стандартных алгоритмов;

140—143 — адрес модуля SMPODANZ, содержащего параметры работы монитора;

144—147 — адрес DCB для продолжения рабочего набора данных на другом томе.

Поля системной области сохранения не используются в явном виде программистом. Они нужны монитору, операторам и некоторым сервисным программам ППП СМПО. Причем явные смещения полей от начала области не используются, а применяются символические имена. Например, адресу 104 соответствует имя ODDISP (смещение адреса области данных), адресу 116 — имя VVDDISP (смещение адреса DCB набора данных для ввода) и т. д. Значения символическим именам задаются операторами оформления программных единиц с помощью оператора DISPLACE, обращение к которому они содержат.

Рассмотрим использование общих регистров в основной программе. В качестве базовых регистров в основной программе используются регистры 10, 11, 12. Они позволяют адресовать программу длиной до 12 Кбайт. Использование других базовых регистров не допускается, поэтому программы длиной более 12 Кбайт требуют разбиения на стандартные алгоритмы. Регистр 13 содержит адрес системной области сохранения, регистр 15 — адрес монитора. Регистры со 2-го по 9-й полностью находятся в распоряжении пользователя. Ни один оператор ППП СМПО не изменяет их содержимого. Регистры 0, 1, 14 могут быть использованы, но операторы ППП СМПО изменяют их, поэтому их использование ограничено пространством между двумя соседними выполнимыми операторами. Регистры с плавающей точкой пользователь может использовать свободно.

В основной программе могут использоваться все операторы ППП СМПО, кроме операторов оформления СА и СТП.

4.3. СТАНДАРТНЫЙ АЛГОРИТМ

Стандартный алгоритм является многоуровневой стандартной программной единицей и применяется, как правило, для реализации относительно самостоятельных задач или частей одной задачи. Он может содержать произвольное число обращений к другим СА и СТП. Единственное ограничение на вложенность обращений СА друг к другу — запрет рекурсивного вызова.

СА должен начинаться оператором STAL, содержащим список формальных параметров. Формальные параметры по типу и числу должны соответствовать фактическим параметрам, значения которых они получат. Под формальные параметры, имена которых указаны в операторе STAL, должны быть зарезервированы поля, неявная длина которых соответствует числу байтов, занимаемых фактическими параметрами. Для матриц должны выделяться двойные слова, в которые будут пересылаться дескрипторы матриц, являющихся фактическими параметрами.

Исполняемая часть СА должна завершаться оператором KSTAL, который завершает выполнение стандартного алгоритма и возвращает управление вызывающей программе.

Текст СА должен завершаться оператором KSTP, за которым должен следовать оператор END.

Правила использования регистров в СА те же, что и в основной программе.

Разберем, как происходит передача параметров в СА. При передаче матрицы ее дескриптор переписывается в выделенное в СА двойное слово. Матрица может изменять свое значение, но если дескриптор ее не меняется, новое значение матрицы будет передано в вызывающую программу после завершения СА. Если в СА матрица запоминается, что изменяет ее дескриптор, то ее нужно в этом же СА удалить, восстановив дескриптор. Таким образом, матрицы передаются в СА по значению и результату. Скалярные величины передаются в СА по значению. Те скалярные параметры, которые должны быть переданы и по результату, должны явно указываться в операторе KSTAL. Так же могут быть переданы матрицы, дескрипторы которых изменяются.

Имеется дополнительная возможность передачи параметров по результату. Для этого используются операторы SPFOR и SPFAK. Оператор SPFOR записывается в вызывающей программе. В нем перечисляются фактические результаты работы СА. Этот оператор может быть использован и как неисполняемый, и как исполняемый. В последнем случае он должен исполняться до вызова алгоритма, формирующего перечисленные результаты. Оператор SPFOR определяет имена результирующих параметров и отводит для них 8-байтовые поля. Он может использоваться для передачи в вызывающую программу как дескрипторов матриц, так и числовых величин.

В паре с оператором SPFOR должен использоваться оператор SPFAK. Он кодируется в СА и должен содержать имена сформированных результатов. Матрицы, передаваемые этим оператором, не должны быть запомнены в момент передачи. Поэтому удобно использовать оператор SPFAK перед оператором KSTAL.

Операторы SPFOR и SPFAK связываются специальным параметром, передаваемым через список параметров оператора STAL.

В СА можно использовать все операторы ППП СМПО, кроме операторов оформления основной программы и СТП. Вызов СА осуществляется с помощью операторов SA или SASP.

4.4. СТАНДАРТНАЯ ПРОГРАММА

Стандартная подпрограмма является одноуровневой стандартной программной единицей, которая не содержит обращений к другим программам ППП СМПО или к монитору пакета. СТП используется для реализации отдельных операций. Написание СТП требует хорошего знания ассемблера.

СТП должна начинаться оператором STP, в котором программист может указать номер базового регистра. Если номер регистра не указан, в качестве базового будет использован регистр 15.

Исполняемая часть СТП должна завершаться оператором RETURN.

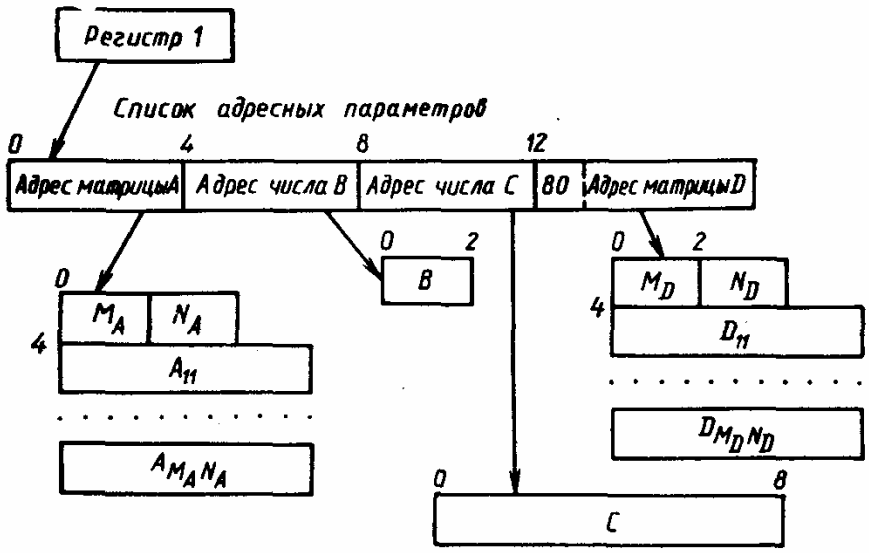


Рис. 4.1. Пример списка параметров, передаваемого в стандартную программу

Текст СТП должен завершаться оператором KSTP, за которым должен следовать оператор END.

Передача параметров в СТП осуществляется по адресу. После оператора STP адрес списка параметров находится в регистре 1. Список параметров в соответствии со стандартом ОС ЕС [28] состоит из адресных параметров. Он может иметь переменную длину. В качестве признака конца списка в старшем байте последнего параметра содержится X'80'.

Когда в качестве параметров СТП используются матрицы, то монитор размещает их в ОП, если они уже не были запомнены. Адрес в списке параметров, соответствующий передаваемой матрице, будет указывать на ее размеры, расположенные перед первым ее элементом.

На рис. 4.1 показан пример списка параметров. В СТП передается четыре параметра: матрица A, целое число в формате полуслова B, число двойной точности C и матрица D. Соответственно и список параметров содержит четыре адреса.

В числе параметров СТП может быть только одна результирующая матрица, которая должна быть последним параметром в списке, и произвольное число входных матриц. Если алгоритм, который требуется реализовать, должен иметь несколько результирующих матриц, его нужно будет реализовывать в виде СА, а не СТП.

Программист может свободно пользоваться в СТП всеми регистрами, кроме базового и регистра 13. При необходимости можно использовать несколько базовых регистров. СТП не имеет собственной области сохранения. Все действия по сохранению и восстановлению регистров производятся монитором пакета.

В СТП можно передавать переменное число параметров. В этом случае СТП должна настраиваться на число параметров, используя то, что последний параметр списка содержит признак X'80' в старшем байте.

Так как параметры передаются в СТП по адресу, изменение формальных параметров вызовет изменение фактических. Нельзя изменять матрицы, которые переданы не в качестве последнего параметра из-за неоднозначности действия монитора по отношению к ним. Если исходные матрицы не были запомнены в ОП перед обращением к СТП, то монитор их вызовет в ОП из рабочего набора данных, а после выполнения СТП — уничтожит их. Если они были предварительно запомнены, то монитор не производит с ними никаких действий. В этом случае модифицированная исходная матрица доступна вызывающей программе, чего не будет в первом случае.

В СТП можно использовать операторы DCHR, DIAG, EOJ, GDS, GEQU, GSTD, GSTH, KONTR, SCAL, SLED, ADSP, CSP без всяких ограничений. Можно также использовать операторы IR, UPAR VCHT, ZR, но они изменяют содержимое регистров 0,1,14. Операторы CON, OTPE, PTIMER, UTIMER, TEKST, VVOD требуют также, чтобы базовым был регистр со 2-го по 12-й. Остальные операторы СМПО использовать в СП нельзя. Вызов СТП осуществляется операторами V, VP, VPL, SOZ.

4.5. УПРАВЛЕНИЕ СТАНДАРТНЫМИ ПРОГРАММНЫМИ ЕДИНИЦАМИ

Стандартные программные единицы (СПЕ) в ППП СМПО оформляются в виде отдельных загрузочных модулей. При выполнении задачи пользователя они загружаются в ОП динамически, по мере надобности, благодаря чему достигается значительная экономия памяти.

Некоторые СПЕ в процессе прохождения задачи пользователя вызываются неоднократно. Если операционная система работает в режиме мультипрограммирования с переменным числом задач (*MVT* или *SVS*), то используя для загрузки в ОП и передачи управления макрокоманду LINK [28], мы избегаем повторного ввода СПЕ с диска, если в ОП осталась копия нужной программы после предыдущего обращения. Такие копии удаляются из ОП службой памяти лишь в том случае, когда недостаточно места для размещения новых СПЕ. В этом случае отпадает необходимость в специальном блоке управления СПЕ.

Если ОС ЕС работает в режиме мультипрограммирования с фиксированным числом задач (MFT), то при использовании макрокоманды LINK для загрузки и передачи управления СПЕ ранее задействованные копии не сохраняются и возможны большие потери времени на поиск и загрузку программ, особенно если они вызываются в цикле. Поэтому в ППП СМПО предусмотрен специальный сегмент управления СПЕ, который является составной частью монитора пакета.

В основе работы сегмента лежит следующий принцип: пока в разделе ОП, где выполняется задача пользователя, есть свободное место, ранее загруженные СПЕ не удаляются. Если же места нет, то удаляются ранее загруженные программы, за исключением

активных, т. е. тех, чье выполнение не было завершено к моменту реорганизации из-за вызова ими других СПЕ. Поскольку реорганизации выполняются относительно редко, такая схема весьма эффективна.

Супервизор, размещая загрузочный модуль в ОП, настраивает его на место в ОП, поэтому нежелательно переписывать введенную программу в другую область ОП. Только самоперемещаемые программы сохранили бы при этом свою работоспособность. Поскольку в ППП СМПО не все программы самоперемещаемые (например, подпрограммы, написанные на фортране, не обладают этим свойством), сегмент управления СПЕ не переписывает их на другое место. Вместо этого, благодаря использованию информации из таблиц супервизора, просто определяется наличие свободной памяти и момент, когда надо выполнять реорганизацию. При этом супервизор сам решает, с какого адреса размещать программу.

Для размещения СПЕ используется макрокоманда ОС LOAD, для их удаления — DELETE. Передачу управления загруженной программе можно было бы организовать непосредственно, но в данном сегменте для этого используется макрокоманда LINK, которая формирует признак активности СПЕ. Для вычисления размера СПЕ, расположенной в библиотеке на диске, используется макрокоманда BLDL.

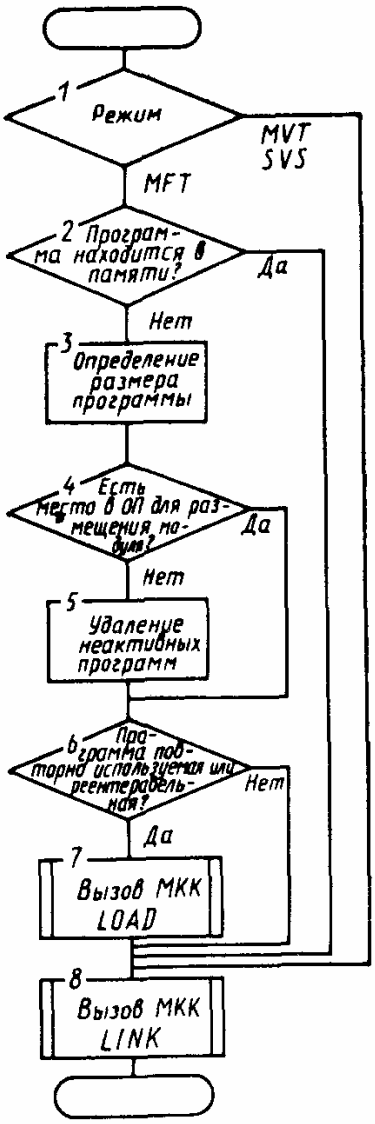


Рис. 4.2. Схема сегмента управления стандартными программными единицами

Рассмотрим схему работы сегмента управления СПЕ (рис. 4.2).

Блок 1. С помощью резидентной таблицы векторов связей (CVT) [39] анализируем режим работы ОС ЕС. Если ОС работает в *MVT* или *SVS*, переходим к блоку 8.

Блок 2. Просматриваем список загруженных программ, отыскивая в нем нужную СПЕ. Адрес начала списка берем из блока *TCB* задачи [28]. Если СПЕ найдена, переходим к блоку 8.

Блок 3. Для данной СПЕ выполняем макрокоманду BLDL, которая помещает в ОП элемент оглавления библиотеки загрузочных модулей. Из него выбираем размер СПЕ.

Блок 4. Просматриваем список свободных участков ОП раздела и определяем, достаточен ли объем ОП для размещения СПЕ.

Адрес начала списка расположен в блоке *TCB* задачи. Если ОП достаточно, переходим к блоку 6.

Блок 5. Снова просматриваем список загруженных СПЕ и удаляем все неактивные с помощью макрокоманды DELETE. При этом удаляются только программы СМПО, которые распознаются по префиксу SMPO. Модули супервизора не удаляются.

Блок 6. Анализируя элемент оглавления библиотеки для СПЕ, определяем, является ли она повторно используемой. Если нет, переходим к блоку 8.

Блок 7. Загружаем СПЕ в ОП макрокомандой LOAD.

Блок 8. С помощью макрокоманды LINK вызываем СПЕ и передаем ей управление.

Глава 5. Управление матричными данными в ППП СМПО

Главной целью создания ППП СМПО являлось обеспечение возможности реализации вычислительных алгоритмов, записанных в матричной форме. При программировании таких алгоритмов на любом из существующих языков программирования ЕС ЭВМ приходится иметь дело с большим числом матриц, размеры которых зависят от особенностей алгоритмов и начальных условий решаемых задач. В условиях ограниченности основной памяти ЭВМ в ней можно разместить только часть используемых матриц. Остальные должны находиться во внешней памяти и вызываться в основную по мере необходимости. Использование при этом явных операторов ввода-вывода значительно усложняет программирование, заставляет программиста концентрировать свое внимание на вспомогательных с точки зрения вычислительного алгоритма операциях по организации и управлению матричными данными. От всего этого пользователя освобождает ППП СМПО.

5.1. ВЫБОР МЕТОДА ДОСТУПА ДЛЯ ОБРАБОТКИ РАБОЧЕГО НАБОРА ДАННЫХ

Матрица в рабочем наборе данных на магнитных дисках хранится в виде совокупности записей неопределенной длины. При этом длина всех записей, кроме последней, равна длине дорожки магнитного диска. Последняя запись (остаток поля матрицы) может занимать часть дорожки. Если длина следующей матрицы * меньше длины дорожки, то она размещается в виде одной записи, либо на последней частично заполненной дорожке, если на той достаточно места для полного размещения записи, либо в противном случае, начиная со следующей дорожки. Матрицы, длина которых больше, чем длина дорожки, всегда начинаются с новой дорожки. Такая организация хранения матриц ускоряет и упроща-

* Длина матрицы размерами $m \times n$ равна $(8mn+4)$ байтов.

ет работу с набором данных. Потери дисковой памяти при этом незначительны. При распределении памяти для рабочего набора данных в ППП СМПО используется принцип стека. В соответствии с ним метод доступа должен обеспечивать появление на верхушке стека новых его элементов и выталкивание ставших ненужными элементов. Появлению новых элементов соответствует распределение памяти (оператор OPR), выталкиванию старых — освобождение (оператор KBL). Кроме того, метод доступа должен обеспечивать ввод-вывод любой матрицы из середины стека (операторы V, VP, VPL, ZAPOM, UDAL). Поскольку записи в рабочем наборе будут иметь различную длину, дисковая память, освобожденная при выталкивании элементов из стека, должна размечаться заново, под новые длины записей. При использовании стандартных методов доступа ОС ЕС [28] такую разметку может обеспечить базисный последовательный метод доступа (*BSAM*), но не базисный прямой метод доступа (*BDAM*). С другой стороны, ввод-вывод матриц из середины стека может быть выполнен как через *BSAM*, так и через *BDAM*. Однако если ввод оба метода доступа выполняют с одинаковой скоростью, то при выводе с помощью *BSAM* выводимую запись необходимо сначала считать в память и лишь после этого записать. Такой режим работы увеличивает почти в 2 раза время записи информации. Итак, получается, что при разметке носителя (определение матриц) нужно использовать *BSAM*, а при вводе-выводе из середины стека (чтение и изменение содержимого матриц) — *BDAM*.

Так как определение матриц выполняется реже, чем их обработка, можно перед определением матрицы закрыть блок управления данными (DCB) метода *BDAM* и открыть аналогичный блок для *BSAM*, а после определения — закрыть блок управления для *BSAM* и открыть для *BDAM*. Поскольку в процессе обработки задачи пользователя матрицы могут определяться в любое время, такой метод работы требует дополнительных временны́х затрат на постоянное открытие-закрытие DCB. Можно поступить следующим образом: постоянно держать открытыми два DCB для одного набора данных. ОС ЕС такую возможность допускает, хотя и не слишком рекомендует [28].

Рассмотренный способ работы использовался ранее в ППП СМПО и доказал свою приемлемость и достаточно высокую эффективность. Однако имеется возможность значительно ускорить выполнение операций обмена. Дело в том, что при использовании стандартных методов доступа ОС ЕС (*BSAM* и *BDAM*) ввод-вывод информации выполняется отдельными записями, длина которых не превышает размера дорожки. Когда матрица располагается на нескольких смежных дорожках, за тот промежуток времени, который требуется для завершения ввода (вывода) одной дорожки и инициирования ввода (вывода) следующей, диск успевает начать следующий оборот. Таким образом, требуется дополнительный оборот, поскольку операция обмена всегда выполняется начиная с первых байтов записи. Например, для ввода матрицы, занима-

ющей 9 дорожек, потребуется 17—18 оборотов, в то время как ее можно было бы прочесть за 9—10 оборотов. Поэтому, в настоящее время в ППП СМПО используется элементарный метод доступа на физическом уровне с использованием макрокоманды EXCP [28].

Все запросы на обмен с рабочим набором данных в ППП СМПО выполняет модуль SMPOEXCP, являющийся составной частью монитора. Входной информацией для этого модуля является дескриптор матрицы и адрес ее расположения в ОП. Модуль выделяет из дескриптора дисковый адрес матрицы в относительной форме (*TTR*) и переводит его в абсолютную (*CCHHR*) с помощью резидентной программы преобразования, входящей в ядро ОС. Зная длину матрицы, модуль формирует программу канала для той ее части, которая размещается в пределах одного цилиндра. В этой программе в одну цепочку команд связываются все операции обмена, причем для переключения дорожек используется команда «установка головки». Затем эта программа канала выполняется. Если матрица продолжается с одного цилиндра на другой, будет сформирована новая программа канала и т. д. Причем эта схема работы применяется как для обработки матриц, так и для их определения. Появляются лишь следующие различия. При обработке используются команды «чтение данных» (X'06') или «запись данных» (X'07'), а поиск выполняется по идентификатору обрабатываемой записи (команда X'31'). При определении используется команда «запись счетчика, ключа и данных» (X'1D'), а поиск выполняется по идентификатору предыдущей записи [16]. Дополнительное ускорение при определении матриц достигается за счет того, что канал не передает нулевую матрицу, а сразу после начала операции отключается. Обнуление поля выполняет устройство без участия канала. Следовательно, операция ввода-вывода заканчивается быстрее, чем при выполнении передачи данных, в соответствии с условием «канал кончил».

Таким образом, в ППП СМПО разработан и используется собственный метод доступа, который в данном случае почти в два раза эффективнее стандартных методов доступа ОС ЕС.

Поскольку управление данными ОС ЕС не поддерживает многотомных наборов данных на диске с произвольным доступом к информации и требует дополнительных временны́х затрат на выполнение программы EOV при переключении томов [28], в ППП СМПО принят простейший путь организации ввода-вывода для набора данных на двух дисках. Фактически используется два отдельных набора данных, каждый из которых располагается на одном томе. Модуль SMPOEXCP обеспечивает автоматическое переключение с одного тома на другой, т. е. с одного набора на другой. Поскольку оба набора постоянно открыты, не требуется никаких временны́х затрат на их переключение. В дескрипторе матрицы (пятый бит четвертого байта) содержится информация о номере тома, на котором она размещена.

При выполнении программы пользователя может оказаться, что той области дисковой памяти, которая выделена для решения

задачи, недостаточно. В этом случае модуль SMPOEXCP вызывает дополнительное распределение памяти на диске с помощью макрокоманды EOV [28] и выдает пользователю сообщение НЕПРАВИЛЬНЫЙ ЗАПРОС В/В. МОНИТОР ВЫДЕЛЯЕТ ДОП. ПАМЯТЬ НА ДИСКЕ! Если на диске нет свободных участков памяти или набору уже выделено 16 непрерывных участков, программа супервизора ввода-вывода EOV завершит задачу пользователя аварийно. При нормальном завершении задания ППП СМПО сообщит пользователю объем реально использованной дисковой памяти.

После завершения любой операции ввода-вывода определяется правильность ее выполнения. Для анализа ошибок используется макрокоманда ОС SYNADAF [28], которая готовит текст сообщения об ошибке. Так как в этом тексте операционная система сообщает вместо реального адреса сбойной дорожки адрес первой дорожки цилиндра, в котором произошел сбой, его необходимо исправить, что и выполняет модуль SMPOEXCP перед выдачей сообщения.

Модуль SMPOEXCP автоматически настраивается на характеристики конкретного устройства прямого доступа. Для этого используется макрокоманда DEVTYPE и выбираются соответствующие параметры из таблицы устройств.

5.2. ПРОГРАММА УПРАВЛЕНИЯ ДАННЫМИ ППП СМПО

Программа управления данными (монитор) обрабатывает все операторы обращения ППП СМПО, обеспечивая их необходимыми для выполнения данными и подпрограммами.

Монитор пакета является реентерабельной (повторно входимой) [28] программой, благодаря чему он может одновременно обслуживать любое число подзадач. Оперативная память, необходимая для его работы, подключается во время редактирования основной программы или стандартной подзадачи в виде модуля SMPODANZ, если рабочий набор данных размещается на одном устройстве, или модуля SMPODNZ2, если рабочий набор требует двух устройств. Модуль SMPODANZ, кроме полей информации, содержит макрокоманды DCB [28] для рабочего набора данных, входного символьного набора, выходного набора для печати, набора для магнитной ленты. Модуль SMPODNZ2 содержит также DCB для продолжения рабочего набора на втором томе. Выбор между этими двумя модулями происходит во время макрообработки оператора PROG.

Связь монитора с информацией из SMPODANZ (SMPODNZ2) осуществляется через фиктивную секцию [16]. Адрес DANZ (DNZ2), взятый из области сохранения ППП СМПО, построенной в операторе PROG, загружается в регистр 9 в начале работы монитора, после чего для него становятся доступны поля DANZ (DNZ2).

Монитор пакета загружается в ОП по макрокоманде LOAD [28] на этапе выполнения оператора PROG. Далее оператор PROG настраивает монитор на размеры областей данных и подпрограмм, открывает наборы данных и связывает их с монитором. После этого монитор готов к работе.

Функция монитора заключается в интерпретации его входного языка. Предложения этого языка подготавливаются операторами обращения ППП СМПО на основании информации, указываемой программистом. За одно обращение монитор обрабатывает одно предложение языка. Грамматика его описывается приведенными ниже правилами в нормальной форме Бэкуса:

1. ⟨предложение⟩ : : = ⟨имя СПЕ⟩ [⟨спис отн адр⟩] ⟨СК⟩ [⟨отн адрр⟩]
2. ⟨имя СПЕ⟩ : : = ⟨Н слово⟩ | ⟨имя⟩
3. ⟨спис отн адр⟩ : : = ⟨отн адр⟩ | ⟨спис отн адр⟩ ⟨отн адр⟩
4. ⟨СК⟩ : : = 8000 | 8080 | 8100 | 8200 | 8300 | 8400 | 8500 | 8600 | A000 |
5. ⟨отн адр⟩ : : = ⟨рег базы⟩ ⟨смещение⟩
6. ⟨отн. адрр⟩ : : = ⟨отн адр⟩
7. ⟨Н слово⟩ : : = ⟨Н байт⟩ ⟨Н байт⟩ ⟨Н байт⟩ ⟨Н байт⟩
8. ⟨имя⟩ : : = ⟨НН байт⟩ ⟨НН байт⟩ ⟨НН байт⟩ ⟨НН байт⟩
9. ⟨рег базы⟩ : : = A | B | C
10. ⟨смещение⟩ : : = ⟨цифра⟩ ⟨Ц байт⟩
11. ⟨Н байт⟩ : : = ⟨Н⟩ ⟨Н⟩
12. ⟨Ц байт⟩ : : = ⟨цифра⟩ ⟨цифра⟩
13. ⟨НН байт⟩ : : = ⟨НН⟩ ⟨НН⟩ | ⟨НН⟩ ⟨Н⟩ | ⟨Н⟩ ⟨НН⟩
14. ⟨цифра⟩ : : = ⟨Н⟩ | ⟨НН⟩
15. ⟨Н⟩ : : = 0
16. ⟨НН⟩ : : = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E

Монитор пакета, обрабатывая предложение входного языка в зависимости от значения сервисного кода (нетерминал ⟨СК⟩), выполняет те или иные действия над операндами, чьи относительные адреса (нетерминал ⟨отн адр⟩) указаны в предложении. Кроме того, для сервисных кодов 8000, 8080, 8100, 8200, A000 монитор формирует предложение выходного языка, вызывает программу, чье имя указано в предложении (нетерминал ⟨имя⟩) и передает ей управление. Грамматика выходного языка проста. Его предложение — это имя программы и список параметров, оформленных по правилам ЕС ЭВМ:

17. ⟨пр вых яз⟩ : : = ⟨имя⟩ ⟨спис⟩ пар
18. ⟨спис пар⟩ : : = [⟨спис абс адр⟩] ⟨приз кон⟩ ⟨абс адр1⟩
19. ⟨спис абс адр⟩ : : = ⟨абс адр⟩ | ⟨спис абс адр⟩ ⟨абс адр⟩

20. ⟨абс адр⟩ : : = ⟨Н байт⟩ ⟨Ц байт⟩ ⟨Ц байт⟩ ⟨Ц байт⟩

21. ⟨абс адр 1⟩ : : = ⟨Ц байт⟩ ⟨Ц байт⟩ ⟨Ц байт⟩

22. ⟨приз кон⟩ : : =80

Этот список параметров передается вызываемой программе. После окончания ее выполнения монитор завершает обработку предложения входного языка и возвращает управление вызвавшей его программе.

Рассмотрим функционирование монитора на примере обработки предложения с сервисным кодом X'8200' (рис. 5.1) [2].

Блок 1. Сохраняется содержимое общих регистров и регистров с плавающей точкой в системной области сохранения.

Блок 2. Производится дескрипторный анализ результата (нетерминал ⟨отн адрр⟩), который будет получен после выполнения операции, указанной в предложении. Если в качестве результата выступает дескриптор с дисковым адресом, выполняется блок 3, иначе — блок 5.

Блок 3. Выделяется место в области данных для результата операции.

Блок 4. Программа управления данными вызывает модуль SMPOEXCP для чтения матрицы результата из рабочего набора в область данных.

Блок 5. Определяется число необработанных элементов исходных данных (нетерминал <спис отн адр>). Если все элементы обработаны, переходим к блоку 9.

Блок 6. Производится дескрипторный анализ элементов исходных данных из списка параметров (нетерминал <отн адр>). Если элемент — дескриптор с дисковым адресом, то выполняется блок 7. В противном случае переходим к блоку 5.

Блок 7. Выделяется место в области данных для матрицы элемента исходных данных.

Блок 8. Вызывается модуль SMPOEXCP для чтения матрицы из рабочего набора в область данных. Затем переходим к блоку 5.

Блок 9. Блок управления стандартными программными единицами (см. разд. 4.5) обеспечивает размещение вызываемой СПЕ (нетерминал <имя>) в ОП и передает ей управление. Если включен режим SLED 1, печатаются имя вызываемой программы и текущее время.

Блок 10. Выполняется вызванная программа.

Блок 11. Производится дескрипторный анализ результата. Если результат — не матрица, переходим к блоку 14.

Блок 12. Если установлен режим SLED 2 и матрица результата не была запомнена или установлен ре-

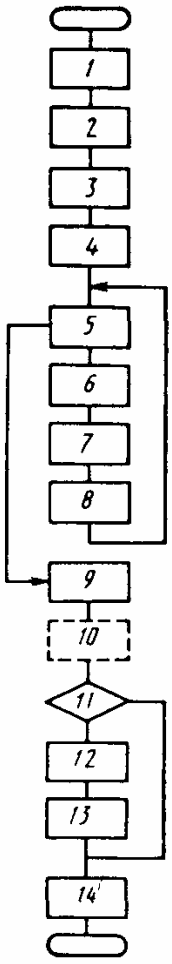


Рис. 5.1. Схема функционирования монитора для сервисного кода X'8200'

жим SLED 4 и она заполнена, то ее содержимое распечатывается. Печать является бесформатной (см. оператор PE), имя матрицы — REZULT.

Блок 13. Вызывается модуль SMPOEXCP для вывода матрицы из области данных в рабочий набор, если выполнялся блок 4.

Блок 14. Восстанавливается содержимое всех регистров. Управление возвращается в вызывающую программу.

Блок-схема функционирования управляющего модуля была описана упрощенно. Рассмотрим подробнее работу некоторых блоков.

Дескрипторный анализ включает в себя следующие действия. По относительному адресу элемента вычисляется абсолютный адрес (a). Проверяется нулевой байт по этому адресу. Если он равен X'80', то считается, что рассматриваемый элемент — дескриптор матрицы. В дескрипторе проверяются младшие 3 бита 4-го байта. Если они равны B'000', это означает что матрица уже запомнена в области данных, и адрес ее начала (b) находится в байтах 5—7 дескриптора. Если они равны B'010', то матрица объявлена оператором MATR, но еще не запомнена и такое ее использование является ошибкой, о чем будет сообщено пользователю. Если они равны B'001' или B'101', то матрица расположена в рабочем наборе на первом или втором его томе соответственно, причем ее дисковый адрес (d) находится в байтах 5—7. В этом случае для матрицы будет отведено место в области данных (c — адрес ее начала). В формируемое монитором предложение выходного языка будет занесен один из адресов:

a — если элемент не является дескриптором;

b — если элемент является дескриптором, описывающим матрицу в области данных;

c — если элемент является дескриптором, описывающим матрицу из рабочего набора.

В первом слове сформированного монитором предложения содержится название вызываемой подпрограммы. Адрес предложения будет увеличен на 4 в момент передачи управления в подпрограмму и будет указывать на список параметров. В последний элемент списка помещается признак конца — X'80'.

Для обработки предложения входного языка с сервисным кодом X'8200' используется максимальное число функциональных блоков монитора. Если действия управляющей программы записать в виде совокупности блоков

⟨1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14⟩,

то обработку предложений с другими сервисными кодами можно представить следующим образом.

Сервисный код CK=X'8000' (оператор V): ⟨1, 5, 6, 7, 8, 9, 10, 14⟩; CK=X'8080' (SA): ⟨1, 5, 9, 10⟩; CK=X'8100 (VP): ⟨1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14⟩; CK=X'8300' (ZAPOM): ⟨1, 5, 6, 7, 8, 14⟩;CK=X'8400' (UDAL): ⟨1, 13, 14⟩

Рассмотрим подробнее реализацию операторов запоминания и удаления матриц. Получив предложение с СК= =X'8300', монитор динамически распределяет оперативную память в области данных. При этом используются два стека (рис. 5.2).

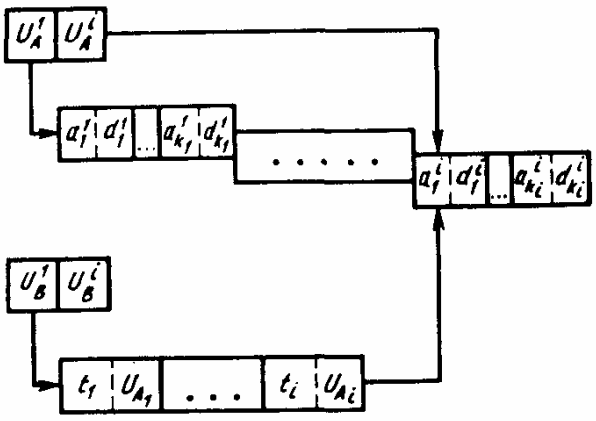


Рис. 5.2. Стеки для запоминания и удаления матриц

Стек A заполняется элементами a_i^j для каждой из матриц S_j^i, указанной в i-м операторе запоминания. Элемент стека содержит абсолютный адрес дескриптора матрицы и дисковый адрес, взятый из байтов 4—7 этого дескриптора. Дисковый адрес в дескрипторе заменяется адресом начала матрицы в области данных. Указатель стека содержит два значения: $U_A{}^1$ — адрес начала стека; $U_A{}^i$ — адрес начала первого свободного элемента в стеке.

Каждый элемент стека B соответствует одному оператору (блоку) запоминания. Элемент содержит адрес области данных t_i, начиная с которого производится выделение памяти для сохраняемых матриц, и значение указателя стека A для данного оператора запоминания ($U_A{}^i$). Указатель стека B содержит два значения: $U_B{}^1$ — адрес начала стека, $U_B{}^i$ — адрес первого свободного элемента в стеке.

Область данных заполняется с конца. Если в начале работы адрес конца области равен t_1, то после запоминания матриц из первого оператора ZAPOM он будет равен $t_2 = t_1 - \sum_{j=1}^{k_1}(m_j n_j + 1)$,

где k_1 — число сохраняемых матриц. Для i-го оператора он будет равен t_i. Именно с этого адреса будет выделяться память для размещения предложения выходного языка, исходных данных, вызываемых подпрограмм и сохраняемых в дальнейшем матриц.

Чтобы освободить область данных от запомненных матриц, используется оператор UDAL. В нем указывается число уничтожаемых блоков запоминания (l). Указатель стека U_B уменьшается на это число. Из элемента стека B выбираются значения адреса конца области данных (t_{i-l}) и указателя стека (U_A^{i-l}). Во всех дескрипторах, адреса которых ранее были сохранены в стеке A, начиная с указателя U_A^{i-l}, происходит восстановление дисковых адресов вместо адресов в ОП. После этого размещение матриц в области данных будет проводиться с адреса t_{i-l}. Стеки будут заполняться, начиная с U_A^{i-l}, U_B^{i-l}. Если ошибочно будет указано число удаляемых блоков запоминания больше, чем их существует в стеке, то значения указателей будут равны $U_A{}^1$, $U_B{}^1$, t_1.

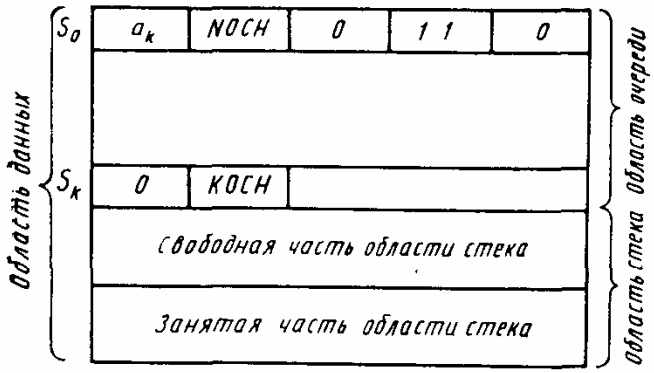


Рис. 5.3. Структура ОП в области данных в режиме очереди

Если в операторе удаления указываются матрицы, подлежащие выводу в рабочий набор данных, в стеке A отыскиваются их дисковые адреса и по ним производится вывод матриц.

Важной функцией монитора является автоматическое распределение памяти (см. гл. 3) при обработке блочных матриц. Этот режим работы включается предложением входного языка с СК=X′8500′ (оператор NOCH). Обнаружив этот код, монитор выделяет в верхней части области данных область очереди, ограничивая ее элементами S_0 и S_k односвязного списка S, оставляя для стековой части области данных ранее распределенную память плюс указанное в предложении число Кбайт (рис. 5.3). При этом устанавливается особый (автоматический) режим работы.

В автоматическом режиме отличие в обработке предложения входного языка заключается в работе блоков 3, 7 (размещение матриц) и блоков 11, 12, 13 (удаление матриц) (см. рис. 5.1). Монитор считает, что при работе в автоматическом режиме дескрипторы элементов состоят из трех слов. Структура первых двух слов та же, а в третьем слове может быть адрес, показывающий размещение дескриптора обрабатываемого блока в метаматрице (для блока блочной матрицы) или ноль. Во втором случае работает обычный алгоритм размещения и удаления, описанный выше, и матрица размещается в свободной части области стека. В первом случае размещение матрицы производится путем создания нового элемента S_i в списке S:

$$S_i = \langle a_i,\ d_i,\ p_i,\ b_i,\ m_i,\ n_i,\ t_i \rangle,$$

где a_i — адрес следующего элемента S_{i+1}; d_i — 2-е слово дескриптора размещаемого блока; p_i — признак «активности» блока; b_i — адрес дескриптора блока в метаматрице; m_i, n_i — размеры матрицы; t_i — элементы матрицы.

Длина элемента составляет $8(m_i n_i + 2)$ байтов.

Признак «активности» устанавливается, если блок выступает по крайней мере однажды в роли элемента результата (нетерминал ⟨отн адрр⟩) в операторах с СК=X′8100′ или X′8200′.

Размещение матриц в списке производится с помощью указателя списка, значение которого равно адресу последнего сформированного элемента. При переполнении очереди значение указателя становится равным адресу S_0, начинается новое заполнение. Новые элементы вытесняют ранее размещенные элементы списка. При этом восстанавливаются дисковые адреса в элементах мета-

матрицы, соответствующих удаляемым из памяти блокам. «Активные» блоки будут выведены в рабочий набор.

Такой алгоритм распределения ОП дает возможность программисту найти удачную реализацию любого алгоритма, комбинируя метод стека и очереди.

Ликвидация очереди происходит по СК=X'8600' (оператор КОСН). Для всех элементов очереди S_i производится восстановление дисковых адресов d_i в полях с адресами b_i+4 и вывод «активных» блоков в рабочий набор. Восстанавливается обычный режим работы монитора.

Последний сервисный код, обрабатываемый монитором, X'A000' используется при выдаче диагностических сообщений. Предложения с таким СК вырабатываются либо оператором DIAG, либо блоком диагностики оператора PROG в случае программного прерывания. При выполнении оператора начала основной программы происходит обращение к супервизору ОС ЕС с помощью макрокоманды SPIE. В результате этого при возникновении программного прерывания задача не завершается сразу аварийно, а происходит обращение к монитору с СК X'A000'.

Монитор подготавливает список параметров для модуля SMPODIAG, загружает его и передает ему управление. Модуль анализирует причину сбоя либо по коду, переданному из оператора DIAG, либо по старому слову состояния программы (PSW). Модуль SMPODIAG выводит необходимую для поиска причины ошибки информацию и завершает выполнение задания.

5.3. ОРГАНИЗАЦИЯ ДОЛГОВРЕМЕННОГО ХРАНЕНИЯ ДАННЫХ

Пользователь ППП СМПО имеет возможность выводить результаты расчета на диски и ленты для долговременного хранения с последующим их восстановлением.

Рассмотрим структуру библиотечного набора данных, используемого для хранения матриц на дисках.

Матрицы хранятся в библиотеке по именам. Каждая матрица занимает один раздел, причем имя раздела совпадает с именем матрицы. В каждом элементе оглавления библиотеки матриц [28] вместе с именем раздела хранятся его относительный дисковый адрес и четыре байта, содержащие размеры матрицы в виде двух полуслов. Блочные матрицы идентифицируются признаком X'8' в старшем полубайте первого размера. Оглавление библиотеки можно распечатать с помощью утилиты ОС IEHLIST [28].

Простая матрица, хранящаяся в разделе библиотеки, размещается в таком же формате, что и в рабочем наборе данных. Зная из элемента оглавления размеры матриц и ее дисковый адрес, всегда можно ее восстановить.

Блочная матрица хранится в разделе библиотеки в специальном формате. Сначала располагается образ метаматрицы, чьи размеры указаны в элементе оглавления. В отличие от метаматриц, расположенных в рабочем наборе данных и содержащих в дескрип-

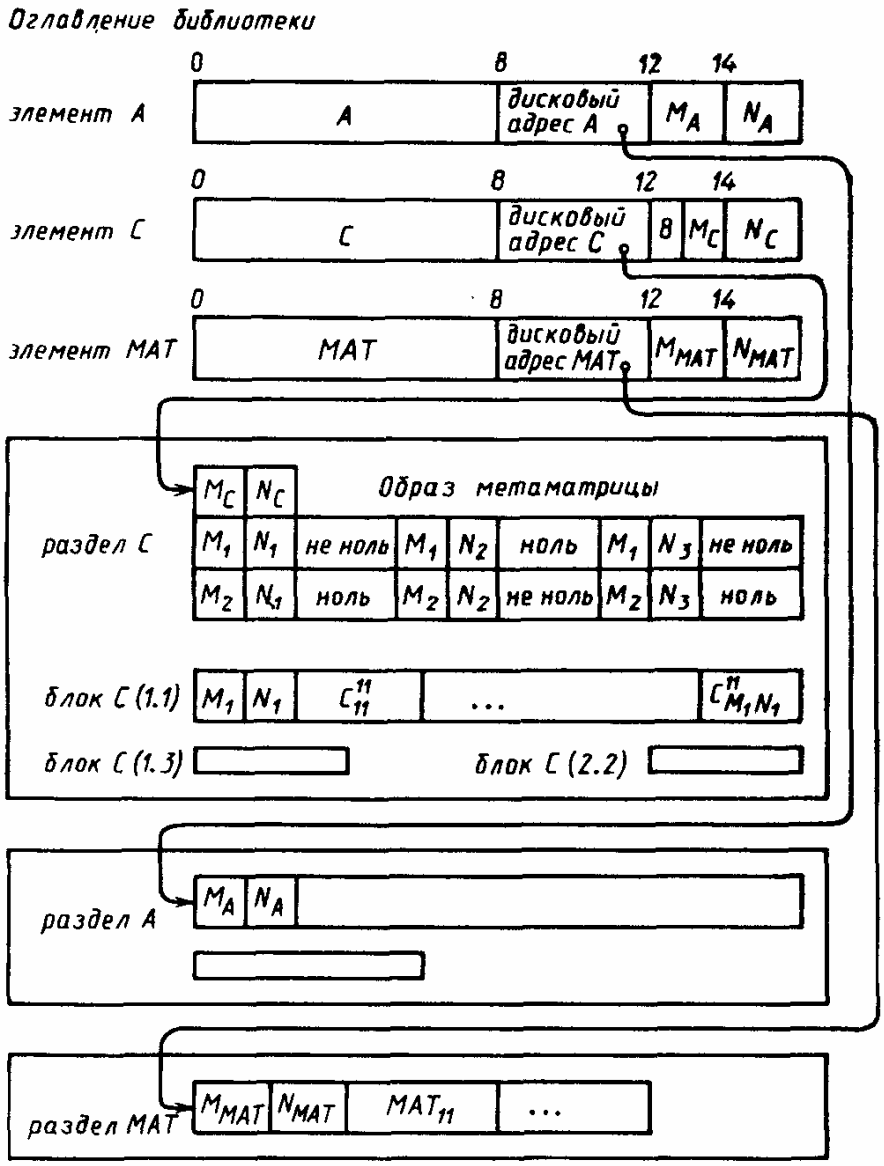


Рис. 5.4. Организация хранения матриц в библиотечном наборе данных

торе каждого блока, кроме его размеров, реальный дисковый адрес, образ метаматрицы из раздела библиотеки содержит не дисковые адреса, а лишь указание на то, существует блок или нет. Существующие блоки располагаются сразу за образом метаматрицы в том порядке, в котором они в ней перечислены. На рис. 5.4 показан пример библиотеки, содержащей три раздела. В разделах A и MAT хранятся простые матрицы, в разделе C — блочная матрица, которая имет 3 ненулевых блока, располагающихся один за другим. Каждая метаматрица, блок или простая матрица размещаются в виде одной или нескольких физических записей. Так, матрица A занимает 2 записи, так как ее размер больше длины дорожки.

Принятый в СМПО способ сохранения блочной матрицы не позволяет легко найти один блок, но это и не требуется — матрицу обычно нужно восстанавливать целиком. Если бы образ метаматрицы содержал дисковые адреса блоков, библиотека стала бы не-

сжимаемой и неперемещаемой. Ее нельзя было бы обрабатывать утилитами ОС ЕС.

Сохранение и восстановление матриц на диске осуществляется с помощью операторов SOXD и VOSD, содержащих имена матриц и DD-имя библиотечного набора данных [28]. Число библиотек, обрабатываемых в задаче пользователя, произвольно. При организации библиотеки пользователю не нужно указывать ее DCB-параметры. Они будут сформированы автоматически: записи будут иметь формат неопределенной длины с максимальным размером, равным длине дорожки диска.

Сервисные программы пакета, выполняющие сохранение и восстановление матриц из библиотек, контролируют правильность выполнения операций ввода-вывода. При обнаружении ошибки как в оглавлении библиотеки, так и в ее разделах, причина сбоя диагностируется, а решение задачи пользователя прекращается.

Рассмотрим организацию хранения матриц на магнитной ленте. Сохранение матриц осуществляется оператором SOXL, восстановление — VOSL. Матрицы хранятся в виде последовательного набора данных. В начале набора располагается каталог, описывающий его содержимое, затем идут сами матрицы.

Программы работы с лентами не зависят от наличия или отсутствия меток, а также порядкового номера набора на ленте. Информация об этом указывается в DD-операторе языка управления заданиями [28]. На одном томе ленты может располагаться несколько наборов данных. Необходимо помнить, что если какой-то набор переписывается заново, то все последующие наборы становятся недоступными для обработки.

При сохранении матриц возможна постоянная ошибка ввода-вывода. В этом случае оператор ЭВМ получит сообщение ОШИБКА ПРИ ЗАПИСИ НА ЛЕНТУ. ДЛЯ ПРОДОЛЖЕНИЯ РАБОТЫ ОТВЕТЬТЕ ДА. Если оператор ответит НЕТ, то задача будет снята. Если ДА, то решение будет продолжено, и вслед за информацией о причине ошибки в распечатке появится сообщение ВЫВОД НА ЛЕНТУ ПРЕКРАЩЕН. РАБОТА ПРОДОЛЖАЕТСЯ. Эта возможность особенно удобна, когда для гарантии сохранение выполняется несколько раз. Ошибка ввода-вывода при восстановлении матриц приводит к аварийному завершению задачи.

Глава 6. Входной язык ППП СМПО

Входной язык ППП СМПО — совокупность операторов, позволяющих осуществлять программирование матричных алгоритмов с использованием возможностей пакета [2]. Операторы представляют собой макросы, расширяющие язык ассемблер. В понятие «макрос» входят: макроопределение, макровызов, макрорасширение [38]. Макроопределение имеет следующую структуру:

MACRO

оператор прототипа

тело макроопределения

MEND

Команды MACRO и MEND являются соответственно первым и последним операторами макроопределения. Оператор прототипа определяет формальный вид макровызова. Любая макрокоманда (макровызов) в программе пользователя должна иметь имя и формат, установленный оператором прототипа. Оператор прототипа имеет следующий формат:

Название	Операция	Операнды	Комментарий
Параметр или пробел	Имя макроса	Список операндов или пробел	Текст

Имя макроса — идентификатор, содержащий не более восьми букв и цифр. Параметр состоит из знака амперсенда &, за которым следует начинающаяся с буквы последовательность, состоящая не более чем из семи букв и цифр. Список операндов может содержать до 100 операндов, разделенных запятыми. Операндом может быть список параметров или одиночный параметр.

Список — заключенная в скобки совокупность параметров, отделенных друг от друга запятыми. Признаком окончания оператора прототипа является пробел после операнда и отсутствие указателя продолжения. Комментарий должен отделяться от последнего параметра хотя бы одним пробелом. Тело макроопределения представляет собой последовательность команд ассемблера, которая определяет прототип текста, вставляемого в программу вместо макровызова. Все макроопределения находятся в специальной макробиблиотеке в форме исходных модулей. Операторы, встречающиеся в программе пользователя, являются макровызовами (макрокомандами) и обрабатываются макропроцессом ассемблера. По макровызову в библиотеке отыскивается соответствующее макроопределение. По нему и аргументам макрокоманды генерируется макрорасширение и подставляется в программу вместо оператора.

Рассматриваемые операторы пакета являются в основном позиционными. В них операнды должны записываться в строго фиксированном порядке, который определяется оператором прототипа. Вместо пропущенного операнда, если он не последний, в макрокоманде должна ставиться запятая. Параметры операторов, начинающиеся со знака &, являются формальными и должны заменяться на фактические при конкретном использовании оператора. При отсутствии специального параметра в поле названия это поле может быть использовано для записи метки оператора. Ключевые параметры, отличительным признаком которых служит знак равенства

в поле операндов, могут перечисляться в любом порядке, но они должны располагаться после позиционных. Отсутствие ключевого параметра не нужно указывать запятой.

Использование при описании форматов квадратных скобок указывает на необязательность указанного в них текста.

Операторы СМПО делятся на исполняемые и неисполняемые. Исполняемые операторы должны быть указаны в соответствии с порядком их выполнения и могут чередоваться с машинными командами. Неисполняемые операторы в СМПО обычно располагаются в конце программы в любом порядке и могут чередоваться с командами ассемблера, описывающими константы и выделяющими память.

Среди исполняемых операторов есть операторы прямого действия и операторы обращения. Операторы прямого действия выполняют свои функции непосредственно, без помощи монитора и служат в основном для обеспечения стенографической записи, т. е. позволяют пользователю писать короткую последовательность символов вместо длинной. Операторы обращения передают нужную информацию монитору, выполняющему требуемые действия.

Если пользователю пакета не нужна печать макрорасширений, получаемых макроассемблером при расшифровке оператора СМПО, он может в начале программной единицы закодировать команду ассемблера:

```
PRINT NOGEN
```

6.1. ОПЕРАТОРЫ ОФОРМЛЕНИЯ ПРОГРАММНЫХ ЕДИНИЦ

Основная программа должна начинаться оператором PROG.

Оператор PROG является исполняемым оператором и имеет следующий формат:

```
& NAME      PROG  [&KB] [,RZ = &RZ] [,D1 = & D1]
```

где & NAME — название программы, начинающееся с буквы и содержащее не более 8 алфавитно-цифровых символов; & KB — число элементов стека дисковой памяти, равное наибольшему количеству операторов OPR, между которыми нет операторов KBL и STEK. По умолчанию размер стека принимается равным 50 элементам; & RZ — число, в соответствии с которым распределяется оперативная память в разделе, выделенном для выполнения программ СМПО. Из общей памяти раздела, переданной в основную программу СМПО через поле PARM оператора EXEC языка управления заданиями, вычитается величина & RZ. Полученное значение принимается в качестве размера области данных в килобайтах. В оставшейся области ОП размещаются программы СМПО и ОС. Таким образом, параметр & RZ определяет размер области СПЕ. По умолчанию величина & RZ принимается равной 40; & D1 — число дорожек, выделенных на первом томе для рабочего набора данных. Параметр должен кодироваться только при размещении рабочего набора на двух накопителях на магнитных дисках.

Если при кодировании оператора PROG указан ключевой параметр D1, это означает, что пользователь использует размещение рабочего набора данных на двух накопителях на магнитных дисках. Размер первой части набора должен быть указан дважды: в параметре &D1 оператора PROG (в дорожках) и в параметре VEL=*ррн* оператора языка управления заданиями EXEC SMPOTRV или EXEC SMPOV (в цилиндрах). Эти величины обязательно должны соответствовать друг другу. Так, для дисков типа ЕС-5050 должно выполняться соотношение &D1=*ррн*·10; для дисков типа 5061 — &D1=*ррн*·20; для дисков типа 5066 — &D1= =*ррн*·19. Вторая часть набора должна быть размещена на диске того же типа, что и первая. Ее нужно описать DD-оператором языка управления заданиями с DD-именем DISKSMP1, который должен быть добавлен после исходных данных:

//GO. DISKSMP1 DD UNIT=SYSDA, SPACE=(CYL, (*ррн2*, *прн2*))

Здесь *ррн2* — размер второй части рабочего набора данных в цилиндрах; *прн2* — дополнительное число цилиндров (приращение), выделяемое на диске, если будет исчерпана отведенная под рабочий набор память. Такое дополнительное выделение при наличии неиспользуемой памяти на диске может быть выполнено до 15 раз.

При необходимости запрос диска может быть сделан специальным, т. е. может быть добавлен параметр VOL=SER=*имя*, где *имя* — имя тома, где будет расположена вторая часть набора.

Пример 6.1.1.

Пусть требуется разместить рабочий набор на двух дисках типа ЕС-5061. На первом диске выделяется 190 цилиндров (3800 дорожек) на втором — сразу 80 цилиндров, а затем дополнительно возможны добавления по 10 цилиндров

```
// EXEC SMPOTRV, VEL=190, R = 500
DVA      PROG  D1=3800
```

текст основной программы

```
//GO.SYSIN DD*,DLM='//'
```

исходные данные

```
//GO. DISKSMP1 DD UNIT = SYSDA, SPACE = CYL, (80, 20))
```

В результате работы оператора PROG инициализируются наборы данных пакета, подключается его монитор, выделяется память под область данных. Затем печатается информационная строка пакета, содержащая номер версии ППП и дату ее создания. Далее выводятся сообщение НАЧАЛО ОСНОВНОЙ ПРОГРАММЫ DVA и текущее время.

Исполняемая часть основной программы должна завершаться оператором KPROG.

Формат оператора KPROG:

KPROG

Оператор выводит на печать сообщение КОНЕЦ ОСНОВНОЙ ПРОГРАММЫ & NAME, текущее время и статистическую информацию о прохождении задачи (см. оператор STAT). После этого выполнение основной программы завершается с кодом условия 0.

Таким образом, основная программа должна иметь такой вид, как в примере 6.1.2.

Пример 6.1.2.

PRIMER PROG

исполняемые операторы СМПО и машинные команды.

KPROG

неисполняемые операторы СМПО, константы и поля

END

В основной программе можно использовать зарезервированное слово NET, которое является именем нулевого двойного слова, генерируемого оператором PROG.

Обращения к некоторым стандартным алгоритмам содержат имя NET.

СА должен начинаться оператором STAL.

Оператор STAL является исполняемым оператором и имеет следующий формат:

&NAME STAL &SP

где & NAME — название СА, содержащее не более четырех алфавитно-цифровых символов и начинающееся с буквы; & SP — список формальных параметров СА, представляющий собой один или несколько позиционных параметров — элементов списка, отделенных друг от друга запятыми и заключенных в круглые скобки.

Количество и порядок следования формальных параметров должны соответствовать количеству и порядку следования фактических параметров. Параметры должны также соответствовать по типам.

Формальные параметры должны быть описаны в СА, т. е. для них должна быть выделена память в неисполняемой части программы. Для динамических матриц и чисел с плавающей точкой двойной точности выделяются двойные слова, для сложных списков, чисел с плавающей точкой обычной точности, целых чисел в формате F — полные слова, целых чисел в формате H — полуслова. Такое выделение может быть выполнено с помощью операторов GDS или команд ассемблера DS.

Оператор STAL инициализирует выполнение СА и производит подстановку фактических параметров на место формальных.

Исполняемая часть СА должна завершаться оператором KSTAL.

Формат оператора KSTAL:

KSTAL [&SPV]

где &SPV — список параметров, передаваемых не только по значению, но и по результату. Он представляет собой один или несколько параметров, отделенных друг от друга запятыми и заключенных в круглые скобки. Если в списке присутствует один параметр, брать его в скобки необязательно.

В список &SPV не нужно включать матрицы, дескрипторы которых не изменяются, потому что они и так передаются не только по значению, но и по результату.

В списке &SPV могут перечисляться только формальные параметры СА, т. е. параметры списка &SP оператора STAL. В операторе KSTAL они могут быть перечислены в любом порядке.

Оператор KSTAL завершает выполнение СА и передает значения указанных в нем параметров в вызывающую программу. Если стандартный алгоритм не выполнился указанное число раз, оператор KSTAL снова передает управление оператору STAL, вызывая повторное выполнение СА с новыми данными, в противном случае передается управление вызывающей программе.

После оператора KSTAL располагается неисполняемая часть СА, текст которой должен завершаться оператором KSTP.

Оператор KSTP является неисполняемым оператором и имеет следующий формат:

KSTP

Вслед за оператором KSTP должна располагаться команда ассемблера END, кодируемая без параметров.

Рассмотрим пример кодирования СА.

Пример 6.1.3.

```
SALG     STAL  (A,B,C)
. . . . . . . . . . . . . . .
         MVC   C,=D'1'
         MVC   A,=D'2'
         KSTAL (C,A)
D        GDS   (A,B,C)
         KSTP
         END
```

В примере приведен СА с именем SALG, имеющий три параметра — A, B, C. Параметры A и C получают новые значения в алгоритме (A=2, C=1) и передаются оператором KSTAL в вызывающую программу.

В отличие от основной программы при необходимости использовать в СА слово NET его нужно описать после оператора KSTAL следующим образом:

```
NET      DC    D'0'
```

В этом случае слово NET можно использовать при вызове других алгоритмов.

Стандартная программа должна начинаться оператором STP.

Оператор STP является исполняемым оператором и имеет следующий формат:

```
&NAME    STP   [&R]
```

где &NAME — название СТП, содержащее не более четырех алфа-

витно-цифровых символов и начинающееся с буквы; & R — номер базового регистра, который может принимать значения от 2 до 12. Если параметр & R опущен, в качестве базового будет использован регистр 15. Если СТП содержит один из операторов VVOD, TEKST, CON, OTPE, PTIMER, UTIMER или макрокоманды ОС ЕС, параметр & R обязательно должен быть указан.

После выполнения оператора STP стандартной программе доступен список адресов фактических параметров. Адрес этого списка находится в регистре 1. Список может содержать переменное число параметров; последний параметр списка содержит в старшем байте признак конца — шестнадцатеричную константу X'80'. Так как параметры передаются по адресу, формальные параметры непосредственно совпадают с фактическими. СТП может изменять значение только последнего параметра, не меняя значений предыдущих.

Если одним из параметров СТП является матрица, то независимо от того, запомнена она или нет, монитор пакета обеспечит ее размещение в ОП на время выполнения операции.

Рассмотрим примеры передачи параметров в СТП.

Пример 6.1.4

Пусть в СТП CP1 передается два параметра. Тогда ее начало может иметь следующий вид

```
CP1     STP
        LM      2, 3, 0(1)
```

Теперь в регистре 2 содержится адрес первого параметра, а в регистре 3 — второго.

Пример 6.1.5

Пусть СТП CP2 должна выполнять сложение чисел с плавающей точкой, причем число слагаемых может быть произвольным. В этом случае список параметров будет иметь переменную длину. Последний элемент списка будет содержать адрес результата.

```
CP2     STP
        SDR     0, 0 чистим сумматор
CIKL    L       2, 0(1) регистр 2 — адрес параметра
        CLI     0(1), X'80' конец списка?
        BE      REZ да переход к REZ
        AD      0, 0(2) сложение
        LA      1, 4(1) обработаем
        B       CIKL следующий параметр
REZ     STD     0, 0(2) формируем результат
```

В этом примере в цикле выполняется просмотр списка параметров. Входные параметры суммируются. При обнаружении конца списка формируется результат.

Исполняемая часть СТП должна завершаться оператором RETURN, который имеет следующий формат

```
        RETURN
```

Оператор RETURN завершает выполнение СТП и передает управление монитору пакета, который возвращает его вызывающей программе.

После оператора RETURN располагается неисполняемая часть СП, текст которой должен завершаться оператором KSTP, за которым должна располагаться команда ассемблера END, кодируемая без параметров.

Таким образом, чтобы завершить приведенную в примере 6.1.5 программу, к ней нужно дописать следующий фрагмент:

```
        RETURN
        KSTP
        END
```

6.2. ОПЕРАТОРЫ ОПРЕДЕЛЕНИЯ И РАЗМЕЩЕНИЯ МАТРИЦ. РАЗМЕРЫ МАТРИЦ

Для определения матриц, которые могут обрабатываться программными средствами ППП СМПО, должны использоваться операторы SMATR, OPR, MATR.

Оператор SMATR определяет статическую матрицу и имеет следующий формат:

```
&A      SMATR &S1, &S2
```

где &A — имя матрицы; &S1 — операнд, задающий размеры матрицы в виде списка (&M, &N); &S2 — список значений элементов матрицы.

Элементы матрицы задаются по строкам в виде чисел с плавающей точкой. Матрица с указанными элементами строится на этапе трансляции. Первый байт поля, содержащего размеры матрицы, идентифицируется именем &A. Каждый элемент матрицы идентифицируется именем &AL, где L — порядковый номер элемента. Оператор является неисполняемым и должен располагаться в области констант и полей.

При трансляции оператора SMATR возможно диагностическое сообщение РАЗМЕРЫ НЕ ПОЛОЖИТЕЛЬНЫ, если &M$\leqslant$0 или &N$\leqslant$0.

Пример 6.2.1

Оператор M SMATR (2, 3), (0, 1, 2, — 3.1, 4.22, 5) определяет поля:

```
M       DC      H'2, 3'
M1      DC      D'0'
M2      DC      D'1'
M3      DC      D'2'
M4      DC      D' — 3.1'
M5      DC      D'4.22'
M6      DC      D'5'
```

для матрицы M размерами 2×3 с элементами $M_{1,1}=0$; $M_{1,2}=1$; $M_{1,3}=2$; $M_{2,1}=3{,}1$; $M_{2,2}=4{,}22$; $M_{2,3}=5$.

Операторы OPR и MATR служат для создания динамически размещаемых матриц и имеют идентичный формат:

&IM &S1,..., &SI,..., &SN

где &IM — имя оператора, равное OPR или MATR; &SI — операнд (I=1, ..., N).

В качестве операнда &SI могут выступать: $\&A_i$ — имя матрицы; ($\&K_i$, $\&A_i$) — список, эквивалентный перечислению $\&K_i$ матриц с именами $\&A_i1$, $\&A_i2$, ..., $\&A_i\&K_i$; ($\&A_{i1}$, ..., $\&A_{i,n_i}$) — список имен матриц $\&A_{i1}$, ..., $\&A_{i,n_i}$; ($\&M_i$, $\&M_i$) — размеры матрицы или группы матриц S_{i-1}.

При трансляции каждого из операторов OPR или MATR строится каталог матриц, состоящий из восьмибайтовых элементов — дескрипторов матриц, идентифицированных указанными именами. Имена матриц должны быть уникальными в пределах данной программной единицы.

Размеры матриц могут вноситься в дескрипторы на этапе трансляции, если они явно указаны в операторе, на этапе выполнения — с помощью оператора ZR или вводиться с перфокарт. Операторы являются исполняемыми.

Явно указанные размеры должны следовать за операндом &SI в следующем операнде.

В этом случае все матрицы, указанные в операнде &SI, имеют одни и те же размеры ($\&M_{i+1}$, $\&N_{i+1}$).

Пример 6.2.2

OPR A, (2, 3), (B, C, D), (3,4), (2, M), (5,5), F

Матрицы, указанные в операторе, имеют следующие размеры: A — размеры 2×3; B, C, D — размеры 3×4, M1, M2 — размеры 5×5; F — размеры не определены.

Внесение размеров матриц может быть осуществлено на этапе выполнения программы оператором ZR (занести размер). Оператор ZR является исполняемым и должен быть выполнен раньше оператора OPR (MATR).

Оператор ZR имеет следующий формат:

ZR &R, (&M1,..., &MI,..., &MN)

где &R — номер общего регистра (1—9), который содержит значение размера; &MI — обозначение имени матрицы, в котором непосредственно перед именем матрицы может располагаться цифра 1 или 2, соответствующая первому или второму размеру матрицы. Если цифра опущена, предполагается первый размер. При N=1 скобки могут быть опущены.

Пример 6.2.3

ZR 4, (2A, 1B, 2B, D)

```
ZR    5 ,A
OPR   A, B, D,( 0,3)
```

Здесь содержимое общего регистра 4 будет использовано в качестве второго размера A, обоих размеров B и первого размера D, при замене указанного в OPR нулевого значения. В качестве первого размера A будет использовано содержимое регистра 5.

Если в момент выполнения оператора OPR (MATR) первый размер первой из определяемых матриц равен нулю, т. е. не указан в операторе OPR (MATR) и не занесен оператором ZR, считается, что информация о размерах всех матриц оператора расположена среди исходных данных и будет введена с перфокарт. В этом случае игнорируются ранее сформированные размеры других матриц оператора.

Размеры набиваются на картах через пробел в позициях 1—78 в виде

$r_1\ r_2\ r_j$

где r_j — обозначение размеров в виде $m_j : n_j$ или $l_j * m_j : n_j$. Вторая запись эквивалентна повторению размеров $[m_j \times n_j]\ l_j$ раз.

При наличии двух и более пробелов подряд читается следующая карта. Ввод прекращается при обнаружении карты конца файла (/*). При вводе контролируется соответствие количества размеров числу матриц оператора.

Формирование размеров производится одинаково для операторов OPR и MATR. Дальнейшие их действия существенно отличаются друг от друга. Оператор OPR создает нулевые матрицы в рабочем наборе данных. Оператор MATR не создает никаких матриц, но в дальнейшем позволяет выделить для них место в области данных ОП. Обращаться к матрицам, объявленным оператором MATR, можно только после того, как они будут запомнены оператором ZAPOM.

Начальный дисковый адрес, с которого происходит размещение указанных в операторе OPR матриц, берется из стека дисковой памяти. Обращение к верхушке стека производится через его указатель, адрес которого находится в системной области сохранения. Дисковые адреса в стеке хранятся в виде $NTTR$, где $N=0$ для рабочего набора, расположенного на одном томе, $N=1$ для рабочего набора на втором томе; TT — относительный номер дорожки в наборе; R — номер записи на дорожке. После работы i-го оператора OPR в стеке появляется новый $(i+1)$-й элемент, содержащий дисковый адрес последней записи в занятой области набора данных. При этом указатель стека увеличивается и показывает на этот новый элемент (рис. 6.1).

В процессе определения матриц возможны следующие диагностические сообщения:

НЕДОПУСТИМЫЙ СИМВОЛ — если при вводе размеров с перфокарт обнаружен символ, отличный от «цифры», «*», «:», «пробела», «/*»;

НЕВЕРНЫЙ СПИСОК ПАРАМЕТРОВ — если количество фактически определенных (имеющих размеры) матриц не равно количеству указанных в операторе OPR;

НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ — если один из размеров определяемой матрицы равен 0;

СТЕК БЛОКОВ МАТРИЦ ЗАПОЛНЕН — размер стека дисковой памяти, равный параметру & KB — количеству незакрытых операторов OPR в операторе PROG, оказался недостаточным;

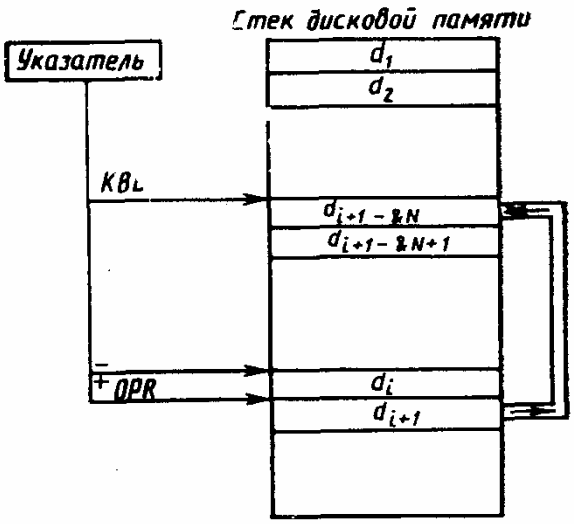


Рис. 6.1. Состояние стека дисковой памяти

НЕПРАВИЛЬНЫЙ ЗАПРОС В/В. МОНИТОР ВЫДЕЛЯЕТ ДОП. ПАМЯТЬ НА ДИСКЕ — размер рабочего набора данных, заданный в операторе языка управления заданиями EXEC SMPOTRV или EXEC SMPOV, мал. Если на диске есть свободное место, решение будет продолжено, такое выделение может быть выполнено до 15 раз.

Закрытие открытых ранее блоков стека дисковой памяти, сводящееся фактически к уничтожению определенных в этих блоках матриц, выполняется оператором:

KBL & N

где & N — число закрываемых блоков (операторов OPR).

При выполнении оператора KBL, который стоит между i-м и $(i+1)$-м операторами OPR, указатель стека блоков уменьшается на & N·4 байтов; это приводит к тому, что $(i+1)$-й оператор OPR будет использовать в качестве начального адреса дисковый адрес $d_{i+1-\&N}$ и завершит свое выполнение заполнением элемента стека под номером $(i+1)$—& N+1 (см. рис. 6.1).

Может возникнуть ситуация, когда необходимо уменьшить количество занятых элементов стека блоков без уничтожения определенных ранее матриц. Это можно осуществить оператором

STEK &N

где &N — число элементов, на которое уменьшается стек блоков.

При выполнении оператора STEK после i-го оператора значение указателя стека уменьшается на & N·4 байтов, т. е. указатель устанавливается на элемент $(i+1$—& N) стека блоков. В этот элемент переносится дисковый адрес d_{i+1} взятый из $(i+1)$-го элемента стека (см. рис. 6.1).

Пример 6.2.4

Блок 1 [.....

```
Блок 2  ┌OPR   A, (4, A)
        │...                                  (6.1)
        └KBL   1
Блок 3  ┌OPR   B, BO, (3, B)
        │...                                  (6.2)
        └STEK  1
```

Определяются матрицы A, A1, A2, A3, A4 в программном блоке 2. После того, как этот блок будет закрыт (6.1), матрицы B, B0, B1, B2, B3 будут размещены на диске с того же адреса, что и матрицы A—A4, причем последние матрицы будут уничтожены. Оператор (6.2) уменьшил значение указателя стека и, по существу, перенес матрицы B—B3 из блока 3 в блок 1. Произошло объединение блоков 1 и 3. После этого становится невозможным уничтожение только матриц B—B3, не затрагивая при этом тех матриц, которые были определены в блоке 1.

Обратным по отношению к оператору занесения размеров (ZR) является оператор извлечения размеров (IR).

Оператор применяется для определения размеров ранее определенных матриц. Он является исполняемым и имеет следующий формат:

```
IR   (&RI,... &RI,..., &RN),
     (&MI,..., &MI,..., &MN)
```

где &RI — номер общего регистра — (1—9); &MI — обозначение имени матрицы, в котором непосредственно перед именем матрицы может располагаться цифра 1 или 2, обозначающая первый или второй размер матрицы. Если цифра опущена, предполагается первый размер. При N = 1 скобки могут быть опущены.

В результате выполнения оператора IR в регистр &RI будет загружено значение соответствующего размера матрицы &MI.

Пример 6.2.5

Рассмотрим использование операторов обработки размеров. Пусть имеются матрицы A[M×N] и B (1×K). Требуется определить матрицы C (1×2M), D [K×N] и объявить оператором MATR матрицы E (10×12), F[N×M]. Требуется также определить матрицу NEI, размеры которой будут набиты на перфокартах.

```
IR     (2, 3, 4), (A, 2A, 2B)      (6.3)
ZR     2, 2F                       (6.4)
ZR     3, (2D, F)                  (6.5)
ZR     4, D                        (6.6)
AR     2, 2                        (6.7)
ZR     2, 2C                       (6.8)
OPR    C, (1, 0), D                (6.9)
MATR   E, (10,12), F               (6.10)
OPR    NEI                         (6.11)
```

Оператор (6.3) помещает в регистр 2 значение M, в регистр 3 — N, в регистр 4 — K. Операторы (6.4)—(6.6) заносят эти значения в матрицы F и D. Опе-

ратор (6.7) удваивает значение M, находящееся в регистре 2, а оператор (6.8) заносит это значение в качестве второго размера матрицы C. Оператор (6.9) определяет матрицы C и D, причем указанный в описании матрицы C второй нулевой размер к моменту выполнения оператора уже заменен на реальное значение в операторе (6.8). Оператор (6.10) объявляет матрицы E и F. Оператор (6.11) определяет матрицу NEI, размеры которой будут введены с карт во время выполнения программы.

С целью сокращения количества операций, осуществляющих обращение к магнитным дискам, ранее определенные матрицы можно запомнить в области данных с помощью оператора ZAPOM.

Оператор ZAPOM является исполняемым и имеет следующий формат:

ZAPOM (&A1,..., &AI,..., &AN)

где &AI — имя матрицы, определенной ранее в операторе OPR или MATR.

При выполнении этого оператора указанные в нем матрицы, определенные предварительно оператором OPR, вводятся из рабочего набора данных и размещаются в ОП. Для матриц, объявленных оператором MATR, просто выделяется участок памяти в области данных. В начале участка в соответствии с форматом матрицы в СМПО размещаются размеры матрицы, но сам участок не обнуляется и может содержать произвольную информацию, оставшуюся от предыдущих операций.

Все действия по обработке оператора ZAPOM выполняются монитором ППП СМПО.

После выполнения оператора ZAPOM изменяются дескрипторы перечисленных в нем матриц. Вместо дискового адреса (для матриц, определенных по OPR) или признака несуществующей матрицы (для матриц, объявленных по MATR) теперь будет располагаться адрес матрицы в ОП, указывающий на ее размеры. Это можно использовать для непосредственной обработки запомненной матрицы в СА или в основной программе с помощью машинных команд.

Пример 6.2.6

Пусть необходимо загрузить в регистр с плавающей точкой значение элемента A (1.1). Матрица A запомнена.

```
L     1, A+4        (6.12)
LD    0, 4(1)       (6.13)
```

Команда (6.12) помещает в 1-й регистр адрес матрицы A. Команда (6.13), пропуская размеры A, загружает в регистр 0 значение A (1.1).

Каждый оператор ZAPOM открывает новый блок запоминания. При необходимости память, занимаемая матрицами блока, может быть освобождена. Запомненные матрицы образуют вложенную структуру, так что пользователь имеет возможность удалить ранее запомненные матрицы, оставив запомненные позже. Группа матриц, запомненная последней, должна удаляться первой.

Для освобождения оперативной памяти, занимаемой запомненными ранее матрицами, и возможно, вывода новых значений мат-

риц в рабочий набор данных, используется оператор UDAL (удаление матриц из ОП).

Оператор UDAL является исполняемым и имеет следующий формат:

UDAL &N[,(&BI,..., &BJ,..., &BM)]

где &N — число блоков запоминания, т. е. операторов ZAPOM, действие которых отменяется. Вся ОП, занятая перечисленными в этих операторах матрицами, освобождается; &BJ — одна из матриц, определенная оператором OPR и указанная в списке матриц одного из отменяемых операторов ZAPOM. Значение этой матрицы будет выведено в рабочий набор. Порядок следования матриц &BJ в списке оператора UDAL произволен.

Таким образом, при выполнении оператора UDAL на диск выводятся только те матрицы, которые явно в нем указаны. Остальные матрицы закрываемых блоков запоминания просто вычеркиваются из области данных. При этом матрицы, объявленные оператором MATR, которые были запомнены, а потом удалены, снова являются недоступными для использования до тех пор, пока не будут снова запомнены.

Операторы ZAPOM и UDAL реализуют стековый принцип распределения памяти: матрицы, запомненные последними, удаляются первыми. Оператор ZAPOM создает новый элемент на верхушке стека, оператор UDAL выталкивает несколько (&N) элементов стека.

В процессе работы оператора UDAL возможно диагностическое сообщение УДАЛЯЕМАЯ МАТРИЦА НЕ НАЙДЕНА! Оно означает, что одна из матриц BJ не была обнаружена в составе тех блоков, которые подлежат закрытию. Обычно это происходит, когда нарушается вложенность, т. е. стековая структура действий операторов ZAPOM — UDAL. Например, это сообщение может появиться в том случае, когда матрица &BJ уже удалена предыдущим оператором UDAL либо находится в глубине стека, и для вывода ее на диск требуется закрыть большее число блоков, чем указал пользователь в операторе UDAL.

Все действия по обработке оператора UDAL выполняет монитор ППП СМПО. В процессе обработки дескрипторы всех матриц из закрываемых блоков запоминания (как перечисленные в операторе UDAL, так и неперечисленные) принимают первоначальный вид, т. е. тот вид, который они имели до оператора ZAPOM. При этом вместо адреса в ОП дескриптор удаленной матрицы снова будет содержать дисковый адрес или признак матрицы, объявленной оператором MATR.

Рассмотрим пример применения операторов ZAPOM и UDAL.

Пример 6.2.7

```
OPR     C, A
MATR    B, M1
ZAPOM   (A, B, C)                                    (6.14)
```

. . . . . .

OPR M2, M3

ZAPOM (M1, M2, M3) (6.15)

. . . . . .

UDAL 2, (C, M3, A) (6.16)

В примере оператор (6.16) освобождает ОП, занимаемую матрицами A, B, C, M1, M2, M3 (см. (6.14), (6.15)). Матрицы C, M3 и A выводятся в рабочий набор. В число указанных в (6.16) матриц нельзя включать матрицы B и M1, потому что они объявлены оператором MATR, а не OPR.

Оператор ZAPOM может применяться только к матрицам, которые не являются еще запомненными. Часто в стандартном алгоритме неизвестно, была ли запомнена перед обращением к СА передаваемая ему матрица или нет. В этом случае удобно воспользоваться оператором ZAPO.

Оператор ZAPO является исполняемым и имеет следующий формат:

ZAPO $(\&A_1, \ldots, \&A_i, \ldots, \&A_N)$

где $\&A_i$ — имя матрицы, которая будет запомнена оператором ZAPO, если она не запомнена к моменту его выполнения. Если матрица $\&A_i$ уже запомнена, оператор ZAPO не выполняет никаких действий по ее обработке. При $N=1$ скобки могут быть опущены.

В рамках одной СПЕ каждая матрица может появиться только в одном операторе ZAPO.

Разберем, как работает оператор ZAPO. Он просматривает список обрабатываемых матриц слева направо. Для каждой матрицы проверяется, не является ли она запомненной, и в специально отведенном поле сохраняется информация о ее состоянии до оператора ZAPO. Если матрица запомнена, никаких действий не выполняется. В противном случае формируется и выполняется оператор ZAPOM $(\&A_i)$.

Таким образом, создается столько блоков запоминания, сколько матриц из списка оператора ZAPO не были запомнены до его выполнения.

В паре с оператором ZAPO должен использоваться оператор UD. Он предназначен для освобождения памяти и вывода на диск матриц, которые были действительно запомнены оператором ZAPO. С теми матрицами, которые были запомнены до ZAPO, оператор UD никаких действий не выполняет.

Оператор UD является исполняемым и имеет следующий формат:

UD $(\&A_1, \ldots, \&A_i, \ldots, \&A_K), (\&A_{K+1}, \ldots, \&A_j, \ldots, \&A_N)$

где $\&A_i$ — имя матрицы, которая должна быть вычеркнута из ОП, если она не была запомнена до оператора ZAPO. При $K=1$ первая пара скобок может быть опущена, $\&A_j$ — имя матрицы, которая должна быть вычеркнута из ОП и выведена в рабочий набор

данных, если она не была запомнена до выполнения оператора ZAPO. При $N=K+1$ вторая пара скобок может быть опущена.

Матрицы в операторе UD должны перечисляться в том же порядке, что и в соответствующем операторе ZAPO. Если ни одна из матриц не должна быть выведена на диск, вторую пару скобок нужно опустить вместе со стоящей перед ней запятой; если все матрицы выводятся на диск, опускается первая пара скобок, а ее отсутствие указывает запятая.

Разберем, как работает оператор UD. Он просматривает списки справа налево. Сначала обрабатывается второй список. Используя информацию о состоянии матриц до ZAPO, созданную оператором ZAPO, оператор UD для каждой матрицы второго списка, запомненной в ZAPO, формирует и выполняет оператор UDAL 1, ($\&A_j$), а для матриц первого списка — оператор UDAL 1. Если используются вложенные пары ZAPO—UD, то они должны образовывать вложенную стековую структуру, например:

```
ZAPO  (A, B, C)
ZAPO  (E, D)
UD    (E), (D)
UD    ,(A, B, C)
```

Если внутри блока ZAPO—UD используются операторы ZAPOM, то все созданные ими блоки запоминания должны быть закрыты оператором UDAL перед выполнением оператора UD.

Рассмотрим пример применения операторов ZAPO—UD.

Пример 6.2.8.

Алгоритм, реализующий блочное сложение матриц (CA BSL), имеет три параметра — A, B, C. Матрицы A и B — входные, C — результат (C=A+B). В начале CA BSL закодирован оператор

```
ZAPO  (A, B, C)                    (6.17)
```

а перед концом алгоритма — оператор

```
UD  (A, B), C                      (6.18)
```

Здесь в рабочий набор выводится только метаматрица C, которая могла измениться в процессе выполнения алгоритма. Вывод ее будет выполнен только в том случае, если она не была запомнена перед вызовом CA BSL. Если бы алгоритм реализовывал формулу A=B+C, то вместо операторов (6.17), (6.18) нужно было бы закодировать

```
ZAPO  (C, B, A)
UD    (C, B), A или
ZAPO  (B, C, A)
UD    (B, C), A
```

Вообще говоря, при кодировании оператора ZAPO сначала нужно перечислять матрицы, которые не претерпевают изменений, а потом — изменяемые матрицы. Только в этом случае удается правильно закодировать оператор UD.

Для того чтобы включить режим автоматического накопления в области данных после выполнения операций СМПО используе-

мых в них блоков блочных матриц (режим очереди), применяется оператор NOCH (начать режим очереди).

Оператор NOCH является исполняемым и имеет следующий формат:

NOCH [&R]

где &R — имя поля, содержащего число в формате полуслова, равное размеру участка ОП в килобайтах, который добавляется к занятой части области данных и образует вместе с ней область стека, в которой размещаются запоминаемые матрицы, а также матрицы, не участвующие в режиме очереди. Если параметр опущен, предполагается, что &R=0. Остальная часть области данных называется областью очереди и используется для автоматического размещения в ней блоков блочных матриц.

Когда включен режим очереди, при выполнении всех операций, за исключением запоминания и удаления матриц, монитор ППП СМПО предполагает, что дескрипторы всех матриц состоят из трех слов (лишь дескрипторы запомненных матриц могут иметь длину в два слова). Чтобы дескриптор блока некоторой блочной матрицы участвовал в режиме очереди, его нужно выбирать с помощью специального формата программы GVBL (смотрим также оператор VBL):

V GVBL,(&B, &BIJ+8),&IJ

которому соответствуют поля в неисполняемой части программы

&IJ DC S(1,&I,&J, &BIJ)

&BIJ DC 3F

где &B — имя блочной матрицы (метаматрицы), которая должна быть запомнена до начала обработки ее блоков в режиме очереди и может быть удалена только после отмены режима; &BIJ — поле размером в три слова, в которое программа GVBL поместит выбранный из метаматрицы &B дескриптор нужного блока (первые два слова расширенного дескриптора) и адрес дескриптора в метаматрице (третье слово расширенного дескриптора); &IJ — поле, содержащее координаты (&I, &J) выбираемого блока. Значение координат, если они изменяются динамически, можно перед вызовом программы GVBL разнести по адресам &IJ+2 (координата &I) и &IJ+4 (координата &J).

Незапомненные матрицы, которые не участвуют в автоматическом распределении памяти, в третьем слове расширенного дескриптора при включенном режиме очереди должны содержать ноль.

Для отмены режима очереди используется оператор КОСН (конец очереди), который является исполняемым и имеет следующий формат:

КОСН

Оператор КОСН ликвидирует область очереди. При этом в рабочий набор выводятся те содержащиеся в ней блоки блочных матриц, значения которых изменились после размещения их в памяти.

6.3. ОПЕРАТОРЫ ОБРАЩЕНИЯ К СТАНДАРТНЫМ ПРОГРАММНЫМ ЕДИНИЦАМ

Для выполнения матричных операций с помощью одноуровневых СТП должны использоваться операторы: V — выполнить операцию, результатом которой является статическая матрица или поле определенных размеров, которое всегда находится в ОП; VP — выполнить операцию, результатом которой является матрица, первоначальные (до операции) значения элементов которой в операции не участвуют и полностью замещаются новыми; VPL — выполнить операцию, результатом которой является матрица, чьи первоначальные значения используются в операции или не полностью замещаются новыми; SOZ — создать матрицу указанного вида (оператор функционально совпадает с VP, отличаясь лишь формой записи).

Все эти операторы являются исполняемыми.

Первые три оператора имеют одинаковую форму записи:

& IM & OP, [&SP], & IR

где & IM — имя оператора, равное V, VP или VPL; & OP — название операции (имя СТП, которое содержится в поле & NAME оператора STP); & SP — список исходных параметров, разделенных запятыми, помещенный в круглые скобки; & IR — имя результата операции.

Формат оператора SOZ:

SOZ & MA, & IR, [& SP]

где & MA — название создаваемой матрицы (имя СТП); & IR — имя результирующей матрицы; & SP — список исходных параметров.

Пример 6.3.1

```
        V     VEL, (A, IJ), AIJ        (6.19)
        SOZ   EM, B                    (6.20)
        VP    DP, (B, A), C            (6.21)
        VPL   ZEL (AIJ, IJ), C         (6.22)
..      ...   . . . . . . .
IJ      DC    H'3, 1'                  (6.23)
AIJ     DS    D                        (6.24)
```

В примере из матрицы A выбирается элемент (6.19) с координатами IJ (6.23) и помещается в поле AIJ (6.24). Создается единичная матрица B (6.20). Матрица B поэлементно делится на матрицу A и результат помещается в матрицу C (6.21). В матрицу C по координатам IJ вставляется элемент AIJ (6.22).

По использованию указанных операторов можно сделать следующее замечание.

Внешний вид операторов не зависит от того, были или не были ранее запомнены матрицы, указанные в качестве операндов. Однако нужно быть внимательным при использовании одной и той

же матрицы в качестве исходной и результирующей. Например, можно записать оператор транспонирования матрицы А в виде

```
VP    TR, (A), A
```

Операция будет всегда выполнена правильно, если А не была ранее запомнена. В этом случае монитор пакета выделит место для результата операции и разместит исходную матрицу в разных местах области данных ОП. Если же матрица А была ранее запомнена, то место под результат и размещение исходной матрицы не выделяется, так как и результат, и исходная матрица уже находятся в ОП. В первом случае в ОП будет находиться две матрицы, во втором — только одна. Естественно, что результат транспонирования матрицы «саму в себя» в общем случае будет неверным. С другой стороны, операция сложения A=A+A (VP SL,(A,A),A) будет выполнена верно и с большей эффективностью, если А предварительно запомнить.

Вызов и выполнение многоуровневого СА производится операторами SA и SASP.

Оператор SA является исполняемым и имеет следующий формат:

```
SA    & IM [, & IS]
```

где & IM — имя СА, которое содержится в поле & NAME оператора STAL вызываемого стандартного алгоритма; & IS — имя списка фактических операндов.

По умолчанию & IS = & IM. Перечисление фактических операндов в списке должно находиться в полном соответствии со списком формальных операндов, указанных в операторе STAL. Список фактических операндов может быть задан с помощью оператора SPSA, который является неисполняемым и имеет следующий формат:

```
&IS      SPSA   & K, & S1, ..., & SI, ..., & SK
```

где & IS — имя списка фактических операндов; & K — количество выполнений СА; & SI — список фактических операндов для 1-го выполнения СА (I = 1, ..., & K).

Например, СА BUM имеет 3 параметра. Обратимся к нему 2 раза для вычисления произведений C=AB и D=BC

```
BUM      SASP  2, A, B, C, B, C, D
```

Возможно сокращение записи путем указания списков параметров трех видов:

простой список — (& PJ1, ..., & PJL, ..., & PJ & K)

индексируемый список — (& K, & PJ)

список повторений (& PJ, & N)

Напомним, что здесь & K — количество выполнений СА.

При наличии в списке операндов хотя бы одного простого или индексируемого списка считается, что фактические операнды записаны только для первого из выполнений СА, т. е. относятся к & S1. Для следующих выполнений списки операндов будут сфор-

мированы аналогично первому списку & S1. В этом случае в каждом списке & SI будут повторяться одиночные, не заключенные в скобки, параметры. Вместо простого списка будет вставлен параметр & PJI (для списка & S1 будет взят параметр & PJ1), вместо индексируемого списка — & PJ & I. Список повторений заменяется во всех & K списках & N параметрами & PJ.

Пример 6.3.2

```
            SA     PR
• • • • • • • • • • • • • • • • • • • • • • • • • • • • •
PR          SPSA   3, A1, X, C, C, D, A2, Y, C, C, D, A3, Z, C, C, D      (6.25)
PR          SPSA   3, A1, X, (C, 2), D, A2, Y, (C, 2), D, A3, Z, (C,2), D (6.26)
PR          SPSA   3, (3, A), (X, Y, Z), (C, 2),  D                        (6.27)
PR          SPSA   3, (3, A), (X, Y, Z), C, C, D                          (6.28)
```

СА с именем PR имеет список фактических операндов с тем же именем PR. Этот список может быть записан в одной из перечисленных эквивалентных друг другу форм (6.25), (6.26), (6.27), (6.28). СА должен выполняться три раза. В первом выполнении принимают участие параметры A1, X, C, C, D. Во втором — A2, Y, C, C, D. В третьем — A3, Z, C, C, D.

Для сокращения записи вместо операторов SA и SPSA можно использовать оператор SASP, который является исполняемым и имеет следующий формат:

```
&IM         SASP   &K, &S1, ..., &SI, ..., &S&K
```

где & IM — имя СА, содержащееся в поле & NAME оператора STAL вызываемого алгоритма; & K — количество выполнений СА; & SI — список фактических операндов для I-го выполнения СА (I = 1, ..., & K), который записывается по правилам оператора SPSA.

Пример 6.3.3

Оператор

```
PR        SASP  2, (2, A), (X, Y), C, E, D
```

эквивалентен оператору

```
          SA     PR, SP
```

со списком

```
SP        SPSA  2, A1, X, C, E, D, A2, Y, C, E, D
```

Программа, вызывающая СА, может не определять размеры и сами результирующие матрицы. Эти матрицы сформируют СА и передадут их дескрипторы в вызывающую программу. Имена матриц не нужно кодировать в списке параметров операторов STAL и SASP (SPSA). Для этих целей используются два оператора SPFOR и SPFAK. Оператор SPFAK кодируется в вызываемом СА. Он является исполняемым и имеет следующий формат:

```
          SPFAK (&FO1, ..., &FOI, ..., &FON), &FOR
```

где & FOI — формальный результат, значение которого передается

в вызывающую программу; & FOR — формальный параметр, содержащийся в списке формальных параметров оператора STAL и служащий только для связи оператора SPFAK с соответствующим оператором SPFOR. Этот параметр не требует дополнительного описания в стандартном алгоритме. Поле для него резервирует оператор SPFAK.

В вызывающей программе кодируется соответствующий оператору SPFAK оператор SPFOR. Он может быть использован и как неисполняемый оператор, и как исполняемый. В последнем случае он обязательно должен выполняться до оператора обращения к тому СА, который его использует. Оператор SPFOR имеет следующий формат:

&FAKT SPFOR (& FA1, . . . , &FAI, . . . , &FAN)

где & FAKT — фактический параметр, содержащийся в списке параметров оператора SASP (SPSA) и служащий для связи оператора SPFOR с соответствующим оператором SPFAK; & FAI — фактический результат, значение которого будет передано из вызываемого СА.

Оператор SPFOR описывает указанные в нем параметры & FAKT, & FA1, ..., & FAN, так что они не требуют другого определения.

Количество и порядок следования параметров оператора SPFOR должны совпадать с количеством и порядком следования параметров оператора SPFAK. В момент выполнения оператора SPFAK указанные в нем матрицы не должны быть запомнены, если в результате выполнения СА они будут удалены.

Пример 6.3.4.

Вызывающая программа:

D	SPFOR (F1, F2, F3)	(6.29)
PR	SASP 1, A, X, C, E, D	(6.30)

Стандартный алгоритм:

PR	STAL (A, X, C, E, D)	(6.31)
	OPR (2, B), M	(6.32)
	SPFAK (B1, B2, M), D	(6.33)

Матрицы F1, F2, F3, указанные в операторе (6.29), могут обрабатываться в вызывающей программе только после выполнения оператора (6.30), который вызывает СА PR (6.31). В СА результирующие матрицы B1, B2, M (6.32), (6.33) определяются и передаются в вызывающую программу.

6.4. ОПЕРАТОРЫ ОРГАНИЗАЦИИ ВВОДА-ВЫВОДА

Для вывода на печать одной строки с произвольной символьной информацией используется оператор TEKST. Оператор TEKST является исполняемым и имеет следующий формат:

TEKST &SP [, [&DL] [, &KV]]

где & SP — выводимое сообщение. Сообщение может кодироваться

либо в виде текстовой строки, указанной непосредственно в операторе TEKST и взятой в кавычки, либо в виде адреса поля выводимой информации. В качестве адреса может быть использовано символическое имя, номер общего регистра, указанный в скобках, или комбинация «смещение(база)». Например, POLE, (8), 10(2), SOOB+6; & DL — длина выводимой строки. Если & SP — непосредственная текстовая строка, параметр кодироваться не должен. Если & SP — символическое имя, параметр & DL необязателен. В этом случае в качестве длины выводимой информации будет использована неявная длина, приписываемая ассемблером данному имени. & DL может кодироваться в виде числа или в виде содержимого общего регистра, который нужно указывать в скобках. Если & SP записан в форме «(регистр)» или «смещение(база)», параметр & DL обязателен; & KV — необязательный параметр, равный числу печатаемых строк с одинаковой информацией, может кодироваться в виде числа или быть загруженным в регистр, номер которого нужно указывать в скобках. Если параметр опущен, текст печатается один раз.

Так как параметры оператора TEKST являются позиционными, наличие операнда & KV при отсутствии операнда & DL должно указываться запятой.

Рассмотрим примеры правильного кодирования оператора TEKST:

```
TEKST 'ЭТО СООБЩЕНИЕ'
TEKST '',3
TEKST SOOB
TEKST SOOB+3,4
TEKST SOOB, (2)
TEKST 0 (2), (3)
TEKST SOOB,, 3
TEKST 6(2), 4(3)
TEKST (4), 6
```

Поле SOOB может быть описано в неисполняемой части программы, например, так:

```
SOOB    DC    C' ПЕЧАТАЕМ ТЕКСТ'
```

Примеры неправильного кодирования:

```
TEKST    'ЭТО СООБЩЕНИЕ', 8
```

нельзя использовать длину совместно с непосредственной текстовой строкой;

```
TEKST    1(2)
```

если адрес выводимого сообщения задан в форме «(регистр)» или «смещение(база)», должна быть указана длина сообщения.

Для вывода на печать элементов матрицы предназначен оператор PE. Оператор PE является исполняемым и имеет следующий формат:

```
PE    &M, &SP
```

где &M — имя матрицы (выводимой); &SP — необязательный список вида (&F1,&F2), задающий формат выводимой части числовой информации для первого (&F1) и последующих (&F2) столбцов матрицы.

Если список не указан (бесформатная печать), то на печать выводятся: имя матрицы, ее размеры, номера и содержание строк (столбцов). Печать матрицы с размером $M \times N$ производится построчно, если $M \leqslant N$. В противном случае печать производится по столбцам. В одной выводимой строке может быть напечатано не более восьми чисел, которые представляются в виде десятичной дроби (со знаком «—» или без него) и показателем степени:

$0.b_1b_2b_3b_4b_5b_6b_7E \pm c_1c_2$

Перед тем, как произвести печать матрицы, осуществляется анализ всех ее элементов. Если хотя бы один элемент является дескриптором матрицы (первый байт равен X'80'), то считается, что &M — метаматрица. В этом случае по данной метаматрице печатается структура блочной матрицы. Печать производится построчно, нулевой блок обозначается как «.», ненулевой — «*».

Если список &SP присутствует в операторе, то он указывает на константы:

```
&F1    DC    H'&F11, &F12'
&F2    DC    H'&F21, &F22'
```

где &F11, &F21 — длина поля в символах, занимаемая одним числом, которое выдается на печать (десятичная точка и знак не входят в эту длину); &F12, &F22 — количество дробных десятичных разрядов, которые должны быть выведены на печать.

Согласно указанному формату производится построчная печать элементов матрицы. Количество чисел в выводимой строке зависит от формата печати. Название матрицы с ее размерами печатается только в том случае, если перед печатью установлен программный признак 2 (KONTR 2). Число, целая часть которого не умещается в указанном формате, печатается в виде * * *. Если длина поля равна нулю, то производится печать структуры матрицы. В этом случае нулевое число печатается в виде «.», ненулевое — в виде «*»

```
[* * * . . * . .]
[. . . * * . * *]
[. . . . * * . *]
```

Во время выполнения оператора возможны диагностические сообщения: НЕВЕРНЫЙ ФОРМАТ, если длина поля для всего числа меньше длины поля для дробной части числа; длина поля для чис-

ла больше 120; НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если при печати структуры матрицы (метаматрицы) количество ее столбцов больше 128.

Для подготовки к печати положительной целочисленной информации, имеющей формат полуслова (H), используется оператор VCHT. Оператор VCHT является исполняемым и имеет следующий формат:

```
VCHT  &CH, &T, &P, &DL
```

где &CH — имя поля, имеющего формат полуслова; &T — имя текстовой строки, в которую оператор VCHT запишет значение &CH, преобразованное в символьный формат; &P — номер позиции в текстовой строке, начиная с которой будет размещаться символьное значение &CH. Первый символ строки имеет номер позиции 0, второй — 1 и т. д.; &DL — длина числового поля в тексте.

В текстовой строке в том месте, куда засылаются цифры, должны быть предварительно размещены нули. Это может быть сделано командой ассемблера DC.

Для работы оператора VCHT в неисполняемой части программы должны быть зарезервированы поля PEC1 и PEC длиной по 8 байт.

Подготовленная строка может быть напечатана оператором TEKST.

Пример 6.4.2

```
          VCHT    NU, NETAP, 7, 3
          TEKST   NETAP
..................  .........
NU        DC      H'8'
NETAP     DC      C'ЭТАП N 000'
D         GDS     (PEC, PEC1)
```

В результате работы этого фрагмента будет напечатан текст ЭТАП № 008.

Для ввода одной перфокарты произвольной текстовой информации используется оператор VVOD. Оператор VVOD является исполняемым и имеет следующий формат:

```
VVOD  &OBL, &VIH
```

где &OBL — адрес поля длиной 80 байт, куда будет помещена вводимая информация. В качестве адреса может быть использовано символическое имя, номер общего регистра, указанный в скобках или комбинация смещение (база), например: (5), 4(2), KARTA+1; &VIH — метка, по которой будет передано управление, если введенная перфокарта будет содержать в первых двух позициях признак конца файла (символы/*). Если эта ситуация не нуждается в отдельной обработке, т. е. карта конца файла должна обрабатываться так же, как и другие, в качестве &VIH нужно задать метку следующей за оператором VVOD команды или оператора.

Для ввода с перфокарт десятичных чисел в качестве элементов матрицы используется оператор VV. Оператор VV является исполняемым и имеет следующий формат:

VV &M [, &C]

где &M — имя вводимой матрицы, &C — имя поля, имеющего формат полуслова, в которое программа ввода поместит число введенных чисел. Если параметр &C опущен, количество вводимых чисел должно равняться числу элементов матрицы, иначе задача будет завершена аварийно. Если же параметр &C (так называемый «ввод с подсчетом») присутствует, то количество вводимых чисел может быть меньше размеров матрицы или равно ему, и это количество будет помещено в поле, а остаток матрицы &M будет заполнен нулями. Но задача будет завершена аварийно даже при наличии параметра &C, если размеров матрицы недостаточно для размещения всех введенных чисел.

Значения элементов матрицы пробиваются на картах в позициях 1—78 построчно через пробел в одном из следующих форматов:

a a.b a*E*c *a.b*E*c k*∗*A k*(*BC*...)*k*[*BC*...]

где *a* — целая часть числа; *b* — дробная часть числа; E*c* — десятичный порядок числа ($a.b\mathrm{E}c = a{,}b \cdot 10^c$); *k* — коэффициент повторения числа *A*, записанного в любой из четырех предыдущих форм ($k{*}A = \underbrace{AA \ldots A}_{k\ \text{раз}}$); *k*(*BC*...) — повторение группы чисел *BC*..., записанных в любой из пяти предыдущих форм. Например:

3(0 1.2 —1E6) ≡ 0 1.2 —1E6 0 1.2 —1E6 0 1.2 —1E6

k[*B C* ...] — повторение каждого из чисел внутри скобок *k* раз. Числа *B* и *C* могут записываться в любой из пяти первых форм. Например:

3[0 1.2 —1E6] ≡ 0 0 0 1.2 1.2 1.2 — 1E6 —1E6 — 1E6

Во всех форматах числа *a* и *c* могут быть со знаком «—» (отрицательные) или без него (положительные).

Вложенность скобок одинаковых типов не допускается, но возможна вложенность скобок различных типов.

При вводе чисел переход к следующей по порядку карте происходит при обнаружении двух или более пробелов подряд. Не допускается перенос части числа на другую карту. Ввод чисел каждой вводимой матрицы заканчивается при обнаружении карты, содержащей в первых двух позициях признак конца файла (символ/∗).

Если задан режим слежения (см. оператор SLED), то при вводе печатается каждая вводимая карта.

При вводе информации возможны следующие диагностические сообщения: НЕДОПУСТИМЫЙ СИМВОЛ, если обнаружен символ, отличный от цифры, ∗, —, E, (,), [,], пробела; КОЛИЧЕСТВО

ВВЕДЕННЫХ ЧИСЕЛ НА N БОЛЬШЕ (или МЕНЬШЕ), ЧЕМ ЧИСЛО ЭЛЕМЕНТОВ МАТРИЦЫ, если размеры матрицы не соответствуют вводимой информации. Напомним, что при задании параметра & C это сообщение будет напечатано, только когда количество введенных чисел больше размеров матрицы.

Рассмотрим пример применения операторов VV и VVOD.

Пример 6.4.3

```
            OPR     A, (2, 30),B, (3, 6)
            VV      A, CHISLO
            VVOD    KARTA, OSHIBKA
            VV      B
   ...      ......  ..........
CHISLO      DS      H
KARTA       DS      CL80
   ...      ......  ..........
//GO.SYSIN DD *,DLM='//'
2*0 —3.45 3 (8*4 —2.77.8)
/*
ТИП=A
—7.4 2,2 8*0
4 [3.3E5 —2E.8]
/*
```

В примере определяются матрицы A и B. Затем выполняется ввод с подсчетом матрицы A. Несмотря на несоответствие размеров, ввод завершается, и в поле CHISLO помещается значение 33. Затем вводится текстовая строка ТИП= =A и помещается в поле KARTA. Предусмотрен переход на метку OSHIBKA, если вместо текстовой информации будет введена карта конца файла. Затем вводится матрица B.

Для ввода с перфокарт целых десятичных чисел в качестве элементов простого динамического списка используется оператор VVS. Оператор VVS является исполняемым и имеет следующий формат:

```
VVS    &A [, [&C] [, &T]]
```

где & A — имя матрицы, в которую будет упакован введенный список. Имя должно быть уникальным. Оно не требует дополнительного определения; & C — имя поля длиной в четыре полуслова, которое может быть описано в исполняемой части программы как

```
&C     DS    4H или &C     DS    D
```

Если поле & C присутствует (ввод с подсчетом), программа VVS поместит в него 4 числа в формате полуслова: M, N, K, MN—K. Здесь K — количество введенных чисел (не считая размеров), а M и N — размеры введенного списка, набитые на картах (MN должно быть $\geqslant$K, иначе задание будет завершено аварийно). Если пара-

метр & C опущен, числа не передаются, программа VVS проверит равенство MN=K, и, если оно не выполняется, прекратит обработку задания; & T — текстовая информация, которая указывается в кавычках. Она появится в диагностике программы VVS при обнаружении ошибки. Если параметр & T отсутствует, в диагностике будет указано имя списка — & A.

Вводимая информация набивается на перфокартах в позициях 1—80 построчно через пробел в следующем виде M : N числа, где числа набиваются в одном из следующих форматов:

$$a \quad a, b \quad c, a, b \quad k * A \quad k(BC\ldots) \quad k[BC\ldots].$$

Здесь a — целое число; a,b — арифметическая прогрессия, эквивалентная перечислению $a, a+1, \ldots, b$, если $a<b$ и $a, a-1, \ldots, b$, если $a>b$; c,a,b — арифметическая прогрессия с разностью a, содержащая c членов, эквивалентная перечислению $b, b+a, b+2a, \ldots, b+(c-1)a$; k — коэффициент повторения числа (или прогрессии) A, записанного в любой из трех предыдущих форм. В случае прогрессии k раз повторяются все ее члены; $k(BC\ldots)$ — повторение k раз группы чисел $B, C, \ldots$, записанных в любой из четырех предыдущих форм; $k[BC\ldots]$ — повторение каждого из чисел внутри скобок k раз. Числа B, C могут быть записаны в любой из четырех первых форм.

Во всех форматах числа a и b могут быть как со знаком «—» (отрицательные), так и без него (положительные).

Вложенность скобок одинаковых типов не допускается, но возможна вложенность скобок различных типов.

Так как элементы списка имеют формат полуслова, размеры матрицы, в которую он упакован, будут M и [N/4]+1, где квадратные скобки обозначают целую часть. Сама матрица будет иметь вид <M, [N/4]+1, M, N, эл 1, эл 2...>.

При вводе чисел переход к следующей по порядку карте происходит при обнаружении двух и более пробелов подряд. Не допускается перенос части числа на другую карту. Ввод чисел каждого вводимого списка заканчивается при обнаружении карты конца файла (/*).

Если информация на картах отсутствует и вместо данных сразу указывается карта конца файла, то при наличии параметра & C поля & A и & C будут обнулены, если же & C отсутствует, задание завершается аварийно.

Если среди вводимых карт будет присутствовать карта, на которой набито #, то будут напечатаны вводимые карты текущего списка, начиная с этой. Если встретится карта ##, то после работы программы будут напечатаны и карты, и введенный список.

При вводе списка программа VVS проверяет допустимость данных и может выдать диагностические сообщения. Место ошибки идентифицируется символом ¤ и печатается подробное описание ошибки, например: ВЛОЖЕННАЯ СКОБКА, ДВЕ ЗВЕЗДЫ ПОДРЯД и т. п.

Дадим примеры кодирования оператора VVS

```
        VVS   A
        VVS   SBAL, CB, 'БАЛОЧНЫХ ЭЛЕМЕНТОВ'
        VVS   MI, CM
        VVS   M,, 'МАТРИЦЫ ИНДЕКСОВ'
...     ...   . . . . . . . . . . . . . . . . . . .
D       GDS   (CB, CM)
```

Пример кодирования вводимой информации:

```
# #
2:17 — 3 2(5 2*6,9 10) 4,2 — 1 3[2*0 5]
```

Это эквивалентно:

```
# #
2:17 — 3 5 6 6 7 7 8 8 9 9 10 5 6 6 7 7 8 8 9 9
10 4 3 2 — 1 0 0 0 0 0 0 5 5 5
```

Для ввода с перфокарт целых десятичных чисел в качестве элементов сложного динамического списка предназначен оператор VVSS. Оператор VVSS является исполняемым и имеет следующий формат:

```
VVSS & A
```

где & A — имя вводимого сложного динамического списка. Имя его должно быть уникальным и не требовать дополнительного определения.

После работы оператора VVSS введенный сложный список будет размещен в оперативной памяти оператором ZAPOM и вывести его на диск нельзя: оператор UDAL просто исключает список из ОП.

После VVSS поле & A будет содержать адрес 1-го элемента адресного списка.

Вводимая с перфокарт информация должна располагаться в следующем порядке:

```
описатель
подсписок 1
подсписок 2
:
:
подсписок N
/*
```

Описатель и подсписки кодируются в том же формате, что и списки, вводимые оператором VVS. Таким образом, описатель от

подсписков и подсписки друг от друга должны отделяться картами конца файла (/*), а в конце сложного списка будут две карты конца файла.

Описатель должен содержать номера введенных подсписков; первый его размер должен быть равен 1. Возможны нарушения порядка следования номеров, повторения номеров. Возможно также задание нулевого номера, что эквивалентно появлению адреса нулевого поля в адресной части сложного списка, т. е. кодированию слова NET в адресной шкале ADSP.

Рассмотрим пример данных для оператора VVSS.

```
1:7 2 3*1 0 2 3 описатель
/*
3:2 2(1 4 5)      подсписок 1
/*
1:5 10 1,4        подсписок 2
/*
1:6 1,3 8 11 10   подсписок 3
/*
/*
```

Адресная часть этого сложного списка содержит адрес подсписка 2, три адреса подсписка 1, адрес нулевого поля, адрес подсписка 2 и адрес подсписка 3. Таким образом, логический порядок следования подсписков такой: 2, 1, 1, 1, 0, 2, 3.

Для выдачи сообщений на консоль оператора используется оператор CON. Оператор CON является исполнительным и имеет следующий формат:

```
CON   &S
```

где &S — текст выводимого сообщения в кавычках.

При выполнении оператора CON закодированный в нем текст будет выведен на консоль оператора, поэтому он не должен превышать 120 символов. Это же сообщение появится в распечатке программы перед сообщением операционной системы о завершении пункта GO задания.

Для сохранения (восстановления) матриц в библиотеке на диске используются операторы SOXD (VOSD). Операторы являются исполняемыми и имеют следующий формат:

```
&DDNAME &IM (&M1,..., &MI,..., &MN)
```

где & IM — имя оператора, равное SOXD или VOSD; & DDNAME — имя DD-оператора языка управления заданиями, описывающего библиотеку; & MI — имя сохраняемой (восстанавливаемой) матрицы. На момент сохранения в библиотеке не должно быть матриц с теми же именами. Имя восстанавливаемой матрицы должно быть уникальным в программном модуле и не требовать дополнительного определения.

Каждая матрица сохраняется в виде отдельного раздела библиотеки.

Операторам SOXD или VOSD должен соответствовать DD-оператор языка управления заданиями, описывающий библиотеку. Он должен располагаться после исходных данных. Его формат

// GO .*dd имя* DD DSN = *имя*, DISP = (*дисп*, KEEP),

// UNIT = SYSDA, VOL = SER = *том*, SPACE = (*ед*, (*к*, *пр*, *огл*))

Здесь *ddимя* — то же, что & DDNAME в операторе SOXD (VOSD); *имя* — имя набора данных (библиотеки);

$$дисп = \begin{cases} \text{NEW, если библиотека создается;} \\ \text{OLD, если она уже существует;} \end{cases}$$

том — серийный номер диска, где размещена библиотека; *имя* — ее имя

$$ед = \begin{cases} \text{TRK, если память для библиотеки выделяется в дорожках;} \\ \text{CYL, если память выделяется в цилиндрах;} \end{cases}$$

k — количество выделяемой первоначально памяти в указанных ранее единицах; *пр* — приращение памяти (если первоначальный объем будет превзойден, ОС выделит указанное количество единиц дополнительно и так до 16 раз); *огл* — количество 256-байтовых блоков, выделяемых для справочника библиотеки. Одного блока справочника достаточно для хранения информации о 12—16 матрицах. Параметр SPACE=... нужно кодировать только когда набор данных создается (*дисп*=NEW).

Пример 6.4.4

```
NAB1     VOSD  (C, K,ETA)
NAB2     SOXD  (C, ETA)
NAB3     SOXD  (K, C)
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
//GO.NAB1 DD DSN = IZ140, DISP = (OLD, KEEP),
//                UNIT = SYSDA, VOL = SER = WORKER
//GO.NAB2 DD DSN = TT3, DISP = (OLD, KEEP),
//                UNIT = SYSDA, VOL = SER = WORK02
//GO.NAB3 DD DSN = ZAPAS, DISP = (NEW, KEEP),
// UNIT = SYSDA, VOL = SER = WORKER, SPACE = (CYL, (9, 5, 3))
```

По первому оператору из библиотеки IZ140, расположенной на диске WORKER, восстанавливаются матрицы C, K и ETA. Матрицы C и ETA сохраняются в уже существующем наборе, а K и C — в новом наборе ZAPAS, который создается на диске WORKER.

Для удаления матриц, ранее записанных в библиотеку, применяется оператор UDB. Оператор UDB является исполняемым и имеет следующий формат:

& DDNAME UDB (& M1 ,. . . ,&MI ,. . . ,&MN)

где &DDNAME — имя оператора DD, описывающего библиотеку; &MI — имя удаляемой матрицы.

Формат оператора DD тот же, что и при использовании операторов SOXD и VOSD.

Необходимо помнить, что в результате работы оператора UDB имена матриц лишь вычеркиваются из каталога. Место, которое занимали сами матрицы, использоваться не будет до тех пор, пока не будет произведено сжатие библиотеки. Сжатие библиотеки можно выполнить с помощью утилиты ОС IEBCOPY [28].

Перед сжатием обязательно нужно снять копию библиотеки.

Для сохранения (восстановления) матриц на магнитной ленте используются операторы SOXL (VOSL). Они являются исполняемыми и имеют следующий формат:

```
&KAT     SOXL  (&MI ,..., &MI ,..., &MN) [, D = &DISP]
&KAT     VOSL  (&MI ,.... &MI ,..., &MN) [, &PR] [, D = &DISP]
```

где &KAT — имя каталога матриц, который описывает сохранение (восстановление) матриц. Это же имя является именем DD-оператора, описывающего набор данных на ленте; &MI — имя сохраняемой (восстанавливаемой) матрицы. Имя восстанавливаемой матрицы должно быть уникальным в рамках программной единицы (СПЕ) и не требует дополнительных описаний; &PR — признак. Если в рамках одной СПЕ повторно происходит чтение из набора данных, расположенного на ленте, нужно кодировать вместо &PR слово POVTOR. В противном случае параметр кодировать не нужно; &DISP — необязательный параметр, указывающий, что делать с лентой после сохранения (восстановления) матриц. Может принимать следующие значения: P — ленту нужно перемотать в начало; R — ленту нужно перемотать и разгрузить. Если параметр D=&DISP опущен, лента продвинется к концу набора данных. При восстановлении матриц с магнитной ленты, созданной в СМПО-ДОС, нужно обязательно кодировать D=P или D=R.

Имя каталога &KAT должно быть уникальным в рамках СПЕ. Его повторное использование допустимо только при кодировании параметра &PR.

Оператору SOXL (VOSL) должен соответствовать DD-оператор языка управления заданиями:

```
//GO. кат DD DSN = кат, DISP = (дисп, KEEP), UNIT = 5010,
//                VOL = SER = том, LABEL = (ном. тип)
```

Здесь *кат* — то же, что и &KAT в операторе SOXL (VOSL);

$$дисп = \begin{cases} \text{NEW, если набор данных создается;} \\ \text{OLD, если набор данных уже существует;} \end{cases}$$

том — серийный номер тома магнитной ленты; *ном* — число, означающее порядковый номер набора данных на томе ленты; *тип* — тип меток на томе. Если он опущен, то предполагаются стандартные метки. В этом случае можно кодировать просто LABEL=*ном*. Именно такой тип меток является наиболее предпочтительным.

Для лент без меток вместо *тип* нужно кодировать NL. Для лент, созданных в СМПО-ДОС, нужно кодировать LABEL=(2,BLP).

При обработке оператора VOSL возможна диагностика: НЕВЕРНОЕ ИМЯ КАТАЛОГА, указывающая либо на попытку восстановления матриц не с тем именем каталога, с которым они были сохранены, либо на то, что информация на ленте испорчена.

6.5. ОПЕРАТОРЫ СБОРА СТАТИСТИКИ, ОТЛАДКИ И УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ

Для печати текущего времени дня используется оператор TIMER, который является исполняемым и имеет следующий формат:

TIMER

Для печати статистической информации о прохождении задачи используется оператор STAT, который является исполняемым и имеет следующий формат:

STAT

Оператор STAT печатает текущее время, имя стандартной программной единицы, выдавшей оператор STAT и таблицу, содержащую 7 чисел: число операторов обращения; число матриц, которые были считаны в ОП из рабочего набора данных (РНД); число отдельных операций чтения информации из РНД; число матриц, записанных из ОП в РНД; число отдельных операций записи информации в РНД; число обращений к стандартным программным единицам, обработанных монитором. Седьмое число имеет смысл только при работе операционной системы в режиме *MFT* и равно числу реорганизаций, выполненных сегментом управления СПЕ при заполнении всей отведенной для размещения СПЕ памяти.

Далее печатается максимальное число дорожек, на которых во время выполнения размещалась информация рабочего набора.

Если в процессе выполнения программы пользователя не использовался режим очереди, печатается размер использованной части области данных в килобайтах. Если использовался режим очереди, такого сообщения выдать нельзя, так как в области очереди используется вся выделенная память, и неизвестен минимальный размер области данных, достаточный для решения задачи.

Если был закодирован оператор ТКО, то далее печатается таблица использованных подпрограмм, вид которой будет рассмотрен ниже.

В СМПО имеется возможность определить, сколько раз выполнялась каждая программа. Для этого используется оператор ТКО (таблица количества обращений), который является исполняемым и имеет следующий формат:

ТКО

Оператор ТКО должен кодироваться в основной программе. После его выполнения включается подсчет программ. Каждая новая про-

грамма включается в формируемую таблицу программ. При повторном вызове программы увеличивается на один счетчик число ее выполнений. Таблица распечатывается оператором STAT вместе с другой статистической информацией. Каждая строка таблицы содержит имя программы и число ее выполнений.

СМПО позволяет запросить у операционной системы интервал процессорного времени. Такая возможность является особенно удобной при решении больших итерационных задач. Когда интервал будет исчерпан, управление можно будет передать на метку пользователя. Эти функции выполняет пара операторов: UTIMER и PTIMER.

Для запроса временнóго интервала используется оператор UTIMER (установить таймер). Оператор UTIMER является исполняемым и может быть закодирован в любом из 2-х форматов:

1-й формат:

UTIMER H = &H, M = &M, C = &C

где &H; &M; &C — количество выделяемых часов, минут и секунд соответственно.

Если любой из параметров &M, &H, &C опущен, его значение предполагается равным нулю. Параметры могут кодироваться в любом порядке;

2-й формат:

UTIMER A = &ADR

где &ADR — адрес в форме символического имени или в форме «смещение(база)», указывающий на поле, содержащее три числа в формате полуслова (H): &H, &M, &C. В этом поле в отличие от первого формата не допускается пропуск или перестановка параметров &H, &M, &C.

Для проверки завершения интервала времени, запрошенной оператором UTIMER, используется оператор PTIMER (проверить таймер). Оператор PTIMER является исполняемым и имеет следующий формат:

PTIMER &VIH

где &VIH — метка, на которую будет передано управление, если интервал времени, выделенный оператором UTIMER, исчерпан. В противном случае управление получит оператор, следующий за оператором PTIMER. Если до выполнения оператора PTIMER оператор UTIMER не был выполнен, оператор PTIMER сразу передаст управление на метку &VIH.

Обычно бывает удобно выполнять оператор PTIMER в цикле, выход из которого будет производиться по истечению заданного времени.

Операторы UTIMER и PTIMER могут кодироваться в разных программных единицах. Например, оператор UTIMER можно указать в основной программе перед вызовом алгоритма, содержащего оператор PTIMER.

При обнаружении ошибок в вычислительном процессе может потребоваться выдать диагностическое сообщение и прекратить выполнение задачи. Для этого используется оператор DIAG. Оператор DIAG является исполняемым оператором и имеет следующий формат:

```
DIAG &K
```

где &K — код причины (двузначное число), в зависимости от значения которого печатаются следующие сообщения:

01. НЕДОПУСТИМЫЙ СИМВОЛ!
02. НЕВЕРНЫЙ СПИСОК ПАРАМЕТРОВ!
03. НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ!
04. НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ!
05. СТЕК БЛОКОВ МАТРИЦ ЗАПОЛНЕН!
06. НЕКВАДРАТНАЯ МАТРИЦА!
07. ВЫРОЖДЕННАЯ МАТРИЦА!
08. НЕВЕРНЫЙ ФОРМАТ!
12. НЕВЕРНЫЕ РАЗМЕРЫ!
13. НЕВЕРНЫЕ КООРДИНАТЫ!
14. НЕРЕГУЛЯРНАЯ СТРУКТУРА!
16. НУЛЕВОЙ БЛОК!
17. ОБЛАСТЬ ДАННЫХ ЗАПОЛНЕНА!
18. МАТРИЦА, ОБЪЯВЛЕННАЯ ОПЕРАТОРОМ MATR, НЕ ЗАПОМНЕНА!
19. ОШИБКА ВВОДА-ВЫВОДА!
20. НЕВЕРНОЕ ИМЯ КАТАЛОГА!
21. НЕДОСТАТОЧНЫЙ РАЗМЕР ПАМЯТИ ДЛЯ ОЧЕРЕДИ!
22. МАТРИЦА НЕВЕРНО ПЕРЕДАНА ИЗ ПОДЗАДАЧИ!
23. АВОСТ В ПОДЗАДАЧЕ. СЧЕТ ПРЕКРАЩЕН
24. УДАЛЯЕМАЯ МАТРИЦА НЕ НАЙДЕНА

После вывода сообщения печатается дамп, структура которого описывается в разд. 9.4.

Если пользователь желает выдать собственное диагностическое сообщение, отличное от перечисленных, он может напечатать его оператором TEKST, а затем применить оператор DIAG с кодом &K=99. Если же &K примет значение, отличное от вышеуказанных и не равное 99, будет напечатано сообщение НЕОПОЗНАННОЕ ПРОГРАММНОЕ ПРЕРЫВАНИЕ!

Для прекращения выполнения задачи (обычно при обнаружении неисправимой ошибки) используется оператор EOJ. Оператор EOJ является исполняемым и имеет следующий формат:

```
EOJ
```

Оператор EOJ эквивалентен макрокоманде OC ABEND с кодом пользователя 4. Он прекращает выполнение задачи пользователя. Дамп не печатается, если в пакет задания не добавлен DD-оператор языка управления заданиями

```
//GO. SYSUDUMP DD SYSOUT=A
```

Для отладки программ в СМПО имеется режим слежения, для включения и выключения которого используется оператор SLED. Оператор SLED является исполняемым и имеет следующий формат:

SLED &K

где &K — код причины (цифра от 1 до 7), указывающий вид слежения.

Слежение бывает трех видов:

а) прослеживание входов в подпрограммы. При входе в подпрограмму печатается ее название и текущее время;

б) прослеживание незапомненных результатов матричных операций. После выполнения стандартных программ, вызванных операторами VP, SOZ, VPL, результаты их выполнения печатаются, если они не были запомнены оператором ZAPOM;

в) прослеживание запомненных результатов матричных операций — то же, что и в режиме *б*, но печатаются только запомненные матрицы.

Режиму *а* соответствует &K=1, *б* — &K=2; *в* — &K=4. Возможно объединение режимов. Например, оператор SLED 5 включает режимы *а* и *б*, а оператор SLED 7 включает полный режим слежения.

Если параметр &K опущен, предполагается &K=2.

Повторное кодирование оператора SLED с тем же кодом причины вызывает выключение режима слежения.

Слежение с кодом &K=2 вызывает одновременно распечатку карт, вводимых оператором VV.

Выполняемая в режиме слежения печать результатов матричных операций является бесформатной (см. оператор PE), все прослеживаемые матрицы печатаются под именем REZULT.

Для отладочной печати содержимого регистров с плавающей точкой, отдельных полей информации с плавающей точкой двойной точности, матриц, размещенных в памяти, используется оператор OTPE. Оператор OTPE является исполняемым и имеет следующий формат:

OTPE [&SP1], [&SP2], [&SP3]

где &SP1 — список номеров регистров с плавающей точкой, которые нужно печатать; &SP2 — список имен полей в памяти длиной в двойное слово, которые нужно напечатать; &SP3 — список полей, содержащих адреса матриц, которые нужно напечатать. Для запомненных матриц элементы списка могут иметь вид &M+4, где &M — имя матрицы.

Печать бесформатная. Ее производит программа PE, подключаемая к отлаживаемому модулю статически, что увеличивает потребности в ОП. Поэтому оператор OTPE рекомендуется использовать только для отладки.

Регистры и поля печатаются в следующем виде: все они логически объединяются в одну матрицу-строку в том порядке, в котором они перечислены. Перед значениями печатается слово ЧИСЛА

и приводится размер матрицы-строки. Печать бесформатная (см. оператор PE).

Матрицы печатаются обычным образом (бесформатная печать).

Все параметры оператора OTPE необязательные. Их отсутствие должно обозначаться запятой, если вслед за пропущенными параметрами следуют присутствующие. Если любой из списков & SP1, & SP2, & SP3 состоит из одного элемента, ограничивающие его скобки могут быть опущены.

Оператор OTPE может применяться в основной программе, в стандартном алгоритме и в стандартной программе. В последнем случае в операторе STP этой программы должен быть указан номер базового регистра, отличный от 0, 1, 13, 14, 15.

Примеры кодирования оператора OTPE

```
OTPE  (4,2)

OTPE  (0,6), (D1, AB), (MAT + 4,DA + 4)

OTPE  ,, DI

OTPE  0,, MAT + 4
```

6.6. ДОПОЛНИТЕЛЬНЫЕ ОПЕРАТОРЫ

Рассмотрим сначала группу операторов СМПО, которые применяются вместо нескольких однотипных машинных команд или команд ассемблера и служат для сокращения записи.

Для рассылки второй половины общего регистра в несколько полей можно использовать оператор GSTH, который является исполняемым и имеет следующий формат:

```
GSTH  &R, (&P1, ..., &PI, ..., &PN)
```

где &R — номер общего регистра; &PI — адрес в памяти, по которому происходит рассылка. В качестве адреса может использоваться символическое имя или комбинация «смещение(база)».

Оператор полностью эквивалентен последовательности команд

```
STH   &R, &P1
. . . . . . . . . .
STH   &R, &PN
```

Примеры кодирования оператора GSTH:

```
        GSTH  2, (IJ, IJ2 + 2, IJ1 + 6, KIJ)
M2      GSTH  1, (II + 2, JJ, KK + 4)
```

Для рассылки значения регистра с плавающей точкой в несколько полей используется оператор GSTD, который является исполняемым и имеет следующий формат:

```
GSTD  &RP, &RB, (&S1, ..., &SI, ..., &SN)
```

где & RP — номер регистра с плавающей точкой; & RB — номер

общего регистра, который совместно с описанными далее смещениями участвует в формировании адресов, по которым производится рассылка значения из регистра &RP; &SI — значение смещения, которое складывается с содержимым регистра &RB для получения адреса, по которому происходит запись в память регистра &RP. Полученное значение адреса в соответствии с требованиями ЕС ЭВМ должно быть кратно восьми.

Оператор полностью эквивалентен следующей последовательности команд:

```
STD    &RP,&S1 (&RB)
. . . . . . . . . . . . .
STD    &RP,&SN (&RB)
```

Рассмотрим пример кодирования оператора GSTD.

Пример 6.6.1

Пусть регистр 1 содержит адрес матрицы, размещенной в области данных, т. е. адрес ее размеров. Известно, что размеры матрицы 3×3. Требуется присвоить элементам (1.1), (1.2) и (2.1) значение из регистра с плавающей точкой 0. Это сделает оператор

```
GSTD  0, 1, (4, 12, 28)
```

Здесь смещение 4 соответствует элементу (1,1), 12—(1,2) и 28—(2,1).

Для присвоения нескольким символическим именам одного перемещаемого значения используется оператор GEQU. Он может кодироваться как в форме исполняемого, так и неисполняемого оператора. Формат оператора GEQU:

```
&APZ      GEQU  (&IM1,...,&IMI,...,&IMN)
```

где &APZ — ранее определенное имя, чье значение присваивается другим именам; &IMI — имя, которому присваивается значение &APZ.

Оператор эквивалентен следующей последовательности команд ассемблера:

```
&IM1      EQU   &APZ
. . . . . . . . . . . . .
&IMN      EQU   &APZ
```

Оператор GEQU не создает новых полей информации, а просто присваивает уже имеющемуся полю &APZ новые имена &IM1, ..., &IMI, ..., &IMN.

Примеры кодирования оператора GEQU:

```
NET       GEQU  (SL, SQ, SRV, SDC, SRS, SVC)
E         GEQU  (F1, F2, F3)
```

Для резервирования нескольких статических полей оперативной памяти одинакового размера используется оператор GDS, который является неисполняемым и имеет следующий формат:

```
&T        GDS   (&P1,...,&PI,...,&PN)
```

где &T — буквенно-цифровой параметр, начинающийся с буквы и определяющий тип и длину резервируемых полей информации. Для выделения двойных слов в качестве &T можно использовать символ D, простых слов — F, E или A, полуслов — H, однобайтовых полей — X. Для выделения полей другой длины в качестве &T можно использовать комбинацию CL&L, где &L — длина резервируемых полей в байтах. В этом случае оператор GDS не производит выравнивания полей на какие бы то ни было границы; &PI — имя резервируемого поля.

Оператор GDS эквивалентен следующей последовательности команд ассемблера:

```
&P1      DS    &T
. . . . . . . . . . .
&PN      DS    &T
```

Поля, созданные оператором GDS, не обнуляются и содержат перед выполнением программы произвольную информацию.

Примеры кодирования оператора GDS:

```
D       GDS   (A, B, C, K, K1V, D, D1, A2)
A       GDS   (ADRA, ADRB, M, N, FF)
H       GDS   (K, L, I, J, I1, KL2)
CL10    GDS   (PP1, PP3, PP8)
CL20    GDS   (STROKA, STRZ, STR1)
```

Для изменения значений целочисленных переменных, заданных в формате полуслова (H), наряду с машинными командами можно использовать оператор UPAR. Он является исполняемым и имеет следующий формат:

```
UPAR  (&P1,..., &PI,..., &PN), &C
```

где &PI — имя изменяемой переменной; &C — целое число, которое алгебраически складывается со значениями переменных. Величина &C может быть как положительной, так и отрицательной.

Если список содержит имя только одной переменной, скобки могут быть опущены.

Оператор UPAR изменяет значения общих регистров 0 и 1.

Рассмотрим примеры применения оператора UPAR:

```
UPAR  I, 1
```

— переменная I увеличивается на единицу;

```
UPAR  (J, K, L), —2
```

— значения переменных J, K и L уменьшаются на 2.

Для записи дескрипторов матрицы в качестве блока в метаматрицу блочной матрицы используется оператор ZBL, который является исполняемым и имеет следующий формат:

```
ZBL   &M, (&IJ1,..., &IJK,..., &IJN), &BLM
```

где &M — имя матрицы размером $M_0 \times N_0$, которая в качестве блока записывается в метаматрицу; &IJK — имя поля, содержащего две координаты (i_k, j_k) в формате полуслова, по которым производится запись дескриптора; &BLM — имя метаматрицы размером $M \times N$.

Оператор ZBL обычно применяется при формировании метаматриц. Он вызывает программу GZBL, при передаче управления которой монитор обеспечивает нахождение в ОП матрицы &BLM. Программа GZBL выполняет запись дескрипторов в &BLM и возвращает управление вызывающей программе.

При выполнении оператора ZBL обычно нужно обеспечить, чтобы записываемый блок располагался в рабочем наборе данных, а не был бы запомнен или накоплен в области очереди. В противном случае после удаления блока из ОП метаматрица будет содержать неверную информацию. В режиме очереди оператор ZBL должен стоять сразу после оператора OPR для матрицы &M или, во всяком случае, до того, как матрица &M начнет участвовать в очереди.

В процессе выполнения оператора ZBL (программы GZBL) возможна диагностика: НЕВЕРНЫЕ КООРДИНАТЫ!, если нарушено одно из неравенств: $1 \leqslant i_k \leqslant M$, $1 \leqslant j_k \leqslant N$.

Для выборки дескриптора блока из ранее запомненной метаматрицы используется оператор VBL, который является исполняемым и имеет следующий формат:

```
VBL    &BLM,(&IJ), &BLOK
```

где &BLM — запомненная метаматрица блочной матрицы; &IJ — имя поля, содержащего две координаты (i_k, j_k) в формате полуслова, по которым производится выборка дескриптора; &BLOK — ранее определенное 8-байтовое поле, в которое оператор поместит выбранный дескриптор блока.

В отличие от оператора ZBL оператор VBL выполняет действия по выборке дескриптора самостоятельно, без участия монитора и вспомогательных программ. Поэтому оператор VBL может обрабатывать только запомненные метаматрицы и не проверяет правильность координат — это должен обеспечить программист. Если метаматрица не запомнена или координаты не проверены, выборку дескрипторов можно осуществить, обратившись к программам GVBL или VEL.

Если оператор VBL используется для выборки дескриптора блока, который участвует в режиме очереди, нужно использовать другой формат обращения к оператору VBL:

```
VBL    &BLM,(&IJ, &BLOK+8), &BLOK
```

Здесь &BLOK — ранее определенное поле размерами 3F (12 байт, выравнивание на границу слова), в которое оператор VBL поместит 12-байтовый дескриптор блока, участвующего в режиме очереди.

Для включения и выключения программно-управляемых признаков (переключателей) используется оператор KONTR, который является исполняемым и имеет следующий формат:

```
KONTR &P
```

где & P — шестнадцатеричная цифра, означающая номер включаемого (выключаемого) признака.

Всего имеется четыре независимых признака с номерами 1, 2, 4, 8. Остальные признаки — составные, например 6=4+2, F=8+ +4+2+1. Оператор включает все выключенные признаки, входящие в состав & P, и выключает все включенные признаки из & P. Оператор KONTR может кодироваться в любой СПЕ. Включенные признаки образуют маску программы, доступную для любой программной единицы СМПО. Проверку маски можно осуществить командами

```
TM    0(13), X'0&P'
BNO   &M                                    (6.34)
```

где & M — метка некоторой команды (оператора). Эти команды передадут управление на метку & M, если хотя бы один из простых признаков в & P выключен. Если вместо выражения (6.34) указать

```
BZ    &M
```

переход к & M произойдет, если все простые признаки выключены. Если же вместо (6.34) указать

```
BO    &M
```

переход к & M произойдет при всех включенных простых признаках из & P.

Программные переключатели удобно использовать вместо параметров, задающих режимы работы подпрограмм. В СМПО переключатели широко используются при управлении режимами информационной печати из стандартных алгоритмов. Так, например, при указании признака

```
KONTR 2
```

программа BRXL выдает дополнительную информацию о своем выполнении: через каждые 10 обработанных суперстрок печатается текущая структура разлагаемой матрицы. При включении же признака

```
KONTR 4
```

наоборот, отменяется печать названия алгоритма и времени начала и конца его выполнения, которые выводятся, если все переключатели выключены.

Оператор

```
KONTR &P
```

примененный повторно, возвращает все переключатели к тому виду, который они имели до первого его выполнения. Если в процессе выполнения программы оператор KONTR не выполнялся ни разу, все переключатели выключены.

Для создания простых и сложных статических списков используются операторы ADSP и CSP, которые являются неисполняемы-

ми. Заголовок сложного статического списка формирует оператор ADSP (адресный список), имеющий следующий формат:

```
& IS      ADSP   & S1 ,..., & SI ,..., & SN
```

где & IS — имя заголовка сложного статического списка, которое также будет и именем всего сложного списка. Оно должно быть уникальным в рамках СПЕ; & SI — элемент заголовка, который может быть закодирован в следующем виде: A_i — имя подсписка сложного списка; (k_i, A_i) — эквивалентно перечислению имен A_i1, $A_i2, ..., A_ik_i$; (A_i, k_i) — эквивалентно указанию имени A_i k_i раз.

Заголовок сложного статического списка компилируется Макроассемблером к виду

```
& IS      DC     A (* + 4, A1, A2 ,..., Aj ,..., Am, 0)
```

где A_j — имя, полученное при расшифровке элементов & SI.

При формировании простых статических списков и подсписков сложных статических списков используется оператор CSP (числовой список), имеющий следующий формат:

```
& IS      CSP    & S1,..., & SI,..., & SN
```

где & IS — имя простого списка или подсписка сложного списка; & SI — элемент списка, который может быть закодирован в следующем виде: b_i — число; (b_i, c_i) — арифметическая прогрессия вида b_i, b_i+1, $b_i+2, ..., c_i$ если $b_i<c$, или вида b_i, b_i-1, $b_i-2, ..., c_i$, если $b_i>c_i$; (d_i, b_i, c_i) — арифметическая прогрессия, состоящая из d_i членов, с шагом b_i и первым членом c_i : c_i, c_i+b_i, $c_i+2b_i, ..., c_i+$ $+(d_i-1)b_i$.

Числовой список компилируется к виду

```
& IM      DC     H'M, l1, l2,..., l'M,
```

где l_i — число, полученное при расшифровке элементов & SI; M — количество полученных при расшифровке чисел l_i.

При трансляции оператора CSP возможно диагностическое сообщение НЕПОЛОЖИТЕЛЬНО ЧИСЛО ПОВТОРЕНИЙ: (d_i, b_i, c_i), если $d_i \leqslant 0$.

Рассмотрим пример применения операторов ADSP и CSP.

Пример 6.6.4

```
S     ADSP  (2, S), S2,(S3, 3)        (6.35)
S1    CSP   1,4,(6,8)                 (6.36)
S2    CSP   (-1,3), 2, (3, 4, 2)      (6.37)
S3    CSP   (4, -2,1), 3, 6, 8        (6.38)
```

Оператор (6.35) порождает список S, содержащий адреса S1, S2, S2, S3, S3, S3. Оператор (6.36) порождает числовой список S1, содержащий следующую информацию: 5,1,4,6,7,8; оператор (6.37) — список S2: 9,—1,0,1,2,3,2,2,6,10; оператор (6.38) — список S3: 7,1,—1,—3,—5,3,6,8.

Глава 7. Библиотека стандартных программных единиц

Библиотека загрузочных модулей ППП СМПО содержит тексты программ СМПО, отредактированных и полностью готовых к выполнению. В главе даются описания программ библиотеки. Возможные диагностические сообщения будут приведены для конкретных подпрограмм и обозначены символом Д.

7.1. СОЗДАНИЕ СПЕЦИАЛЬНЫХ МАТРИЦ. СПЕЦИАЛЬНЫЕ ПРЕОБРАЗОВАНИЯ МАТРИЦ.

Для создания матриц определенного вида [2] используется оператор

```
SOZ    &T, &M
```

где &T — тип создаваемой матрицы; &M — имя создаваемой матрицы размером M×N.

Параметр &T может принимать следующие значения: NM — 0-матрица (нулевая), EM — E-матрица (единичная), SM — Σ-матрица (суммирующая), AM — α-матрица, BM — β-матрица, LINM — L-матрица (линейная):

$$0=\begin{bmatrix}0&0&0&\dots&0\\0&0&0&\dots&0\\&&\cdot&\cdot&\cdot\\0&0&0&\dots&0\end{bmatrix};\ E=\begin{bmatrix}1&0&0&\dots&0\\0&1&0&\dots&0\\&&\cdot&\cdot&\cdot\\0&0&0&\dots&1\end{bmatrix};$$

$$\sum=\begin{bmatrix}1&0&0&\dots&0\\1&1&0&\dots&0\\&&\cdot&\cdot&\cdot\\1&1&1&\dots&1\end{bmatrix};\ \alpha=\begin{bmatrix}1&0&0&\dots&0&-1\\-1&1&0&\dots&0&0\\&&\cdot&\cdot&\cdot&\cdot\\0&0&0&\dots&-1&1\end{bmatrix};$$

$$\beta=\begin{bmatrix}1&0&0&\dots&0&1\\1&1&0&\dots&0&0\\&&\cdot&\cdot&\cdot&\\0&0&0&\dots&1&1\end{bmatrix};\ L=\begin{bmatrix}1&&2&3&\dots&N\\N+1&&N+2&N+3&\dots&2N\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\(M-1)N+1&\dots&\dots&\dots&\dots&MN\end{bmatrix}.$$

Для создания константы-матрицы необходимо использовать программу KM:

```
          SOZ    KM, &M, (&K)
&K        DC     D'&C'
```

где &M — имя матрицы, все элементы которой равны &C; &K — имя константы C.

Ленточная матрица создается из векторов

```
SOZ    LENM, &M, (&V1,..., &VI,..., &VL)
```

где &M — имя симметричной матрицы, в которой элементы (i, j)

равны нулю для $|i-j| \geqslant L-1$; &VI — имя вектора размером $K_I \times 1$, компоненты которого рассылаются в элементы (i, j) матрицы &M ($|i-j|=I-1$; $I=1, \ldots, L$).

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $M \neq N$ или $K_I \neq M-(I-1)$.

Для создания матрицы псевдослучайных чисел, равномерно распределенных на промежутке [0,1), используется программа MSC:

```
SOZ   MSC, &M, (&I)
```

где &M — имя создаваемой матрицы; &I — имя поля размером в простое слово, содержащее до обращения к программе MSC начальное значение датчика псевдослучайных чисел, после обращения — новое значение датчика. При последующих вызовах программы MSC рекомендуется в качестве &I использовать значение, полученное при предыдущем к ней обращении. Начальное значение &I может быть произвольным. Если требуется полная воспроизводимость процесса, в качестве начального значения используется любая константа (например, &I DC F'0'). Если процесс должен каждый раз при новом запуске задачи выдавать новую последовательность случайных чисел, в качестве &I можно использовать содержимое таймера ЭВМ.

В результате работы программы MSC создается матрица &M, элементы которой — числа с плавающей точкой двойной точности. Псевдослучайными являются только первые половины этих чисел, имеющие формат чисел с плавающей точкой простой точности; вторые половины — нулевые.

Для генерации псевдослучайных чисел используется линейный конгруэнтный метод [21], реализующий формулу

$$X_{n+1}=(aX_n+c) \bmod m,$$

причем в качестве m использована величина 2^{31}, $a=5^{13}$, $c=45381655$. Такой выбор гарантирует высокое качество датчика; длина его цикла равна 2^{31}. В качестве элементов &M используются первые 24 из полученных 31 бита псевдослучайного числа, дополненные слева характеристикой X'40'. Если нужны все 31 бит, то пользователь может использовать величину &I в качестве источника псевдослучайных чисел, при этом размеры &M можно выбрать как 1×1.

Фоном блочной матрицы назовем метаматрицу, содержащую информацию о размерах всех блоков; дисковые адреса блоков равны нулю. Создать фон можно командами:

```
        SOZ     FONM, &M, (&S)
&S      ADSP    &S1, &S2
&S1     CSP     k1, m1, ..., ki, mi, ..., kl, ml
&S2     CSP     r1, n1, ..., rj, nj, ..., rq, nq
```

где &M — имя матрицы размером $M \times N$; &S — имя шкалы, содержащей имена (адреса) подшкал, описывающих размеры блоков по строкам (&S1) и столбцам (&S2); k_i — число суперстрок, содер-

жащих блоки с первым размером $m_i \left(\sum_{i=1}^{l} k_i = M\right)$; r_j — число суперстолбцов, содержащих блоки со вторым размером $n_j \left(\sum_{j=1}^{q} r_j = N\right)$.

Заполнение матрицы производится с первой суперстроки до последней группами по k_i. Затем заполняются суперстолбцы. В первый байт размера по строкам заносится признак X'80'.

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $\sum_{i=1}^{l} k_i > M$ или $\sum_{j=1}^{q} r_j > N$.

Для создания блочной диагональной матрицы из вектор-строки, содержащей элементы диагонали, используется СА BDIM:

```
BDIM     SASP  1,&A,&LEN
```

где &A — блочная квадратная матрица, фон которой создан. Блочный размер &A M×M, абсолютный — N×N; &LEN — простая вектор-строка размером 1×N, содержащая диагональные элементы матрицы &A.

Для создания реальной блочной нулевой матрицы используется СА BNM:

```
BNM      SASP  1, &A
```

где &A — блочная матрица, либо имеющая реальные блоки, либо матрица, для которой создан фон. В результате работы программы все существующие блоки будут обнулены, остальные блоки будут созданы и обнулены.

Операторное преобразование матриц соответствует умножению матрицы специального вида на данную матрицу.

Общий вид обращения к подпрограммам:

```
VP     &I,(&A), &C
```

где &I — имя подпрограммы; &A — имя матрицы размером M×N, к которой применяется оператор; &C — имя результирующей матрицы размером K×L.

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $M \neq K$ или $N \neq L$.

Параметр &I может принимать следующие значения: AO — альфа-оператор, соответствует выполнению операции $C = \alpha A$; ATO — альфа-транспонированный оператор ($C = \alpha^{T} A$); AMO — альфа-модифицированный оператор ($C = \alpha^{*} A$, где α^{*} — модифицированная α-матрица, т. е. α-матрица, у которой последний столбец нулевой); AMTO — альфа-модифицированный транспонированный оператор ($C = (\alpha^{*})^{T} A$); BO — бетта-оператор ($C = \beta A$); BTO — бетта-транспонированный оператор ($C = \beta^{T} A$); BMO—бетта-модифицированный

оператор ($C=\beta^*A$, где β^* — модифицированная β-матрица, т. е. β — матрица, у которой последний столбец нулевой); BMTO — беттамодифицированный транспонированный оператор ($C=(\beta^*)^T A$); SO — суммирующий оператор $\left(C=\sum A\right)$; SMO — суммирующий модифицированный оператор $\left(C=\sum^* A,\ \sum^*=\sum-E\right)$.

7.2. ПОЭЛЕМЕНТНЫЕ ОПЕРАЦИИ НАД МАТРИЦАМИ

К группе поэлементных операций относятся операции, результат которых получается посредством простых алгебраических операций над отдельными элементами исходных матриц.

Для бинарных операций общий вид обращения:

VP & I, (&A, &B), &C

где & I — название поэлементной операции (подпрограммы); & A — имя первой исходной матрицы размером $M\times N$; & B — имя второй исходной матрицы размером $S\times T$, либо имя константы $в$, заданной в виде &B DC D'$в$'; &C — имя результирующей матрицы размером $K\times L$.

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если не выполняются условия $M=S=K$ или $N=T=L$.

Обозначим через a_{ij}, b_{ij}, c_{ij} — элементы матриц A, B, C. Параметр & I может принимать следующие значения; SL — сложение матриц ($c_{ij}=a_{ij}+b_{ij}$); V — вычитание матриц ($c_{ij}=a_{ij}-b_{ij}$); UMP — поэлементное умножение матриц ($c_{ij}=a_{ij}b_{ij}$); DP — поэлементное деление матриц ($c_{ij}=a_{ij}/b_{ij}$); SLK — сложение матрицы с константой ($c_{ij}=a_{ij}+b$); VK — вычитание из матрицы константы ($c_{ij}=a_{ij}-b$); UMK — умножение матрицы на константу ($c_{ij}=a_{ij}b$); DK — деление матрицы на константу ($c_{ij}=a_{ij}/b$); KV — вычитание из константы матрицы ($c_{ij}=b-a_{ij}$); KD — деление константы на матрицу ($c_{ij}=b/a_{ij}$); NUL — обнуление относительно малых элементов матрицы (если $m=\max|a_{ij}|\neq 0$, то для каждого элемента a_{ij} вычисляется $d_{ij}=|a_{ij}|/m$, для $d_{ij}<b$ — элемент $c_{ij}=0$, иначе $c_{ij}=a_{ij}$); NULA — обнуление элементов $c_{ij}\leqslant b$

VPL NULA,(B), C

Для унарных операций общий вид обращения:

VP &I, (&A), &C

Параметр & I может принимать значения:

VKP — поэлементное возведение матрицы в квадрат ($c_{ij}=a_{ij}^2$); IKP — поэлементное извлечение квадратного корня ($c_{ij}=\sqrt{a_{ij}}$).

Д: НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ, если $a_{ij}<0$; ABS — вычисление абсолютной величины ($c_{ij}=|a_{ij}|$); EXP — поэлементное вычисление экспоненты ($c_{ij}=e^{a_{ij}}$); LGP — поэлементное вычисление натурального логарифма ($c_{ij}=\ln a_{ij}$).

Д: НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ, если $a_{ij}\leqslant 0$. VSP — поэлементное вычисление синусов ($c_{ij}=\sin a_{ij}$);

VCP — поэлементное вычисление косинусов ($c_{ij}=\cos a_{ij}$); ATP — поэлементное вычисление арктангенса ($c_{ij}=\mathrm{arctg}\, a_{ij}$); THP — поэлементное вычисление гиперболического тангенса ($c_{ij}=\mathrm{th} a_{ij}$); SLD — сложение квадратной матрицы C с диагональной матрицей A, записанной в виде вектор-строки или вектор-столбца; результат — на месте C ($c_{ii}=c_{ii}+a_i$: VPL SLD, (& A), & C); PER — перепись элементов одной матрицы в другую ($c_{ij}=a_{ij}$).

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $MN\neq KL$; IZ — изменение знака матрицы ($c_{ij}=-a_{ij}$); TR — транспонирование матрицы ($c_{ij}=a_{ij}$).

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $M\times N\neq K\times L$; размеры в поле матрицы C устанавливаются равными $N\times M$, независимо от размеров, хранящихся в ее дескрипторе; NORJ — нормирование элементов строк матрицы (для $i=1, ..., M$; $j=1, ..., N$ $c_{ij}=a_{ij}/d_i$, где $d_i=\max\limits_j|a_{ij}|\neq 0$, если $\max\limits_j|a_{ij}|=0$, то $d_i=1$); OTR — отражение элементов матрицы относительно главной диагонали ($a_{ji}=a_{ij}$, если $a_{ij}\neq 0$, в противном случае $a_{ij}=a_{ji}$; результат формируется в исходной матрице: VPL OTR,,&A).

Бинарные поэлементные операции над блочными матрицами A, B, C осуществляются с помощью операторов

```
& BI      SASP  1, &A, &B, & C
```

Д: НЕВЕРНЫЕ РАЗМЕРЫ, если размеры метаматриц не соответствуют смыслу выполняемой операции.

Название операции & BI может принимать следующие значения:

BSL — блочное сложение (A+B=C); BV — блочное вычитание (A—B=C); BUMP — блочное поэлементное умножение ($C=A\odot B$, где $\odot$ — знак поэлементного умножения); BDP — блочное поэлементное деление ($C=A\odot B$, где $\odot$ — знак поэлементного деления); BOPK — блочные операции с константой (A ⊛ C=A), здесь второй операнд задан в виде &B DC CL4′&I′, где &I — название поэлементной операции матрицы с константой «SLK», «VK», «UMK» и т. д., третий операнд задается в виде & C DC D′C′, где C — константа, участвующая в операциях с блоками матрицы A); BSLD — сложение блочной квадратной матрицы A с блочной диагональной матрицей B, хранимой в виде блочного столбца. Результат получается на месте C.

В унарных операциях может отсутствовать не только операнд &B, но и операнд &C (&BI SASP 1, &A, &C или &BI SASP 1, &A): BTR — блочное транспонирование ($A^{\mathrm{T}}=C$); BPER — блочная перепись (A=C); BOTR — блочное отражение внедиагональных блоков (для $i\neq j$ выполняется $A_{ji}=A_{ij}^{\mathrm{T}}$, если $A_{ij}\neq 0$, в против-

ном случае для $A_{ji} \neq 0$ $A_{ij} = A_{ji}^{\tau}$; BODB — блочное отражение диагональных блоков (над диагональными блоками выполняется операция VPL OTR, , A_{ii}); UNBL — уничтожение нулевых блоков (для $A_{ij} = 0$ дисковый адрес в дескрипторе устанавливается равным нулю и блок перестает реально существовать).

7.3. УМНОЖЕНИЕ МАТРИЦ

Эта группа операций включает в себя умножение матриц без какого-либо преобразования исходных операндов.

Умножение обычных матриц производится с помощью операторов:

VP &I, (&A, &B), &C или VPL &I, (&A, &B), &C

Оператор VPL используется в тех случаях, когда элементы результирующей матрицы C участвуют в образовании результата операции.

Если размеры матриц операндов равны $M \times N$ для A, $S \times T$ — для B, $K \times L$ — для C, то в зависимости от вида умножения производятся следующие проверки.

Д: НЕСООТВЕТСТВУЮЩИЕ ПРОВЕРКИ, если не выполняется хотя бы одно из условий: $(M=K)$, $(T=L)$, $(S=N)$ — для операций $A \times B$; $(M=K)$, $(S=L)$, $(N=T)$ — для $A \times B^{\tau}$; $(N=K)$, $(T=L)$, $(M=S)$ — для $A^{\tau} \times B$; $(N=K)$, $(S=L)$, $(M=T)$ — для $A^{\tau}B^{\tau}$; $(T=K=L)$, $(S=M=N)$ — для $B^{\tau}AB$.

Параметр & I может принимать следующие значения: UM — умножение матриц $(C=A \times B)$; UMS — умножение со сложением $(C=C+A \times B)$; UMV — умножение с вычитанием $(C=C-A \times B)$; UMIZ — умножение с изменением знака $(C=-A \times B)$; UMT — умножение на транспонированную матрицу $(C=A \times B^{\tau})$; UMTS — умножение на транспонированную матрицу со сложением $(C=C+A \times B^{\tau})$; UMTV — умножение на транспонированную матрицу с вычитанием $(C=C-A \times B^{\tau})$; TUM — умножение транспонированной матрицы $(C=A^{\tau} \times B)$; TUMS — умножение транспонированной матрицы со сложением $(C=C+A^{\tau} \times B)$; TUMV — умножение транспонированной матрицы с вычитанием $(C=C-A \times B)$; TUT — умножение транспонированных матриц $(C=A^{\tau}B^{\tau})$; TUTS — умножение транспонированных матриц со сложением $(C=C+A^{\tau}B^{\tau})$; TUTV — умножение транспонированных матриц с вычитанием $(C=C-A^{\tau}B^{\tau})$; 3UM — тройное умножение $(C=B^{\tau} \times A \times B)$; UMD — умножение на диагональную матрицу.

Операция UMD выполняется следующим образом. Среди матриц A и B, начиная с A, отыскивается матрица, у которой одна из размерностей равна 1, и принимается за диагональную. Если диагональная матрица не найдена, выдается диагностическое сообщение. Для случая, когда A — диагональная, $c_{ij} = a_i \times b_{ij}$ (все элементы строки i в матрице B умножаются на a_i). Производится диагностическая проверка: $(K=S)$; $(L=T)$, $(S=M$ для $N=1$, $S=N$ для $M=1)$. Для случая, когда B — диагональная, $c_{ij} = a_{ij}b_j$ (все эле-

менты столбца j в матрице A умножаются на b_j). Проверка: (K=M), (L=N), (N=S для T=1, N=T для S=1).

Такое большое разнообразие видов умножения потребовалось, главным образом, для создания эффективных программ обработки блочных матриц.

Для вычисления векторного произведения двух трехмерных векторов используется программа VEKP:

```
VP    VEKP,(& A, & B), & C
```

где & A, & B — операнды, а & C — результат векторного произведения.

Умножение блочных матриц производится аналогично умножению обычных матриц, где в качестве элементов матрицы выступают целые блоки. Операция производится с помощью операторов:

```
& BI      SASP  1, &A, &B, &C
```

Д: НЕВЕРНЫЕ РАЗМЕРЫ, если размеры метаматриц не соответствуют смыслу конкретного вида умножения.

Параметр & BI может принимать значения: BUM — блочное умножение матриц ($C=A\times B$); BUMT — блочное умножение на транспонированную матрицу ($C=A\times B^T$); BTUM — блочное умножение транспонированной матрицы ($C=A^T\times B$).

Для реализации операций $C=C+AB$ и $C=C-AB$ используется CA BUMM, обращение к которому имеет вид

```
BUMM      SASP  1, &A, &B, &C, &P
```

где & P — имя поля, содержащего константу вида

```
&P        DC    CL4' &I'
```

Здесь & I может принимать значение UMS, что соответствует операции $C=C+AB$, и значение UMV — при вычислении $C=C-AB$.

Для умножения блочной симметричной матрицы A на блочную матрицу B используется CA BUMS:

```
BUMS      SASP  1, &A, &B, &R, &C
```

где & R — имя поля, содержащего константу вида

```
&R        DC    C'&X'
```

Здесь &X — символ, который может принимать значение V, если &A хранится в виде верхней треугольной матрицы и N,— в виде нижней.

Для умножения блочной симметричной матрицы A, хранимой в верхней треугольной форме, на блочную транспонированную матрицу B ($C=AB^T$) используется CA BUTS

```
BUTS      SASP  1, &A, &B, &C
```

Для умножения блочных матриц, одна из которых является диагональной, используется программа BUMD

```
BUMD      SASP  1, &A, &B, &C
```

В процессе выполнения CA BUMD определяет, какая из матриц &A или &B является блочным столбцом (блочной строкой), блоки которого — простой столбец (простая строка). В нем хранится диагональная матрица. Если ни одна из матриц &A, &B не является диагональной или размеры матриц не соответствуют выполняемой операции, будет выдано сообщение НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ!

Для реализации «смешанного умножения», т. е. умножения блочной матрицы на простую вектор-строку, в результате которого формируется простая вектор-строка, используются программы SUT и SUTS, реализующие формулу $C^{т}=A\times B^{т}$.

```
&IM      SASP  1,&A,&B,&C
```

где &IM — имя программы: SUT или SUTS; &A — блочная матрица (для SUT) или блочная симметричная матрица, хранимая в виде верхнего треугольника (для SUTS); &B, &C — простые вектор-строки.

7.4. РЕШЕНИЕ СИСТЕМ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ. ОБРАЩЕНИЕ МАТРИЦ. ВЫЧИСЛЕНИЕ ОПРЕДЕЛИТЕЛЯ

В СМПО имеется группа программ, предназначенных для решения системы алгебраических уравнений $AX=B$ и для обращения матрицы A. В зависимости от вида матрицы A следует предпочесть тот или иной алгоритм.

Обращение несимметричной матрицы и решение несимметричной системы алгебраических уравнений $AX=B$ для простых матриц выполняется методом Гаусса с поиском максимального ведущего элемента по столбцам. Обращение к этим операциям соответственно имеет вид

```
VPL   OBR,,  &A
VPL   RSU,(&A),&B
```

где &A — имя матрицы коэффициентов системы размером $M\times N$; &B — имя матрицы правой части системы размером $S\times T$.

Результаты операций образуются на месте матрицы A при обращении матрицы и матрицы B при решении системы уравнений.

Д: НЕКВАДРАТНАЯ МАТРИЦА!, если $M\neq N$.

НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ!, если $N\neq S$.

ВЫРОЖДЕННАЯ МАТРИЦА!, если максимальный ведущий элемент при обращении матрицы имеет значения меньше 10^{-20}.

Для вычисления определителя простой несимметричной матрицы используется алгоритм триангуляризации [36]:

```
V     DET,(&A),&D
```

где &A — имя матрицы, для которой вычисляется определитель; &D — имя поля размером в двойное слово, в которое программа DET поместит значение определителя матрицы &A.

Обращение несимметричной блочной матрицы A выполняется следующим образом:

```
BOBR     SASP  1, &A
```

Д: НЕВЕРНЫЕ РАЗМЕРЫ, если $M \neq N$.

НУЛЕВОЙ БЛОК, если на диагонали блочной матрицы находится нулевой блок.

Для решения несимметричной системы алгебраических уравнений для блочных матриц используется программа BRSU

```
BRSU     SASP  1, &A, &B
```

Д: НЕВЕРНЫЕ РАЗМЕРЫ, если $M \neq N$, $M \neq S$.

НУЛЕВОЙ БЛОК, если на диагонали блочной матрицы находится нулевой блок.

Для простых положительно определенных симметричных матриц реализован алгоритм решения системы уравнений, основанный на LL^T-разложении Холецкого [36]:

```
RXL      SASP  1, &A, NET
```

Правильное расположение элементов матрицы &A необходимо только в ее верхнем треугольнике. В процессе разложения элементы &A заменяются элементами $\& L^T$. Нижний треугольник не используется и не изменяется.

Д: НЕКВАДРАТНАЯ МАТРИЦА!, если $M \neq N$.

ВЫРОЖДЕННАЯ МАТРИЦА!, если в процессе разложения диагональный элемент равен 0.

МАТРИЦА НЕПОЛОЖИТЕЛЬНО-ОПРЕДЕЛЕННАЯ! НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ, если в процессе разложения диагональный элемент отрицателен.

Последние два вида диагностики сопровождаются сообщением РАЗЛОЖЕНИЕ ХОЛЕЦКОГО ОСТАВЛЕНО В *ааа* СТРОКЕ, где *ааа* — номер строки, в которой обнаружена ошибка.

Для решения системы уравнений после разложения Холецкого используется программа RSS

```
RSS      SASP  1, & A, & B
```

где &A — матрица размером $M \times N$, полученная из матрицы системы уравнений в результате работы программы RXL; &B — матрица правых частей размером $S \times T$.

Решение получается на месте матрицы &B.

Д: НЕКВАДРАТНАЯ МАТРИЦА!, если $M \neq N$.

НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ!, если $M \neq S$.

Для обращения простых положительно определенных матриц используется алгоритм, основанный на LDL^T-разложении Холецкого [36]:

```
VPL   OBRX ,, & A
```

где &A — обращаемая матрица, правильное расположение элементов которой требуется только в ее верхнем треугольнике.

Результат получается на месте & A в виде полной симметричной матрицы.

В процессе обращения в нижней треугольной части матрицы A накапливаются элементы L и D^{-1}, затем L заменяется на L^{-1}, а потом формируется A^{-1}.

Диагностические сообщения те же, что у программы RXL.

Для блочных положительно определенных симметричных матриц реализовано два алгоритма решения системы уравнений, основанных на алгоритме Холецкого. Первый алгоритм основан на LL^T-разложении и образует матрицу L^T на месте исходной

```
BRXL     SASP  1, & A
```

где & A — матрица системы уравнений, правильное расположение элементов которой необходимо только в верхнем ее треугольнике.

Д: НЕКВАДРАТНАЯ МАТРИЦА!, если $M \neq N$.

ВЫРОЖДЕННАЯ МАТРИЦА!, если в процессе разложения диагональный элемент равен 0.

МАТРИЦА НЕПОЛОЖИТЕЛЬНО-ОПРЕДЕЛЕННАЯ! НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ!, если в процессе разложения диагональный элемент отрицателен.

Последние два вида диагностики сопровождаются сообщением РАЗЛОЖЕНИЕ ХОЛЕЦКОГО ОСТАНОВЛЕНО В *ааа* СТРОКЕ *ббб* БЛОЧНОЙ СТРОКИ, где *ааа* — номер строки в *ббб* блочной строке, в которой обнаружена ошибка.

При указании режима KONTR 2 через каждые 10 обработанных суперстрок печатаются текущее время и структура матрицы.

Для решения блочной системы уравнений после разложения Холецкого используется программа BSS

```
BSS      SASP  1, &A, &B
```

где & A — блочная матрица размером $M \times N$, полученная из матрицы системы уравнений программой BRXL; & B — блочная матрица правых частей размером $S \times T$, на месте которой получается решение системы.

Диагностические сообщения те же, что и в программе RSS.

Программы BRXL и BSS используют специальные подпрограммы разреженного умножения, позволяющие учесть разреженность не только на блочном уровне, но и внутри блоков [17, 30], что для типичных задач МКЭ дает ускорение в 1,5—2 раза.

Второй алгоритм разложения Холецкого для блочных матриц основан на блочном LDL^T — разложении и сохраняет исходную матрицу, формируя результат на новом месте:

```
& FC     SPFOR ( &C)
RXOL     SASP 1, &A, &FC
```

где & A — блочная матрица системы уравнений размером $M \times N$, правильное расположение элементов которой требуется только в ее верхнем треугольнике; & FC — имя списка, содержащего фактический результат; & C — блочная нижняя треугольная матрица, в которой хранятся блоки матриц L и D^{-1}.

Д: НЕВЕРНЫЕ РАЗМЕРЫ!, если $M \neq N$.

НУЛЕВОЙ БЛОК!, если на диагонали блочной матрицы A имеется нулевой блок.

ВЫРОЖДЕННАЯ МАТРИЦА!, если в процессе разложения диагональный элемент равен 0.

НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ!, если в процессе разложения диагональный элемент меньше 0.

Последние два вида диагностики сопровождаются сообщением РАЗЛОЖЕНИЕ ХОЛЕЦКОГО ОСТАНОВЛЕНО В *ааа* СТРОКЕ *ббб* БЛОЧНОЙ СТРОКИ, где *ааа* — номер строки в ббб блочной строке, в которой обнаружена ошибка.

При указании режима KONTR 2 через каждые 10 обработанных суперстрок печатаются текущее время и структура матрицы.

Обращение исходной матрицы A выполняется после разложения Холецкого ($A^{-1} = (L^{т})^{-1}D^{-1}L^{-1}$)

```
BOBS     SASP  1, &C
```

где &C — матрица размером $N \times N$, полученная программой RXOL.

В процессе обращения в нижней треугольной части матрицы &C образуется сначала обратная матрица L^{-1}, а затем обратная матрица A^{-1}.

Решение системы уравнений $AX = B$ после разложения Холецкого получается в результате последовательного решения систем $LY = B$ и $DL^{т}X = Y$:

```
BRSS     SASP  1, &C, &B
```

где &C — блочная матрица блочным размером $N \times N$, полученная в результате работы алгоритма RXOL; &B — блочная матрица правых частей блочным размером $S \times T$, на месте которой получается решение системы.

Д: НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ, если $S \neq N$.

Для решения симметричных, не обязательно положительно-определенных систем уравнений, разработаны алгоритмы, основанные на $LDL^{т}$-разложении Холецкого. В книге [29] показано, что такое разложение без перестановки строк, нарушающей ленточную и профильную структуры или симметрию разреженных матриц, может быть выполнено, если все главные миноры невырождены. Разложение неустойчиво, если какой-либо из этих миноров имеет близкое к нулю собственное значение. На практике это обстоятельство трудно проверить, не выполняя разложения. Опыт показывает, что срыв процесса разложения бывает крайне редко. К тому же, неположительно-определенные симметричные матрицы возникают обычно при решении задач вида $(A - \sigma)X = B$ либо $(A - \sigma M)X = B$, где σ — скаляр, значение которого может быть взято приближенно. Это подсказывает следующую простую стратегию: если разложение потерпело неудачу, можно изменить σ на 0,01% и повторить процесс.

Для LDL^T-разложения простых симметричных неположительно-определенных матриц используется программа RXN

```
VPL    RXN, (&M, &P), &A
```

где &A — матрица системы уравнений размером $M \times N$, правильное расположение элементов которой необходимо только в верхнем ее треугольнике; &M — имя поля, содержащего число двойной точности, равное наибольшему по модулю элементу &A. Оно может быть получено предварительно с помощью программы MAXA; &P — имя поля, содержащего целое число в формате полуслова, вырабатываемое программой RXN. В случае успешного разложения оно будет равно нулю. При неудаче оно будет равно номеру строки, в которой разложение остановлено.

После выполнения программы RXN нужно проанализировать величину &P, чтобы проверить успешность ее завершения.

В процессе разложения элементы &A заменяются элементами L^T и D^{-1}. Нижний треугольник матрицы не используется и не изменяется.

Д: НЕКВАДРАТНАЯ МАТРИЦА!, если $M \neq N$.

Для решения системы уравнений после разложения, выполненного программой RXN, используется программа

```
RSSN     SASP  1, &A, &B
```

где &A — матрица размером $M \times N$, полученная из матрицы системы уравнений в результате работы программы RXN; &B — матрица правых частей размером $S \times T$.

Решение получается на месте матрицы &B.

Диагностические сообщения те же, что и в программе RSS.

Для LDL^T-разложения блочных симметричных неположительно-определенных матриц используется программа BRXN

```
BRXN     SASP  1, &A, &P
```

где &A — блочная матрица системы уравнений размером $M \times N$, правильное расположение элементов которой необходимо только в верхнем ее треугольнике; &P — имя поля длиной в слово (4 байта), в которое программа BRXN поместит два числа в формате полуслова. Они будут равны нулю при успешном разложении. При неудаче первое число будет равно номеру блочной строки, а второе — номеру строки в блоке, в которой разложение остановлено.

При неудачном разложении будет напечатано сообщение РАЗЛОЖЕНИЕ ХОЛЕЦКОГО ОСТАНОВЛЕНО В *ааа* СТРОКЕ *ббб* БЛОЧНОЙ СТРОКИ, где *ааа* и *ббб* — координаты строки, в которой прекращено разложение.

В процессе разложения элементы матрицы A заменяются элементами L^T и D^{-1}.

Д: НЕКВАДРАТНАЯ МАТРИЦА, если $M \neq N$.

Для решения системы уравнений после разложения, выполненного программой BRXN, используется программа BSSN

```
BSSN     SASP  1, &A, &B
```

где &A — блочная матрица размером M×N, полученная из матрицы системы уравнений программой BRXN; &B — блочная матрица правых частей размером S×T, на месте которой получается решение системы.

Диагностические сообщения те же, что и в программе RSS.

В ППП СМПО имеются программы, предназначенные для решения «смешанных» симметричных систем уравнений, т. е. систем вида AX=B, где A — блочная матрица, а B — простая, причем правая часть системы уравнений хранится в матрице B в транспонированном виде. Применение этих программ предпочтительнее, если матрица правых частей целиком размещается в ОП.

После того как матрица системы уравнений разложена программой BRXL, для решения «смешанной» системы можно использовать программу SRSS

```
SRSS     SASP  1, &A, &B
```

где &A — блочная матрица блочным размером M×N, абсолютным размером K×K, полученная из матрицы системы уравнений программой BRXL; &B — простая матрица правых частей размером S×T, на месте которой получается решение системы.

Д: НЕКВАДРАТНАЯ МАТРИЦА!, если M≠N.
НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ!, если T≠K.

После того как матрица системы уравнений разложена программой BRXN, для решения «смешанной» системы можно использовать программу SRSN

```
SRSN     SASP  1, &A, &B
```

где &A — блочная матрица блочным размером M×N, абсолютным размером K×K, полученная из матрицы системы уравнений программой BRXN; &B — простая матрица правых частей размером S×T, на месте которой получается решение системы.

Диагностическое сообщение то же, что и в программе SRSS.

В ППП СМПО имеются программы, предназначенные для решения треугольных систем уравнений. Сами треугольные системы могут быть получены как непосредственно, так и в результате разложения Холецкого. Программы, решающие треугольные системы, предполагают, что матрицы систем хранятся в виде верхних треугольных матриц, но это — только форма хранения. Имеются программы решения как для систем вида LX=B, так и для систем вида UX=B (здесь L — нижняя, а U — верхняя треугольная матрица). Программы, решающие системы LX=B, используют в качестве L матрицу U^T.

Для решения нижней (верхней) простой треугольной системы используется программа RTS (RTTS)

```
VPL   RTS, (&A), &X
VPL   RTTS, (&A), &X
```

где &A — простая матрица системы уравнений размером M×N, хранящаяся в виде верхней треугольной матрицы; &X — простая

матрица правых частей размером S×T, на месте которой получается решение системы.

Нижний треугольник матрицы &A не используется и не изменяется.

Диагностические сообщения те же, что и в программе RSS.

Для решения нижней (верхней) блочной треугольной системы используется программа BRT (BRTT)

```
BRT     SASP  1,&A,&B
BRTT    SASP  1,&A,&B
```

где &A — блочная матрица системы уравнений размером M×N, хранящаяся в виде верхней треугольной матрицы; &B — блочная матрица правых частей размером S×T, на месте которой получается решение системы.

Нижний треугольник матрицы &A не используется и не изменяется.

Диагностические сообщения те же, что и в программе RSS.

Для решения нижней (верхней) смешанной треугольной системы используется программа SRT (SRTT)

```
SRT     SASP  1,&A, &B
SRTT    SASP  1,&A,&B
```

где &A — блочная матрица блочным размером M×N, абсолютным размером K×K, хранящаяся в виде верхней треугольной матрицы; &B — простая матрица правых частей размером S×T, на месте которой получается решение системы.

Нижний треугольник матрицы &A не используется и не изменяется.

Диагностические сообщения те же, что и в программе SSS.

Для решения нижней (верхней) простой треугольной системы с единичной диагональю используется программа RTN (RTTE)

```
VPL   RTN,(&A),&X
VPL   RTTE,(&A), &X
```

где &A — простая матрица системы уравнений размером M×N, хранящаяся в виде верхней треугольной матрицы; &X — простая матрица правых частей размером S×T, на месте которой получается решение системы.

Нижний треугольник и диагональ матрицы &A не используются и не изменяются, формировать в явном виде единичную диагональ нет необходимости.

Диагностические сообщения те же, что и в программе RSS.

При решении систем уравнений методом Холецкого важное значение приобретает уменьшение профиля матрицы системы уравнений (рис. 7.1) благодаря удачной нумерации узлов. Это позволяет существенно уменьшить затраты памяти и вычислительную работу при разложении и решении. В состав ППП СУМРАК входит программа, позволяющая перед формированием матрицы жестко-

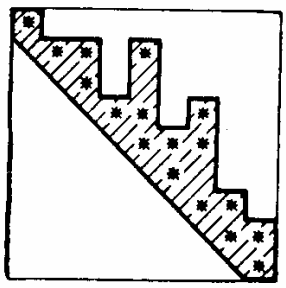


Рис. 7.1. Профиль матрицы (заштрихован) при решении системы уравнений:

* — ненулевые элементы

сти конструкции перенумеровать ее узлы для уменьшения профиля. В принципе эта программа может быть применена и для уменьшения профиля произвольной матрицы, если ее структура может быть описана с помощью двух-, трех- и четырехузловых списков связности. Двухузловой список можно мысленно представить как матрицу D размером N×2, каждой строке которой соответствует ненулевой элемент матрицы A с координатами (D (i, 1), D (i, 2). Трехузловой список отображается на матрицу T [M×3], каждой строке которой соответствуют ненулевые элементы с координатами (T (i, 1), T (i, 2)), (T (i, 1), T (i, 3)), (T (i, 2), T (i, 3)). Четырехузловой список отображается на матрицу H [K×4], каждой строке которой соответствуют $C_4^2=6$ ненулевых элементов матрицы A. При этом каждому узлу может соответствовать не один, а несколько элементов матрицы A. Так, например, если в методе перемещений одному узлу соответствует 6 степеней свободы, то одна строка D описывает уже целую подматрицу размерами 6×6. Но для описываемой программы перенумерации кратность узлов не имеет значения.

Выбор хорошей в смысле размера профиля нумерации узлов осуществляет CA RCM, реализующий обратный алгоритм Катхилла — Макки [17]:

```
&FRCM     SPFOR (&INVP)

RCM       SASP  1,&NUZ,&T,&H,&D,&FRCM
```

где &NUZ — имя поля в формате полуслова, содержащего количество узлов, т. е. размер/кратность матрицы A; &T — сложный список, заголовок которого содержит имена простых статических или динамических трехузловых списков, используемых в качестве подсписков сложного списка. Если трехузловые списки отсутствуют, параметр &T может быть задан как NET; &D — сложный список, заголовок которого содержит имена простых статических или динамических двухузловых списков, используемых в качестве подсписков сложного списка. Если двухузловые списки отсутствуют, параметр &D может быть задан как NET; &H — сложный список, заголовок которого содержит имена простых статических или динамических четырехузловых списков, используемых в качестве подсписков сложного списка. Если четырехузловые списки отсутствуют, параметр &H может быть задан как NET; &FRCM — имя списка, содержащего фактический результат; &INVP — результат работы CA RCM, представляющий собой простой динамический список. i-й элемент этого списка равен номеру i-го узла в новой нумерации.

Алгоритм RCM дает не оптимальное, но близкое к нему в смысле размера профиля упорядочение. После получения списка &INVP

можно формировать матрицу системы уравнений, используя новую нумерацию.

Рассмотрим подробнее организацию и логику работы программы BRXL, типичной при обработке больших разреженных матриц.

На входе у программы BRXL — квадратная матрица A, представленная в универсальной блочной форме. В отличие от профильных и компактных схем хранения [17] блочная схема более проста и, что очень важно, естественна и эффективна по времени при формировании матрицы. Например, при рассылке матриц конечных элементов в глобальную матрицу жесткости в конечноэлементном методе перемещений активная зона работы может включать несколько участков глобальной матрицы, которые вдобавок постепенно перемещаются. Профильная схема здесь менее удобна. Кроме того, матрица A может быть результатом больших матричных вычислений, как в конечноэлементном методе сил, что делает профильную или разреженную схему малоприемлемой. Поэтому наша задача состоит в том, чтобы обеспечить эффективность разложения, если не равную, то во всяком случае приближающуюся к эффективности схем, предложенных в работе [17].

Для разложения применяется блочно-профильный метод. В процессе разложения может происходить заполнение профиля как на уровне блоков, так и на уровне отдельных элементов. Вновь создаваемые блоки, в соответствии с идеологией СМПО, будут «определяться», т. е. создаваться на диске. Желательно так организовать процесс вычислений, чтобы вновь определяемые блоки создавались «по строкам», а не «по столбцам», т. е. сначала должны быть созданы новые блоки 1-й строки, затем второй и т. п. Такой порядок создания новых блоков важен не столько на этапе разложения, как на этапе решения системы, где матрица A последовательно, по строкам считывается в оперативную память, что дает возможность уменьшить временные затраты на перемещение гребенчатого механизма доступа магнитного диска. Этому требованию удовлетворяет следующий алгоритм, который можно принять за основу (пока в нем никак не учитывается разреженность матриц).

Алгоритм 7.4.1

Для $i = 1, 2, \ldots, N$

Для $j = i, i+1, \ldots, N$

$$A_{ij} = A_{ij} - \sum_{k=1}^{i-1} A_{ki}^{\mathrm{T}} A_{kj}$$

Если $i = j$, то $A_{ii} = \mathrm{RXL}(A_{ii})$

иначе $A_{ij} = \mathrm{RTS}(A_{ii}, A_{ij})$

конец j

конец i.

Здесь RXL означает необходимость разложить диагональный блок методом Холецкого, а обозначение RTS (A_i, A_{ij}) — необходи-

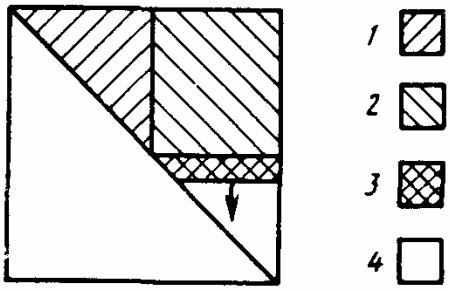


Рис. 7.2. Схема процесса разложения, выполняемого программой BRXL:

1 — дальнейших обращений не требуется; *2* — эта часть вычислена; доступ к ней открыт; *3* — доступ открыт, происходит перестройка; *4* — обращений пока не происходило

мость решить нижнюю треугольную систему вида $A_{ii}X=A_{ij}$, поместив результат на место A_{ij}.

Порядок разложения можно иллюстрировать схемой, показанной на рис. 7.2.

Теперь введем в алгоритм учет разреженности матрицы A на уровне блоков. Напомним, что нулевые блоки не хранятся и не создаются: их просто не существует. Проверить существование блока можно двумя командами, поэтому затраты времени на эти проверки незначительны.

Алгоритм 7.4.2.

Для $i=1, 2, \ldots, N$

  Для $j=i, i+1, \ldots, N$

    Для $k=i-1, i-2, \ldots, 1$

      Если A_{ki} и A_{kj} существуют, то

        Если A_{ij} не существует, то определяем ее. Конец — если

        $A_{ij}=A_{ij}-A^{\mathrm{T}}_{ki}A_{kj}$

      конец — если

    конец k

    Если $i=j$, то $A_{ii}=\mathrm{RXL}(A_{ii})$

    иначе

        Если A_{ij} существует, то $A_{ij}=\mathrm{RTS}(A_{ii}, A_{ij})$. Конец — если.

  конец j

конец i

Анализируя этот алгоритм, видим, что наибольшее время тратится на вычисление $A_{ij}=A_{ij}-A^{\mathrm{T}}{}_{ki}A_{kj}$, которое находится во внутреннем цикле и выполняется $N^3/6$ раз, если все блоки матрицы A существуют. Так как блоки A_{ki} и A_{kj} являются сильно разреженными и после разложения, нужно учесть их разреженность при вычислении произведения. Эту операцию выполняет программа TURV, алгоритм работы которой мы сейчас рассмотрим.

Итак, пусть имеются матрицы A[M×N], B[M×L] и C[N×L] и требуется вычислить произведение C=C—A^{T}B, причем матрицы A и B — разреженные. Специфика поставленной задачи в том, что, если просто сравнивать с нулем сомножители перед выполнением произведения, слепо применяя формулу $c_{ij}=c_{ij}-\sum_{k=1}^{M} a_{ki}b_{kj}$, время

выполнения операции может даже возрасти за счет дополнительных операций сравнения. Возможен другой порядок выполнения операций.

Алгоритм 7.4.3

Для $i = 1, 2, \ldots, M$

  Для $j = 1, 2, \ldots, L$

    Если $b_{ij} \neq 0$,

    то

      Для $k = 1, 2, \ldots, N$

        $c_{kj} = c_{kj} - a_{ik} b_{ij}$

      конец k

  конец j

конец i

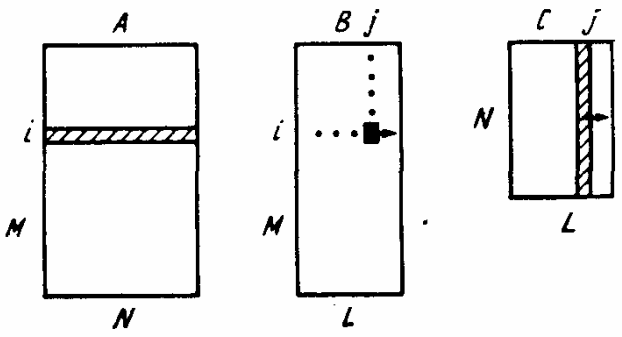


Рис. 7.3. Схема разреженного умножения матриц

Работа этого алгоритма показана на рис. 7.3.

Его реализация по сравнению со стандартным алгоритмом требует выполнения ML сравнений, что является незначительной затратой времени по сравнению с MNL операцией умножения, требуемых для вычисления произведения заполненных матриц (предполагается, что $N > 1$). Зато каждое сравнение, если элемент окажется равным нулю, экономит N умножений.

Приведенный алгоритм весьма эффективен, если разреженной является только матрица B. В нашем случае разреженными являются оба сомножителя. Анализируя рис. 7.3, можно установить, что i-я строка матрицы A последовательно умножается на ненулевые элементы i-й строки матрицы B, а результат этого умножения вычитается из содержимого каждый раз нового, j-го столбца матрицы C. Поэтому, если в i-й строке матрицы B есть ненулевые элементы, можно упаковать i-ю строку матрицы A, т. е. переписать только ненулевые ее элементы в дополнительный участок памяти; одновременно нужно определить смещения соответствующих им элементов столбца матрицы C от начала столбца (эти смещения будут одинаковы для всех столбцов C). После выполненной упаковки умножаться будут только ненулевые элементы. Для выполнения алгоритма требуется дополнительная память: 8N байт для хранения строки A и 4N байт для хранения столбца смещений — всего 12N байт. Этот участок памяти выделяется один раз: в начале работы программы BRXL определяются наибольшие размеры блока и выделяется рабочая область, которая передается программе TURV при каждом ее выполнении.

Опишем полностью алгоритм работы программы TURV.

Алгоритм 7.4.4

Для $i = 1, 2, \ldots, N$

  $P = 0$

```
Для j = 1, 2, ..., L
  Если b_lj ≠ 0,
  то
    Если P = 0,
    то
      Если j = L,
      то
        Для k = 1, 2, ..., N
          c_kj = c_kj − a_lk b_lj
        конец k
      иначе
        упаковываем строку i матрицы A
        P = 1
      конец — если
    конец — если
    Если P = 1,
    то
      Для k = 1, 2, ..., N
        c_kj = c_kj − [a_lk упакованное] b_lj
      конец k
    конец — если
  конец — если
конец — j
конец i
```

Описываемый алгоритм выполняет упаковку матрицы A только в том случае, когда первый ненулевой элемент i-й строки B не является последним элементом этой строки, иначе временные затраты на упаковку не будут оправданы.

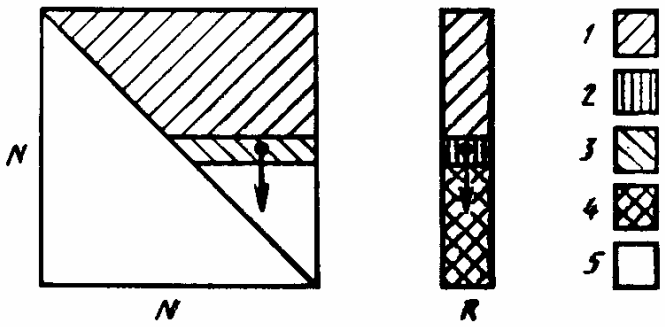
Алгоритм может быть реализован только на языке ассемблер, поскольку на языке более высокого уровня невозможна эффективная организация умножения упакованной строки на b_{ij} с записью результата в соответствующие элементы j-го столбца матрицы C.

Хронометраж показал, что если матрицы A и B заполнены на 50%, учет разреженности матрицы B дает ускорение в 1,7 раза, а учет разреженности обеих матриц — в 2 раза. При разложении Холецкого часто встречаются и более разреженные матрицы.

Анализируя алгоритм 7.4.2, можно определить, что следующий резерв экономии — эффективная организация работы программы RTS, которая выполняется $\approx N^2/2$ раз, если все блоки матрицы A существуют.

Рис. 7.4. Решение треугольной системы уравнений по схеме внешних произведений:

1 — дальнейших обращений не требуется; *2* — эта часть вычислена, доступ к ней открыт; *3* — участвует в вычислениях; *4* — доступ открыт, происходит перестройка; *5* — обращений пока не происходило



Вычисления в программе RTS, реализующей решение нижней простой треугольной системы AX=B, где A — матрица размером N×N, хранимая в транспонированном виде; B — матрица размером N×R, можно организовать по схеме внешних произведений [17], позволяющей учесть разреженность правой части (сама матрица системы уравнений, являющаяся диагональным блоком блочной системы, обычно бывает почти полностью заполненной). Рассмотрим алгоритм решения по схеме внешних произведений.

Алгоритм 7.4.5 (рис. 7.4)

Для $j = 1, 2, \ldots, R$

  Для $i = 1, 2, \ldots, N$

    Если $b_{ij} \neq 0$,

    то

      $b_{ij} = b_{ij}/a_{ii}$

      Для $k = i + 1, i + 2, \ldots, N$

        $b_{kj} = b_{kj} - a_{ik} b_{ij}$

      конец k

    конец — если

  конец — i

конец j

Алгоритм требует выполнения NR операций сравнения чисел с плавающей точкой, что является недорогой ценой по сравнению с $\frac{1}{2}N^2R$ операциями умножения, требуемыми для решения системы. Зато каждое сравнение, если элемент b_{ij} окажется нулевым, экономит в среднем N/2 умножений.

Еще один резерв в алгоритме 7.4.2 связан с выполнением операции $A_{ij} = A_{ij} - A_{ki}^{\mathrm{T}} A_{kj}$ при $i = j$. В этом случае нет необходимости полностью вычислять матрицу A_{ii}, достаточно вычислить лишь верхний треугольник этой симметричной матрицы. Эту операцию выполняет программа TUSV, алгоритм работы которой основан на алгоритме 7.4.3.

Из итерационных методов решения системы уравнений в ППП СМПО реализован алгоритм метода сопряженных градиентов

[36] — алгоритм SOGR. Этот алгоритм может обрабатывать симметричные положительно определенные матрицы любого вида благодаря тому, что матрица системы уравнений не требует явного задания. Вместо этого в качестве одного из параметров алгоритма задается имя стандартного алгоритма, вычисляющего произведение вида AX, где X — простая вектор-строка. Параметры, необходимые для работы этого алгоритма, указываются при обращении к программе SOGR.

Обращение к программе SOGR имеет следующий вид:

```
&FSOGR    SPFOR (&X)

SOGR    SASP   1,&B,&NAP,&N,&FSOGR,&OP,&A1,&A2,&A3
```

где &B — правая часть системы уравнений, заданная в виде простой вектор-строки; &NAP — начальное приближение к решению уравнения, заданное в виде простой вектор-строки. Если хорошее начальное приближение неизвестно, оно должно быть задано как NET; &N — имя поля, содержащего число в формате полуслова, равное максимально допустимому числу итераций; &FSOGR — имя списка фактического результата; &OP — имя поля, содержащего название стандартного алгоритма, вычисляющего произведение вида AX для произвольной простой вектор-строки X; &X — результат решения системы, получающейся в виде простой вектор-строки; &A1, &A2, &A3 — параметры, передаваемые без изменения программе &OP для вычисления произведения AX.

Программа &OP должна быть оформлена в соответствии со следующими правилами. Имя &OP должно задаваться в неисполняемой части программы в виде &OP DC CL4'&NAME', где &NAME — имя стандартного алгоритма, вычисляющего произведение AX.

Алгоритм &NAME должен содержать от трех до пяти параметров и вычислять произведение Y=AX. Первый передаваемый ему параметр — &A1, второй — вектор-строка &X, содержащий значение вектора X, третий — вектор-строка &Y, в котором должен быть получен результат Y. Эти три параметра — обязательные. Параметры &A2 (четвертый) и &A3 (пятый) — необязательные. Например, если матрица A задана в виде блочной симметричной матрицы, в качестве &NAME можно использовать программу SUTS. В этом случае параметры &A2 и &A3 не нужны. Вместо них надо задать NET. Алгоритм &NAME не должен разрушать значение &X.

Поскольку программе SOGR неизвестно, сколько задано параметров &A1, &A2, &A3 и какие из них являются матрицами, она их не запоминает. Для повышения производительности рекомендуется матрицы (метаматрицы) запомнить перед использованием алгоритма.

Д: CA SOGR. МАТРИЦА НЕ ОПРЕДЕЛЕНА ПОЛОЖИТЕЛЬНО. НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ!, если A — неположительно-определенная матрица.

CA SOGR. ТОЧНОСТЬ НЕ ДОСТИГНУТА, если выполнено заданное количество операций, но процесс не сошелся.

Рассмотрим пример использования программы SOGR.

Пример 7.4.1

Пусть матрица A задана в виде блочной симметричной матрицы (хранится только верхний треугольник). Хорошее начальное приближение неизвестно. Число итераций ограничено пятьюдесятью. В качестве &NAME используется программа SUTS

```
                ZAPOM (A)
FSOGR           SPFOR (X)
SOGR            SASP  1, B, NET, N, FSOGR, OP, A, NET, NET
                UDAL  1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
N               DC    H'50'
OP              DC    CL4'SUTS'
```

Здесь в матрице B записана правая часть. Результат работы программы SOGR помещается в матрицу X.

Требуемое для сходимости число итераций достаточно трудно оценить заранее. Если система уравнений не очень плохо обусловлена, приемлемой является величина $N=2M$, где M — размерность системы уравнений. В более сложных случаях может потребоваться большее число шагов, но если значение $N=5M$ недостаточно, то решение в принципе получить довольно трудно. Кроме того, для решения системы с хорошо обусловленной матрицей требуется гораздо меньше шагов, особенно для больших N [36].

Иногда имеется возможность значительно уменьшить число обусловленности и, следовательно, требуемое количество итераций, используя так называемое предобусловливание [53]. Пусть есть матрица $\tilde{A}$, близкая к матрице $\tilde{A}$. Пусть для матрицы $\tilde{A}$ известно разложение Холецкого $A=LL^{\tau}$. Тогда умножая уравнение

$$AX=B \tag{7.1}$$

слева на L^{-1} и делая замену переменной

$$X=L^{-\tau}Y, \tag{7.2}$$

где $L^{-\tau}=(L^{\tau})^{-1}$, получим

$$L^{-1}AL^{-\tau}Y=L^{-1}B.$$

Обозначая

$$C=L^{-1}AL^{-\tau};\ D=L^{-1}B, \tag{7.3}$$

получим новую систему уравнений

$$CY=D. \tag{7.4}$$

Матрица C будет симметричной и положительно-определенной, если таковой является матрица A, но она значительно лучше обусловлена. Решив систему (7.4), можно восстановить значение X из (7.2). Явно вычислять матрицу C нет необходимости, так как в

процессе решения системы уравнений требуется лишь вычислить произведения вида

$$G=L^{-1}AL^{-T}F \tag{7.5}$$

для произвольного вектора E. Произведение (7.5) можно вычислять последовательно справа налево

$$F_1=L^{-T}F; \tag{7.6}$$

$$F_2=AF_1; \tag{7.7}$$

$$G=L^{-1}F_2. \tag{7.8}$$

Для вычисления F_1 по (7.6), нужно решить треугольную систему

$$L^TF_1=F,$$

а для вычисления F_2 по (7.8),— систему

$$LG=F_2.$$

Таким образом, произведение (7.5) можно вычислить так:

1) решить систему $L^TF_1=E$ относительно F_1;
2) вычислить $F_2=AF_1$;
3) решить систему $LG=F_2$ относительно G.

Если матрица $R=A-\tilde{A}$ является сильно разреженной, G можно вычислить по следующей формуле:

$$G=L^{-1}(R+\tilde{A})L^{-T}F=L^{-1}RL^{-T}F+F. \tag{7.9}$$

Если матрицы A и $\tilde{A}$ весьма близки, то число итераций, нужных для отыскания решения, будет исчисляться единицами или десятками. СА SOGR благодаря гибкости задания функции AX допускает использование любого варианта предобусловливания.

Комплекс программ для решения систем линейных алгебраических уравнений включает в себя программы различной производительности, решающие различные типы систем. Дадим рекомендации по их использованию в зависимости от свойств матрицы системы уравнений.

1. Симметричные положительно-определенные матрицы. В этом случае могут применяться программы, основанные на разложении Холецкого (RXL, BRXL, RXOL), и программа метода сопряженных градиентов (SOGR). Основным инструментом следует считать программу BRXL, способную эффективно учесть ленточную или профильную структуру, а также разреженность матрицы. К тому же она имеет то важнейшее преимущество, что разложенная матрица может неоднократно использоваться для решения систем с различными правыми частями. Поскольку основные временные затраты падают на этап разложения, при наличии большого числа правых частей метод Холецкого находится вне конкуренции. После разложения, выполненного программой BRXL, решение системы можно осуществить с помощью программы BSS или SRSS. Вторая программа — предпочтительнее, если правая часть системы целиком

размещается в оперативной памяти. Если матрица системы сильно разрежена, но будет испытывать значительное заполнение при разложении, удобно применять алгоритм SOGR. Он особенно выгоден, если можно эффективно вычислять произведение AX. Так, многие задачи математической физики приводят к положительно-определенным матрицам с небольшим числом ненулевых элементов, расположенных в разных местах матрицы. Эти элементы (a_{ij}) можно вычислить как функцию параметров i, j. Последнее имеет место при переходе от дифференциальных уравнений в частных производных к разностным. Если матрица A — редкая, а функции переменных i, j достаточно просты, алгоритм SOGR может оказаться весьма эффективным. Алгоритм SOGR может применяться также при наличии хорошего приближения к решению и (или) по схеме предобусловливания, если известен фактор Холецкого для матрицы, близкой к A.

2. Симметричные неположительно-определенные матрицы. В этом случае могут применяться программы, основанные на LDL^{τ}-разложении Холецкого (RXN, BRXN). По своей эффективности они приближаются к алгоритмам решения положительно-определенных систем (RXL, BRXL), но могут потерпеть неудачу при разложении. Тем не менее рекомендуется применять именно методы Холецкого, предусматривая при неудаче незначительное изменение системы, либо использование программ для матриц общего вида (RSU, BRSU). После разложения, выполненного программой BRXN, решение системы можно осуществить с помощью программы BSSN или SRSN. Вторая программа выполняется быстрее, если правая часть системы целиком размещается в ОП.

3. Треугольные матрицы. Программы решения систем уравнений ППП СМПО предполагают, что треугольные матрицы хранятся в виде верхних треугольных матриц. В противном случае перед решением системы их необходимо транспонировать. Решать системы можно с помощью программ RTS, RTTS, BRT, BRTT, SRT, SRTT. Если матрица треугольной системы имеет единичную диагональ, ее можно явно не формировать, и решать системы программами RTN, RTTE.

4. Матрицы общего вида. В этом случае могут применяться программы RSU и BRSU. Кроме того, используя гауссову трансформацию, систему AX=B можно привести к виду

$$\mathbf{A}^{\tau}\mathbf{A}\mathbf{X}=\mathbf{A}^{\tau}\mathbf{B}, \qquad (7.10)$$

где $\mathbf{A}^{\tau}\mathbf{A}$ — матрица новой системы уравнений (7.10), которая является симметричной и положительно-определенной. Следовательно, для ее решения могут быть применены методы, указанные в п. 1.

7.5. РЕШЕНИЕ АЛГЕБРАИЧЕСКОЙ ПРОБЛЕМЫ СОБСТВЕННЫХ ЗНАЧЕНИЙ

В СМПО имеется группа программ, предназначенная для решения алгебраической проблемы собственных значений

$$\mathbf{A}\mathbf{x}=\lambda\mathbf{x} \qquad (7.11)$$

и обобщенных линейных проблем собственных значений

$$\mathbf{A}\mathbf{x}=\lambda\mathbf{B}\mathbf{x}; \tag{7.12}$$

$$\mathbf{A}\mathbf{B}\mathbf{x}=\lambda\mathbf{x}. \tag{7.13}$$

Для решения полной симметричной проблемы (7.11) реализован QL-алгоритм [29, 36], основанный на приведении матрицы A к трехдиагональной форме.

Приведение симметричной матрицы **A** к трехдиагональной форме выполняется стандартным алгоритмом PTF, использующим преобразование Хаусхолдера [36]:

```
& FPTF    SPFOR (&D , &E)
PTF       SASP  1 , &A , &FPTF
```

где &A — исходная симметричная матрица размером M×N, правильный порядок расположения элементов которой необходим только в ее нижнем треугольнике. После выполнения алгоритма на ее место записывается матрица преобразования, необходимая для вычисления собственных векторов матрицы **A**; &FPTF — имя списка фактических результатов; &D — результирующая матрица размером 1×N, в которую СА PTF помещает диагональ вычисленной трехдиагональной матрицы $\mathbf{A}_{n-1}$; &E — результирующая матрица размером 1×N, в первые $(n-1)$ элементов которой СА PTF помещает поддиагональные элементы вычисленной трехдиагональной матрицы $\mathbf{A}_{n-1}$. Элемент E (N) = 0.

Д: НЕКВАДРАТНАЯ МАТРИЦА!, если $M \neq N$.

Если в матрице **A** диапазон значений элементов весьма широк, желательно, чтобы наименьшие из них располагались в левом верхнем углу. Это позволит получить даже наименьшие по модулю значения с высокой относительной точностью, равной 2^{-103} [36]. В противном случае ошибка может достигать величины порядка $2^{-103}\|\mathbf{A}\|$.

Для определения собственных значений и, возможно, векторов матрицы **A**, после приведения ее к трехдиагональному виду используется QL-алгоритм с неявным сдвигом [36], который определяется следующими соотношениями:

$$\mathbf{Q}_s(\mathbf{A}_s-\sigma_s\mathbf{E})=\mathbf{L}_s;\ \mathbf{A}_{s+1}=\mathbf{L}_s\mathbf{Q}_s^{\tau}+\sigma_s\mathbf{E};$$

$$\mathbf{A}_{s+1}=\mathbf{Q}_s\mathbf{A}_s\mathbf{Q}_s^{\tau},$$

где $\mathbf{Q}_s$ — ортогональная матрица; $\mathbf{L}_s$ — нижняя треугольная матрица; σ_s — сдвиг, выбираемый для ускорения сходимости процесса.

В ППП СМПО QL-алгоритм используется в два этапа. На первом этапе определяются только собственные значения. При этом сдвиг σ_s выбирается по стратегии Уилкинсона [36], т. е. σ_s равен тому собственному значению матрицы

$$\begin{bmatrix} d_1^s & e_1^s \\ e_1^s & d_2^s \end{bmatrix},$$

которое находится ближе к d_1^s. Алгоритм взят из работы [36] и

имеет очень хорошую сходимость: для отыскания N собственных чисел требуется $\simeq 1{,}7$ N итераций. Если требуется определить собственные векторы, то на втором этапе заново выполняется QL-алгоритм, причем ранее вычисленные собственные значения используются в качестве сдвигов (так называемые финальные сдвиги) и реализованы быстрые масштабированные вращения [29]. Теоретически при использовании в качестве сдвига любого точного собственного значения за одну итерацию должно произойти расщепление матрицы. В силу ошибок округления этого не происходит, поэтому реализована следующая стратегия: первый сдвиг выбирается равным собственному значению, ближайшему к d_1^1. Последующие итерации проводятся со сдвигом по Уилкинсону до тех пор, пока e_1^s не станет пренебрежимо мало. Затем на $(s+1)$-й итерации в качестве сдвига выбирается собственное значение, ближайшее к d_2^s. На $(s+2)$-й итерации снова используется сдвиг по Уилкинсону и т. п. Такой подход сокращает число итераций при определении собственных векторов до $\approx 1{,}5$ N. Так как для вычисления всех собственных значений требуется $\approx 40N^2$ операций умножения, а одна итерация QL-алгоритма требует $\approx 4N^2$ операций, если определяются собственные векторы, то при $N>10$ такой подход дает ощутимый выигрыш. Увеличение времени при определении собственных векторов связано с необходимостью находить произведение матриц преобразования Q_1, что не нужно делать, если определяются только собственные значения. При накоплении матриц преобразования с помощью быстрых масштабированных вращений потребное число умножений уменьшается почти вдвое.

QL-алгоритм является весьма устойчивым к кратным собственным значениям. Более того, при наличии кратных корней время его выполнений уменьшается за счет расщепления трехдиагональных матриц на прямую сумму подматриц меньшего размера.

Для вычисления собственных значений симметричной трехдиагональной матрицы или задачи (7.11) используется алгоритм QLZ

```
&FQLZ     SPFOR (&Z)

QLZ       SASP  1 , &D , &E , &FQLZ
```

где &D — матрица размером $1\times M$, в которой записана диагональ трехдиагональной матрицы; &E — матрица размером $1\times N$, в первых (N—1) элементах которой записаны поддиагональные элементы трехдиагональной матрицы. Она не должна быть запомнена, иначе она будет разрушена алгоритмом QLZ; &FQLZ — имя списка, содержащего название фактического результата; &Z — результирующая матрица размером $1\times N$, в которую СА QLZ помещает вычисленные собственные значения, расположенные в порядке убывания.

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ!, если $M\neq N$.

ДЛЯ ВЫЧИСЛЕНИЯ СОБСТВЕННОГО ЗНАЧЕНИЯ ИЛИ ВЕКТОРА ПОТРЕБОВАЛОСЬ БОЛЕЕ 30 ИТЕРАЦИЙ! ЗАДАЧА

СНИМАЕТСЯ, если сходимость процесса патологически низкая. Такое бывает чрезвычайно редко. В этом случае можно рекомендовать перейти от задачи (7.11) к задаче

$$(\mathbf{A}+\mu\mathbf{E})\mathbf{x}=\bar{\lambda}\mathbf{x} \tag{7.14}$$

для некоторого значения множителя μ. Задача (7.14) имеет те же собственные векторы, что и (7.11), а собственные значения их связаны простым соотношением

$$\lambda=\bar{\lambda}-\mu. \tag{7.15}$$

Если векторы &D и &E получены алгоритмом PTF, то в результате выполнения алгоритма QLZ мы получим собственные значения задачи (7.11).

Для вычисления собственных векторов симметричной трехдиагональной матрицы или задачи (7.11) после алгоритма QLZ используется алгоритм QLV

```
QLV      SASP  1, &D, &E, &Z, &A
```

где &D — матрица размером 1×M, в которой записаны диагональные элементы трехдиагональной матрицы. Матрица &D может быть получена с помощью алгоритма PTF; &E — матрица размером 1×N, в первых (N—1) элементах которой записаны поддиагональные элементы трехдиагональной матрицы. Матрица &E может быть получена программой PTF; &Z — матрица размером 1×K, в которой записаны собственные значения решаемой задачи, полученные программой QLZ; &A — матрица размером L×S. При вычислении собственных векторов трехдиагональной матрицы &A должна быть единичной матрицей; при вычислении собственных векторов (7.11) матрица &A должна содержать матрицу преобразования, полученную программой PTF. В результате выполнения программы QLV на месте матрицы &A будут получены собственные векторы. Векторы располагаются по столбцам таким образом, что соответствующие им собственные значения убывают.

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $M\neq N$ или $M\neq K$ или $M\neq L$ или $M\neq S$.

ДЛЯ ВЫЧИСЛЕНИЯ СОБСТВЕННОГО ЗНАЧЕНИЯ ИЛИ ВЕКТОРА ПОТРЕБОВАЛОСЬ БОЛЕЕ 30 ИТЕРАЦИЙ! ЗАДАЧА СНИМАЕТСЯ — причина та же, что и при выполнении программы QLZ.

Поскольку при выполнении программы QLV собственные значения перевычисляются заново, возможно появление сообщения СОБСТВЕННЫЕ ЗНАЧЕНИЯ БЫЛИ ВЫЧИСЛЕНЫ С НИЗКОЙ ТОЧНОСТЬЮ! Оно означает, что вновь вычисленное собственное значение сильно отличается от вычисленного программой QLZ, т. е. имеет высокую относительную погрешность. Сообщение обычно относится к наименьшим по модулю собственным значениям и появляется для каждого неточно вычисленного значения. Такие соотношения свидетельствуют о плохой обусловленности матрицы.

Собственные векторы, определенные СА QLV, всегда ортогональны с рабочей точностью, в том числе и при кратных собственных значениях.

Рассмотрим пример использования описанных алгоритмов.

Пример 7.5.1

Требуется найти и напечатать собственные значения и векторы симметричной матрицы A

```
FP      SPFOR (D, E)
PTF     SASP  1, A, FP
FQL     SPFOR (Z)
QLZ     SASP  1, D, E, FQL
        PE    Z
QLV     SASP  1, D, E, Z, A
        PE    A
```

QL-алгоритм — наиболее эффективный алгоритм [29] при решении полной собственной проблемы для симметричных матриц, а также при необходимости отыскания больше N/4 собственных значений и векторов. На его основе в СМПО разработаны и применяются алгоритмы для решения полной собственной проблемы вида (7.12) и (7.13), в которых **A** и **B** — симметричные матрицы.

Рассмотрим задачу (7.12). Матрицы **A** и **B** могут не быть положительно определенными. Важно лишь, чтобы этим свойством обладала матрица $\mathbf{A}+\mu\mathbf{B}$ для некоторого μ. Тогда задача (7.12) примет вид

$$(\mathbf{A}+\mu\mathbf{B})\mathbf{x}=(\lambda+\mu)\mathbf{B}\mathbf{x}.$$

Обозначая $\mathbf{C}=\mathbf{A}+\mu\mathbf{B}$; $\bar{\lambda}=\frac{1}{\lambda+\mu}$, получим задачу вида

$$\mathbf{B}\mathbf{x}=\bar{\lambda}\mathbf{C}\mathbf{x}, \tag{7.16}$$

где **C** — положительно-определенная матрица. Указанное преобразование выполнять не нужно, если **A** или **B** положительно определена.

Разлагая **C** по Холецкому: $\mathbf{C}=\mathbf{L}\mathbf{L}^{\mathrm{T}}$, из (7.16) получим

$$(\mathbf{L}^{-1}\mathbf{B}\mathbf{L}^{-\mathrm{T}})\mathbf{L}^{\mathrm{T}}\mathbf{x}=\bar{\lambda}(\mathbf{L}^{\mathrm{T}}\mathbf{x}), \text{ или}$$

$$\mathbf{D}\mathbf{y}=\bar{\lambda}\mathbf{y}, \tag{7.17}$$

где

$$\mathbf{D}=\mathbf{L}^{-1}\mathbf{B}\mathbf{L}^{-\mathrm{T}};\ \mathbf{y}=\mathbf{L}^{\mathrm{T}}\mathbf{x}. \tag{7.18}$$

Здесь **D** — симметричная матрица, поэтому для решения задачи (7.17) может быть использован QL-алгоритм.

Частным случаем задачи (7.12) является задача о собственных колебаниях конструкций, возникающая, например, в методе перемещений:

$$\mathbf{K}\mathbf{x}=\omega^2\mathbf{M}\mathbf{x}. \tag{7.19}$$

Здесь **K** и **M** неотрицательно определены. Если конструкция закреплена, то **K** — положительно определена, в противном случае положительно определена матрица $\mathbf{K}+\mu\mathbf{M}$ для всех $\mu>0$. Таким образом, задача (7.19) легко приводится к виду (7.16).

Для решения задачи (7.16) с помощью описанной методики используется СА QLAB

```
&FQLAB   SPFOR (&Z,&VEK)
QLAB     SASP  1,&C, &B,&FQLAB
```

где &C — положительно-определенная блочная матрица из (7.16), в качестве которой может, в частности, выступать матрица **K** или $\mathbf{K}+\mu\mathbf{M}$ из (7.19). Правильное расположение элементов &C требуется только в верхнем ее треугольнике; &B — симметричная матрица из (7.16), в качестве которой может использоваться матрица **M** из (7.19). Правильное расположение элементов &C требуется только в ее верхнем треугольнике; &FQLAB — имя списка фактических результатов; &Z — результирующий вектор, содержащий вычисленные собственные значения $\bar{\lambda}$ для задачи (7.17), упорядоченные по убыванию. Для определения ω из (7.19) следует помнить, что $\bar{\lambda}=1/\omega^2$, если сдвиг не применялся, и $\bar{\lambda}=1/(\omega^2+\mu)$, если сдвиг равен μ; &VEK — результирующая блочная матрица, содержащая вычисленные собственные векторы. Блочная структура матрицы &VEK та же, что и у матрицы &C. Собственные векторы расположены по столбцам в том же порядке, что и отвечающие им собственные значения, и ортонормированы относительно матрицы &C.

Рассмотрим теперь проблему (7.13). Считая, что матрица **B** положительно определена, разлагаем ее по схеме Холецкого $\mathbf{B}=\mathbf{L}\mathbf{L}^{\tau}$. Тогда задача (7.13) приводится к виду

$$(\mathbf{L}^{\tau}\mathbf{A}\mathbf{L})(\mathbf{L}^{\tau}\mathbf{x})=\lambda(\mathbf{L}^{\tau}\mathbf{x}), \text{ или}$$

$$\mathbf{C}\mathbf{y}=\lambda\mathbf{y}, \tag{7.20}$$

где

$$\mathbf{C}=\mathbf{L}^{\tau}\mathbf{A}\mathbf{L},\ \mathbf{y}=\mathbf{L}^{\tau}\mathbf{x}. \tag{7.21}$$

Здесь **C** — симметричная матрица, поэтому для решения задачи (7.20) можно применить QL-алгоритм. Проблему (7.13) можно, вообще говоря, решать и непосредственно, найдя произведение $\mathbf{D}=\mathbf{A}\mathbf{B}$ и получив стандартную несимметричную проблему вида

$$\mathbf{D}\mathbf{x}=\lambda\mathbf{x}, \tag{7.22}$$

однако преобразование к виду (7.20) более удобно, ибо оно сохраняет симметрию.

Для решения задачи (7.13) преобразованием к виду (7.20) используется СА QLS, работающий для диагональной матрицы **B**:

```
& FQLS   SPFOR (&Z,&VEK)
QLS      SASP  1,&A,&B,&FQLS,
```

где &A — блочная симметричная матрица, правильное расположение элементов которой необходимо только в нижнем ее треуголь-

нике; &B — простая вектор-строка, представляющая собой диагональную положительно-определенную матрицу; &FQLS — имя списка фактических результатов; &Z — результирующий простой вектор, содержащий вычисленные собственные значения, расположенные в порядке убывания; &VEK — результирующая блочная матрица, содержащая вычисленные собственные векторы. Блочная структура матрицы &VEK та же, что и у матрицы &A. Блочные векторы расположены по столбцам в том же порядке, что и отвечающие им собственные значения, и ортонормированы относительно матрицы &B.

Решение полной несимметричной проблемы собственных значений вида (7.22) для действительных матриц проводится в ППП СМПО методом Якоби с понижением нормы [36].

Алгоритм решения собственной проблемы основан на итерационном процессе выполнения преобразований подобия типа Якоби: $\mathbf{T}^{-1}\mathbf{A}\mathbf{T}$, где $\mathbf{T}=\mathbf{T}_1\cdot\ldots\cdot\mathbf{T}_i\ldots$ Каждая матрица $\mathbf{T}_i$ представляется в виде произведения двух матриц $\mathbf{p}_i\times\mathbf{s}_i$, где $\mathbf{p}_i$ определяет вращение, а $\mathbf{s}_i$ — сдвиг или комплексное вращение.

Для решения задачи (7.22) методом Якоби используется СА JOCB

```
&FJOCB   SPFOR (&VEK)

JOCB     SASP  1,&D, &KI, &EP, &FJOCB
```

где &D — имя простой квадратной матрицы, для которой решается собственная проблема; &KI — имя поля, содержащее число в формате полуслова, равное максимальному количеству итераций; &EP — имя поля, содержащего число с плавающей точкой двойной точности, задающее точность, которая должна быть достигнута; &FJOCB — имя списка, содержащего название фактического результата; &VEK — результирующая квадратная матрица, содержащая в своих строках правые собственные векторы. Собственные значения будут располагаться в блоках на диагонали матрицы &D по мере уменьшения их абсолютной величины. Блоки размером 1×1 содержат действительные собственные значения, а блоки размером 2×2 соответствуют комплексным собственным значениям $a_{jj}\pm ia_{j,j+1}$:

$$\begin{bmatrix} a_{jj} & a_{j,j+1} \\ -a_{j,j+1} & a_{jj} \end{bmatrix}.$$

Комплексные векторы $t_j\pm it_{j+1}$ будут располагаться в строках j, $j+1$ матрицы &VEK.

Д: ТОЧНОСТЬ НЕ МОЖЕТ БЫТЬ ДОСТИГНУТА, если задана слишком высокая точность, а достигнутая точность не возрастает из-за ошибок округления. Выполнение алгоритма завершается, но задание продолжает выполняться. Достигнутую величину точности можно оценить по величине внедиагональных элементов &D.

Рассмотрим теперь программы решения частичной собственной проблемы вида (7.12) и (7.13).

Для отыскания нескольких наибольших собственных значений и соответствующих им собственных векторов задачи (7.13) используется метод итераций подпространства (одновременных итераций) следующего вида [29]. Пусть матрицы **A** и **B** имеют размеры $N \times N$.

1. Считая матрицу **B** диагональной, приводим задачу (7.13) к виду (7.20).
2. Выбираем $\mathbf{S}_0$ — начальное приближение размером $N \times M$.
3. Вычислим $\mathbf{Y}_\nu = \mathbf{C}\mathbf{S}_\nu$, где ν — номер итерации, равный 0, 1, ...
4. Вычислим $\mathbf{H}_\nu = \mathbf{Y}_\nu^T \mathbf{Y}_\nu$. Размеры $\mathbf{H}_\nu = M \times M$.
5. Решим полную симметричную собственную проблему для матрицы $\mathbf{H}_\nu$: $\mathbf{V}_\nu \Delta_\nu^2 \mathbf{V}_\nu^T = \mathbf{H}_\nu$.
6. Вычислим новое приближение $\mathbf{S}_\nu = \mathbf{Y}_\nu \mathbf{V}_\nu \Delta_\nu^{-1}$ и перейдем к п. 3.

Итерационный процесс продолжается до достижения заданной точности. Найденные Δ_ν будут представлять собой собственные значения задачи (7.13). Собственные векторы восстанавливаются из (7.21), где в качестве **Y** берется $\mathbf{S}_\nu$.

Описанный алгоритм реализован в CA RED1, обращение к которому имеет вид

```
                UTIMER H=&H, M=&M, C=&C

&FRED           SPFOR  (&R1, &R2, &L)

RED1            SASP   1, &A, &B, &S, &KR, &EQ, &FRED1
```

где &A — блочная симметричная матрица, блочный размер которой $K \times K$, размер блока $\&A_{ij}$ равен $l_i \times l_j$, $i, j = 1, \ldots, K$. Правильное расположение элементов &A необходимо только в верхнем ее треугольнике; &B — блочный вектор-строка, представляющий диагональную матрицу. Блочный размер $1 \times K$, размер блока $\&B_i$ равен $1 \times l_i$, $i = 1, \ldots, K$; &S — имя поля, содержащего число в формате полуслова, равное количеству отыскиваемых собственных значений и векторов; &KR — имя поля, содержащего число в формате полуслова, равное максимальному количеству итераций; &EQ — имя поля, содержащего число с плавающей точкой двойной точности, равное точности, которая должна быть достигнута при вычислении собственных значений; &FRED1 — имя списка фактических результатов.

В процессе работы алгоритма итерируется $m = 2\&S$ векторов; на каждом этапе печатаются вычисленные собственные значения (Δ). В конце работы печатается число сошедшихся собственных пар; собственные значения, собственные частоты $\omega = 1/\sqrt{\lambda}$; собственные векторы.

В результате работы CA RED1 формируются: &R1 — вектор размером $1 \times m$, содержащий вычисленные собственные значения; &R2 — блочная матрица блочным размером $1 \times K$, размер i-го блока $m \times l_i$, $i = 1, \ldots, K$, содержащая расположенные по строкам ортонормированные относительно матрицы &B собственные векторы; &L — блочная матрица блочным размером $m \times 1$, размер блока $1 \times$

$\times N$, где $N=\sum_{i=1}^{n} l_i$ — содержащая расположенные по строкам нормированные по максимальному элементу собственные векторы.

Для предотвращения опасности бесконтрольного счета указывается оператор UTIMER, задающий максимально возможное время обработки СА RED1 центральным процессором: &H — часы; &M — минуты; &C — секунды. Любой из параметров &H, &M, &C может быть опущен и, следовательно, равен нулю.

Таким образом, алгоритм RED1 может завершить свою работу по одной из трех причин: достигнута заданная точность, выполнено заданное число итераций, исчерпано заданное время счета. Причина завершения работы будет отражена в распечатке.

В качестве первого приближения выбираются столбцы матрицы **C**, имеющие наибольшие диагональные элементы.

Для отыскания нескольких наименьших собственных значений и соответствующих им векторов задачи (7.12) используется метод одновременных итераций в форме, приведенной в работе [5]. Считаем, что матрица **A** положительно определена, в противном случае применяем преобразование к виду (7.16). Пусть матрицы **A** и **B** имеют размеры $N\times N$.

1. Разлагаем матрицу **A** по схеме Холецкого.
2. Выбираем начальное приближение $\mathbf{S}_0$ размером $N\times M$.
3. Находим $\mathbf{Y}_\nu=\mathbf{B}\mathbf{S}_\nu$, где ν — номер итерации, равный 0, 1, ...
4. Решаем систему уравнений методом Холецкого $\mathbf{A}\mathbf{C}_\nu=\mathbf{Y}_\nu$.
5. Находим проекцию **A** на **C**: $\mathbf{A}_\nu=\mathbf{C}_\nu^{\mathrm{T}}\mathbf{A}\mathbf{C}_\nu=\mathbf{C}_\nu^{\mathrm{T}}\mathbf{Y}_\nu$. Размеры $\mathbf{A}_\nu$ — $M\times M$.
6. Находим проекцию **B** на **C**: $\mathbf{B}_\nu=\mathbf{C}_\nu^{\mathrm{T}}\mathbf{B}\mathbf{C}_\nu$. Размеры $\mathbf{B}_\nu$ — $M\times M$.
7. Решаем полную обобщенную симметричную собственную проблему для редуцированных матриц: $\mathbf{B}_\nu\mathbf{V}_\nu=\mathbf{A}_\nu\mathbf{V}_\nu\Lambda_\nu$.
8. Вычислим новое приближение $\mathbf{S}_{\nu+1}=\mathbf{C}_\nu\mathbf{V}_\nu$ и перейдем к п. 3.

Итерационный процесс продолжается до достижения заданной точности. Величины $(\Lambda_\nu)^{-1}$ и $\mathbf{S}_{\nu+1}$ представляют собой искомые собственные пары задачи (7.12).

Описанный алгоритм реализован в СА PRED, обращение к которому имеет вид

```
                UTIMER H = &H, M = &M, C = &C

&FPRED          SPFOR  (&VEK, &Z)

PRED            SASP   1, &A, &B, &SP, &N, &EQ, &FPRED
```

где &A — множитель Холецкого, полученный программой BRXL для блочной матрицы **A** блочным размером $K\times K$ с размером блока $l_i\times l_j$, $i, j=1, ..., n$; &B — блочная матрица **B**. Может задаваться в виде квадратной матрицы той же структуры, что и &A или в виде блочного вектор-столбца блочным размером $K\times 1$ с размерами блоков $l_i\times 1$, $i=1, 2, ..., n$; &SP — простой статический или динамиче-

ский список, состоящий из P компонент, указывающих абсолютные номера столбцов, выбираемых для начального приближения. Обычно выбирают номера, соответствующие наибольшим диагональным элементам матрицы &B. Для отыскания q собственных пар нужно задавать список &SP размером $p=\min\{2q, q+8)$; &N — имя поля, содержащего число в формате полуслова, задающее максимальное число итераций; &EQ — имя поля, содержащего число с плавающей точкой двойной точности, равное точности, которая должна быть достигнута при вычислении q собственных значений; &FPRED — имя списка фактических результатов; &VEK — результирующая матрица блочным размером $K\times 1$ с размерами блоков $l_i\times p$, содержащая вычисленные собственные векторы, ортонормированные относительно матрицы A и расположенные по столбцам; &Z — результирующий вектор размером $1\times p$, содержащий вычисленные значения λ^{-1} и приближения к ним, упорядоченные по убыванию.

Для предотвращения опасности бесконтрольного счета должен указываться оператор UTIMER, задающий максимально возможное время обработки CA PRED центральным процессором: &H — часы, &M — минуты, &C — секунды. Любой из этих параметров может быть опущен и, следовательно, равен нулю.

Таким образом, выполнение CA PRED прекращается либо при достижении заданной точности, либо при выполнении заданного числа итераций, либо при исчерпывании заданного интервала времени. Рассмотрим пример применения CA PRED.

Пример 7.5.2

Требуется найти 20 наименьших собственных частот и векторов для закрепленной конструкции с матрицей жесткости K и матрицей масс M. Для решения задаем время 40 мин, 25 итераций и точность $1\cdot 10^{-5}$.

```
          VVS     SR
BRXL      SASP    1,K
          UTIMER  M = 40
&FPR      SPFOR   (VEK, LAM)
PRED      SASP    1, K, M, SR, N, T, FPRED
          VP      KD, (LAM, ED), LAM
          VP      IKP, (LAM), LAM
OMEGA     EQU     LAM
          PE      OMEGA
BPE       SASP    1, VEK, (NET, 3)
...       ...     . . . . . . . . .
ED        DC      D'1'
T         DC      D'1.E — 5'
N         DC      H'25'
...       ...     . . . . . . . . .
```

```
...        ...        . . . . . . . . .
1:28       1,28       исходные данные.
```

В примере с помощью оператора VVS вводится простой динамический список, содержащий 28 направлений. Следовательно, CA PRED будет находить 20 собственных пар. Полученные в матрице LAM значения λ^{-1} преобразуются в круговые собственные частоты по формуле $\omega=(\mathrm{LAM})^{-1/2}$. Полученные значения ω и собственные формы печатаются.

Для решения частичной собственной проблемы (7.19) при диагональной матрице M используется алгоритм, основанный на методе Ланцоша [29, 52]. Алгоритм модифицирован и предназначен для отыскания любого участка спектра. Матрицы **K** и **M** должны быть положительно полуопределены.

Итак, пусть требуется найти собственные пары из интервала $\omega^2\in[a, b]$. Используя некоторое значение сдвига $\sigma\in[a, b]$, например $\sigma=\frac{a+b}{2}$, преобразуем задачу (7.19) к виду

$$[\mathbf{K}-\sigma\mathbf{M}]\,\mathbf{x}=(\omega^2-\sigma)\,\mathbf{M}\mathbf{x}. \tag{7.23}$$

Собственные пары задачи (7.23) требуется отыскивать в интервале $[a-\sigma, b-\sigma]$, для чего можно применять алгоритм из работы [5]. Важно, чтобы матрица $\mathbf{K}-\sigma\mathbf{M}$ была невырождена и допускала разложение Холецкого вида $\mathbf{K}-\sigma\mathbf{M}=\mathbf{LDL}^{\mathrm{T}}$ (см. разд. 7.4). Напомним, что для этого не требуется положительная определенность матрицы $\mathbf{K}-\sigma\mathbf{M}$. Если матрица **K** соответствует закрепленной конструкции и требуется найти несколько наименьших частот, сдвиг можно принять равным нулю. Таким образом, алгоритм метода Ланцоша будет иметь следующий вид.

1. Для некоторого начального вектора **b** решаем систему уравнений $(\mathbf{K}-\sigma\mathbf{M})\mathbf{r}_0=\mathbf{b}$.

2. Вычислим $\beta_j=\sqrt{\mathbf{r}_{j-1}^{\mathrm{T}}\mathbf{M}\mathbf{r}_{j-1}}$.

3. Вычислим $\mathbf{q}_j=\mathbf{r}_{j-1}/\beta_j$.

4. Решим систему уравнений $(\mathbf{K}-\sigma\mathbf{M})\mathbf{r}_j=\mathbf{M}\mathbf{q}_j$.

5. Вычислим $\mathbf{r}_j=\mathbf{r}_j-\mathbf{q}_{j-1}\beta_j$.

6. Вычислим $\alpha_j=\mathbf{r}_j^{\mathrm{T}}\mathbf{M}\mathbf{q}_j$.

7. Для $j>1$ решаем полную собственную проблему $\mathbf{T}_j\mathbf{x}_j=\lambda_j\mathbf{x}_j$, где $\mathbf{T}_j$ — трехдиагональная матрица, образованная диагональными элементами α_i, $i=1,\ldots,j$ и поддиагональными элементами β_i, $i=2, 3,\ldots,j$.

8. Вычислим $\mathbf{r}_j=\mathbf{r}_j-\alpha_j\mathbf{q}_j$ и перейдем к п. 2.

Итерационный процесс продолжается до тех пор, пока вычисленные на этапе 7 сошедшиеся собственные значения (для сравнения используются собственные значения, найденные на предыдущей итерации) не попадут в заданный интервал либо их количество не превысит заданное.

Описанный алгоритм, как хорошо известно [29], является неустойчивым, ибо постепенно векторы Ланцоша $\mathbf{q}_j$ перестают быть **M**-ортогональными. Чтобы этого избежать, в алгоритм добавляется

критерий Саймона [52]. Суть его в следующем. Вводится вектор $\mathbf{W}_j^{\tau}=\mathbf{q}_{j+1}^{\tau}\mathbf{M}\mathbf{Q}_j$, где матрица $\mathbf{Q}_j$ собрана из вектор-столбцов $\mathbf{q}_1, \ldots, \mathbf{q}_j$. Вектор $\mathbf{W}_j$ может быть вычислен по простой рекуррентной формуле

$$\beta_{j+1}\mathbf{W}_j^{\tau}=[\mathbf{W}_{j-1}^{\tau} \vdots 1][\mathbf{T}_j-\alpha_j\mathbf{E}]-\beta_j[\mathbf{W}_{j-2}^{\tau} \vdots 1 \vdots 0], \qquad (7.24)$$

где $\mathbf{E}$ — единичная матрица, причем $\mathbf{W}_0=0$ и $\mathbf{W}_1=\mathbf{q}_2\mathbf{M}\mathbf{q}_1$. Элементы этого вектора показывают потерю ортогональности. Если один из них больше, чем $\sqrt{\varepsilon}$, где ε — машинная точность, то следует ортогонализировать $\mathbf{q}_{j+1}$ ко всем предыдущим $\mathbf{q}_i$, положить $\mathbf{W}_j=\varepsilon$ и продолжить описанный выше алгоритм.

Для решения системы уравнений на этапах 1 и 4 обычно применяются методы Холецкого. Поэтому перед началом процесса Ланцоша матрицу $\mathbf{K}-\sigma\mathbf{M}$ разлагают алгоритмом BRXL, если она положительно определена, или алгоритмом BRXN в противном случае.

Для решения задачи (7.19) с помощью описанной методики используется комбинация алгоритмов RDML и MLSV. Алгоритм RDML строит процесс Ланцоша

```
&FRDML   SPFOR (&QN, &ALFN, &BETN)
RDML     SASP  1, &K, &B, &M, & NI, &KZ, NET, & AO, &BO, & SDV,
                &OP, &FRDML, & QS, &ALFS, &BETS
```

где &K — матрица, полученная алгоритмом BRXN или BRXL при разложении матрицы $\mathbf{K}-\sigma\mathbf{M}$, либо другая матрица, которая требуется специальной программе, решающей систему уравнений $(\mathbf{K}-\sigma\mathbf{M})\mathbf{x}=\mathbf{a}$ относительно $\mathbf{x}$; &B — простая вектор-строка размером $1\times N$, содержащая значение начального вектора $\mathbf{b}$; &M — простая вектор-строка размером $1\times N$, содержащая диагональ матрицы масс; &NI — имя поля, содержащего число в формате полуслова, равное максимально допустимому количеству итераций в процессе Ланцоша. Если программа RDML вызывается повторно, число будет означать суммарное количество итераций; &KZ — имя поля, содержащего число в формате полуслова, равное количеству отыскиваемых значений; (&AO, &BO) — поля в формате двойного слова, определяющие интервал, в котором нужно определить собственные частоты $f=\omega/2\pi$; &SDV — поле в формате двойного слова, содержащее величину сдвига $(-\sigma)$; &OP — имя поля, содержащего название стандартного алгоритма, решающего систему уравнений $(\mathbf{K}-\sigma\mathbf{M})\mathbf{x}=\mathbf{a}$; &FRDML — имя списка фактических результатов работы СА RDML; &QN — результирующая блочная матрица блочным размером $L\times 1$, где L — число итераций СА RDML; размер блока $1\times N$. i-й блок матрицы &QN содержит вектор Ланцоша $\mathbf{q}_i$; &ALFN — результирующая вектор-строка размером $1\times L$, i-й элемент которой содержит число α_i; &BETN — результирующая вектор-строка размером $1\times L$, i-й элемент которой содержит число β_i; &QS, &ALFS, &BETS — результаты предыдущего выполнения программы RDML. Если она выполняется первый раз, они должны быть заданы как NET.

Поле &OP должно задаваться в неисполняемой части программы в виде

```
&OP      DC    CL4'&NAME',
```

где &NAME — имя СА, решающего систему уравнений. СА должен иметь два параметра. Первый из них — матрица &K, второй — правая часть системы, заданная в виде простой вектор-строки. Решение должно получаться на месте правой части. Если &K получена с помощью алгоритма BRXN, в качестве &NAME используется SRSN; если &K получена с помощью алгоритма BRXL, то &NAME должно иметь значение SRSS.

Решение малой редуцированной трехдиагональной собственной проблемы выполняется с помощью QL-алгоритма (СА QLZ).

Итерационный процесс может быть прекращен по одной из следующих причин: сошлось заданное число собственных частот, найденные собственные частоты покрыли заданный интервал, выполнено указанное число итераций. При необходимости итерационный процесс может быть продолжен. В этом случае матрицы &K, &M, &B и параметр &SDV не должны изменяться; параметр &NI должен быть увеличен, если процесс был остановлен ввиду исчерпания заданного числа итераций.

В процессе решения на печать выводятся текущие приближения к собственным частотам. В конце работы печатается причина завершения работы, количество сошедшихся частот и номер первой сошедшейся частоты.

От успешного выбора начального вектора зависит скорость сходимости алгоритма Ланцоша. Если **b** **M**-ортогонален некоторому собственному вектору, то соответствующая собственная пара может оказаться пропущенной. В точной арифметике это всегда так; на практике вследствие ошибок округления эта собственная пара может быть найдена, однако она появится в спектре позднее соседних с ней частот. Если известны приближения к собственным векторам, в качестве **b** можно использовать их сумму. При отсутствии другой информации можно использовать вектор **b**, состоящий из единиц.

Если есть подозрения, что частоты пропущены, их можно проверить методом деления спектра [29]. Для этого нужно разложить с помощью СА BRXN матрицы $\mathbf{K}-a\mathbf{M}$ и $\mathbf{K}-b\mathbf{M}$ и найти число положительных элементов на их диагоналях. Их разность даст количество собственных значений на интервале $[a, b]$.

Для отыскания собственных векторов после программы используется СА MLSV

```
&FMLSV   SPFOR (&ZZ,&FF)
MLSV     SASP  1,&QN,&ALFN, &BETN,&SDV,&SP, &FMLSV
```

где &QN, &ALFN, &BETN — результаты работы СА RDML; &SDV — та же, что и в СА RDML; &SP — простой динамический или смешанный статический список, содержащий номера, для которых будут найдены собственные пары. Номера должны соответст-

вовать номерам в матрице собственных частот, напечатанных на последнем этапе работы CA RDML. Если &SP задан как NET, будут найдены собственные векторы, соответствующие всем частотам редуцированной собственной проблемы, полученной на последней итерации CA RDML, в том числе и тем частотам, которые еще не сошлись к собственным частотам исходной задачи; &FMLSV — список фактических результатов CA MLSV; &ZZ — результирующая матрица собственных частот размером $1 \times p$. Здесь p равно числу элементов списка &SP. Если &SP задан как NET, то $p = L$. Собственные частоты перечисляются в порядке возрастания; &FF — результирующая блочная матрица размером $p \times 1$, размер блока $1 \times N$. Каждый i-й блок содержит собственный вектор, соответствующий i-й собственной частоте. Векторы нормированы относительно наибольшего элемента и ортогональны относительно матрицы **M** с высокой точностью даже при наличии кратных собственных значений.

В процессе решения печатается точность определения собственных векторов, равная L-й строке матрицы собственных векторов редуцированной задачи [29]. Элементы собственного вектора исходной задачи имеют погрешность того же порядка, что и соответствующий элемент этой строки.

Комбинация алгоритмов RDML и MLSV работает быстрее, чем CA PRED. Так, например, при решении задачи собственных колебаний подкрепленной оболочки с 5400 степенями свободы для определения 22 собственных чисел и векторов CA PRED затратил 98 мин на ЭВМ ЕС-1045, а CA RDML и MLSV определили 24 собственные пары за 65 мин. Ускорение особенно заметно при необходимости отыскивать большое число собственных пар. В частности, для задачи размерностью 10000 за 4 ч CA RDML отыскал 60 собственных пар, а CA PRED — 20, после чего сходимость последнего резко замедлилась (итерировались 32 вектора).

7.6. ПЕРЕСТРОЕНИЕ МАТРИЦ

Выборка (запись) отдельного элемента a_{ij} из матрицы A (в матрицу A) размером $M \times N$ осуществляется программой VEL (ZEL)

```
        V     VEL,(&A,&I),&C
        VPL   ZEL,(&C,&I),&A
&C      DC    D
&I      DC    H'i, j'
```

где &I — имя поля, содержащего два числа в формате полуслова, задающего координаты выбираемого (записываемого) элемента; &C — имя поля, в которое будет помещен выбранный элемент (откуда будет взят записываемый элемент).

Групповая выборка дескрипторов блоков из метаматрицы A осуществляется программой GVBL

```
        V     GVBL,(&A), &I
&I      DC    S(K, i1, j1, A1,..., il, jl, AL,..., ik, jk, AK)
```

где &I — имя поля, задающего количество K и координаты выбираемых дескрипторов; адреса полей AL, в которые эти дескрипторы должны быть помещены (L=1, ..., K): (A1, ..., AL, ..., AK).

В результате выборки дескриптор блока из метаматрицы переписывается в отведенную ему область. После этого выбранный блок становится доступным для обработки.

Разборка матрицы — выборка прямоугольной подматрицы C размером K×L из матрицы A, начиная с координат (i, j), осуществляется с помощью программы RAZ

```
VP    RAZ,(&A,&AI),&C
```

Разборка матрицы на отдельные ленты — выборка матрицы C размером 1×L или K×1 из матрицы A, начиная с координат (i, j), осуществляется с помощью программы RLEN

```
VP    RLEN,(&A, &I), &C
```

Элементы матрицы C определяются следующим образом: $c_e = a_{i+r,\, j+r}$, где $r=(l-1)(N+1)$, $l=1, ..., L(K)$.

Сборка матрицы — вписывание матрицы A в качестве прямоугольной подматрицы в матрицу C, начиная с указанных координат (i_1, j_1), ..., (i_l, j_l), ..., (i_n, j_n) осуществляется с помощью программы SBR

```
VPL   SBR,(&A,&I1,...,&IL,...,&IN),&C
```

Наложение матрицы — вписывание подматрицы A в матрицу C со сложением элементов

$$c_{i_l+i,\, j_l+j} = c_{i_l+i,\, j_l+j} + a_{i+1,\, j+1}$$

осуществляется с помощью программы NAL

```
VPL   NAL,(&A, &I1,..., &IL,..., &IN), &C
```

Разборка транспонированной матрицы C из матрицы A, начиная с координат (i, j), осуществляется программой RAZT

```
VP    RAZT,(&A,&I), &C
```

Элементы &C определяются следующим образом:

$$c_{kl} = a_{l+l-1,\, j+k-1}.$$

Сборка транспонированной матрицы A в матрицу C, начиная с координат (i, j), осуществляется программой SBRT

```
VPL   SBRT,(&A,&I),&C
```

Изменяющиеся элементы &C определяются по формуле $c_{kl} = a_{l-j+1,\, k-i+1}$.

Для всех рассмотренных операций возможны следующие диагностические сообщения:

Д: НЕВЕРНЫЕ КООРДИНАТЫ, если начальные координаты выборки элемента из матрицы А выходят за границы размеров матрицы: $i \notin [1, M]$, $j \notin [1, N]$.

НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если размеры выбираемых подматриц меньше соответствующих размеров результирующих матриц: $(M-K)<0$, $(N-L)<0$ для SBR, $K(L)<N-j+1$ или $K(L)<M-i+1$ для RLEN.

Разборка блочной матрицы — выборка простой подматрицы С размером $K \times L$ из блочной матрицы, представленной метаматрицей А размером $M \times N$, начиная с абсолютных координат (i, j), осуществляется с помощью СА BRAZ

```
BRAZ     SASP  1, &A, &I, &C
```

Д: НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ, если дескриптор блока, из которого должны быть выбраны элементы, содержит нулевые размеры.

Сборка блочной матрицы с обычной — вписывание обычной матрицы А в качестве прямоугольной подматрицы в блочную матрицу, представленную метаматрицей В, начиная с абсолютных координат, осуществляется программой BSBR

```
BSBR     SASP  1, & A, &I, & B
```

Наложение обычной матрицы (сборка со сложением) в блочную матрицу В выполняется аналогично сборке

```
BNAL     SASP  1, &A, &I, & B
```

Сборка метаматриц — вписывание метаматриц (суперблоков) A1, ..., AL, ..., AN в метаматрицу С, начиная с указанных координат $(i_1, j_1), \ldots, (i_l, j_l), \ldots, (i_N, j_N)$, осуществляется

```
         VPL   SBRM, (&A1, ..., &AL, ..., &AN, &I), &B
&I       DC    H'i1, j1, ..., il, jl, ..., iN, j'N
```

В процессе сборки происходит разнос размеров блоков по суперстрокам и суперстолбцам, пересекающим вписываемый суперблок. Размеры берутся из дескрипторов суперблока и заносятся только в те дескрипторы матрицы В, которые содержат нулевые размеры. При этом возможна диагностика:

Д: НЕРЕГУЛЯРНАЯ СТРУКТУРА, если в одной суперстроке содержатся блоки, имеющие разное количество строк, или в одном суперстолбце содержатся блоки с разным количеством столбцов.

Необходимо сделать следующее замечание: если дескриптор метаматрицы AL является нулевым (матрица не существует), то независимо от значения координат (i_l, j_l) вписывание этой матрицы не производится.

Для того чтобы разнести размеры и проверить блочную матрицу В на регулярность, можно воспользоваться упрощенной записью:

```
         VPL   SBRM,, & B
```

Пересчет абсолютных координат в блочной матрице в относительные выполняется оператором

```
        V     PKR, (&A, &B), &C
&B      DC    H'i_a, j'_a
&C      DC    4H
```

где &A — имя метаматрицы размером M×N; &B — имя поля с абсолютными координатами; &C — имя результирующего поля, в которое будут занесены ($i_б$, $j_б$) — координаты блока и ($i_о$, $j_о$) — координаты внутри блока.

Д: НЕВЕРНЫЕ КООРДИНАТЫ, если не выполняется хотя бы одно из условий: $\left(0 < i_a < \sum_{i=1}^{n} m_i\right)$, $\left(0 < j_a < \sum_{j=1}^{n} n_j\right)$, где ($m_i$, n_j) — размеры блоков в i-й суперстроке, j-м суперстолбце.

Для переписи структуры метаматрицы A в метаматрицу B, т. е. для формирования в матрице B такого же фона, как в матрице A, используется программа PRST:

```
        VP    PRST, (&A), &B
```

Рассмотрим группу операций по перестроению матрицы A размером M×N в матрицу C размером K×L по информации, заданной в виде сложного списка &S:

```
&S      ADSP  (Q, &S)
. . . . . . . . . . . .
```

где каждый из подсписков S_i (i=1, ..., Q) задает номера строк матрицы A или матрицы C:

```
&SI     CSP   t^i_1, ..., t^i_j, ..., t^i_q_i
```

Номер конкретного подсписка S_i, по которому перестраивается матрица, указывается в виде

```
&I      DC    H'i'
```

Операцию перестроения матрицы A по списку S в матрицу C можно выполнить с помощью программы PSTS

```
        VP    PSTS, (&A, &S, &I, &P), &C
&P      DC    C'&R'
```

Возможны два режима выполнения этой операции. Если размеры матриц таковы, что M>K, то из матрицы A переносятся строки с номерами $t^i_1, \ldots, t^i_j, \ldots, t^i_{q_i}$ в строки 1, ..., j, ..., q_i матрицы C. Если M≤K, то строки 1, ..., j, ..., q_i из матрицы A переносятся в строки $t_1{}^i$, ..., t^i_j, ..., $t^i_{q_i}$ матрицы C. Остальные строки матрицы C не изменяются.

Если $t_j{}^i=0$, то перенос не происходит (элемент игнорируется). Параметр P задает дополнительный режим. Если он задан в виде

C′ —′ (минус), то все элементы, соответствующие строке с номером $t_j^i<0$, переносятся с изменением знака на противоположный. Если параметр задан в виде C′ +′, то независимо от знака t_j^i знаки элементов строк не изменяются.

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $N \neq L$.

НЕВЕРНЫЕ КООРДИНАТЫ, если $|t_j^i|>M$ для $M>K$ или $|t_j^i|>N$ для $M \leqslant K$.

НЕВЕРНЫЙ СПИСОК ПАРАМЕТРОВ, если $|t_j^i|>K$ для $M>K$ или $|t_j^i|>M$ для $M \leqslant K$.

Перестроение матрицы по неповторяющимся элементам списка с внесением нулевых строк осуществляет программа PSTN

```
VP    PSTN, (&A, &S, &I, &R), &C
```

По списку S_i производится выборка строк с номерами t и засылка их в строку с номером j матрицы C (возможно изменение знака элементов строки, если $t_j^i<0$). Если значение встречалось ранее в списках $S_1, \ldots, S_{i-1}$ (сравнение производится без учета знаков), то строка с этим номером не выбирается, а в матрице C строка j заполняется нулями.

Дополнительный параметр может быть задан в виде

```
                    &R      ADSP   &R1
&R      DC      A (0) или  &R1 CSP t1, ..., tj, ..., tn.
```

В последнем случае каждый номер t_j сравнивается с элементами t_j^i $(j=1, \ldots, n)$. Если среди t_j найдено число, совпадающее по абсолютной величине с t_j^i, то перед новой строкой создается дополнительная нулевая строка.

Д: НЕВЕРНЫЕ РАЗМЕРЫ, если $N \neq L$.

НЕВЕРНЫЕ КООРДИНАТЫ, если $|t_j^i|>M$.

НЕВЕРНЫЙ СПИСОК ПАРАМЕТРОВ, если происходит переполнение матрицы C, т. е. количество создаваемых строк больше K.

Разрежение матрицы нулевыми строками осуществляется программой RAZR

```
VP    RAZR, (&A, &R), &C
```

Последовательно из матрицы A переносятся группы по t_j строк в матрицу C. Между группами вставляется нулевая строка.

Д: НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если $N \neq L$.

НЕВЕРНЫЙ СПИСОК ПАРАМЕТРОВ, если

$$M<\sum_{j=1}^{n} t_j \text{ или } K<\sum_{i=1}^{n}(t_i+1)-1.$$

Для определения размеров результирующей матрицы при перестроении рекомендуется использовать следующие операции:

```
      V     RK, (&S, &I, &R), &M
&M    DS    H
```

В поле &M заносится либо количество элементов q_i в подшкале S_i, либо q_i+p, где p — число совпадающих значений в шкалах S_i и R.

```
        V     RS,(&S,&S),&M
&M      DS    2H
```

В поле &M заносится количество различных по абсолютному значению элементов во всех подсписках S1, ..., SI, ... и минимальное количество первых шкал, содержащих все эти элементы. Если операция задана в виде V RS,(&S,&I),&M, то поиск будет производиться не по всем подшкалам, а только по первым $S_1, ..., S_i$.

Перестроение блочной матрицы A размером M×N по списку в блочную матрицу C размером 1×N осуществляется с помощью СА BPSC

```
&FC     SPFOR (&C)
BPSC    SASP  1,&A,&S,&FC
```

Первый подсписок S1 содержит номера суперстрок $t_1^1,\ldots,t_j^1,\ldots,t_{q_1}^1$ матрицы A, участвующих в построении матрицы C. Следующие подсписки S2, ..., SI, ..., SQ1 содержат номера строк в указанных ранее суперстроках матрицы A. Из этих строк $t_1^2,\ldots,t_{q_2}^2,\ldots,t_1^i,\ldots,t_{q_i}^i,\ldots,t_1^{q_1},\ldots,t_{q_{q_1}}^{q_1}$ создаются блоки в суперстроке матрицы C. Число столбцов в полученных блоках равно количеству столбцов в соответствующих блоках матрицы A. Количество строк равно $\sum_{i=2}^{q_1} q_i$.

Рассылка элементов строки A размером 1×N по шкале S в столбцы матрицы C размером K×L осуществляет программа RAST

```
        VPL   RAST,(&A,&S),&C
```

Здесь &S — имя поля, которое имеет следующий вид

```
&S      DC    A(* +4), H'n, l1,...,lj,...,l'n
```

Элементы строки a_{1j} рассылаются в матрицу C по координатам $(l_j, j): c_{l_j,j}=a_{1j}$ $(j=1,\ldots,q,$ где $q=\min\{N, n\})$.

Для $l_j=0$ рассылка в столбец j не производится.

Д: НЕВЕРНЫЕ КООРДИНАТЫ, если не выполняются условия $0<l_j\leqslant K$ или $j\leqslant N$.

Перестроение матрицы по строкам осуществляется с помощью программы PST

```
        VP    PST,(&A,&S),&C
&SP     DC    A(* +4), F'q', E1,...,En
```

Список S содержит n перестраивающих элементов E, каждый из которых определяет вид i-й строки матрицы C, создание которой ведется последовательно, начиная с первой строки до строки q.

Элемент E может быть записан в одной из указанных форм: H'1, j' — строка j матрицы A помещается в строку i матрицы C; E'c_i' — строка i заполняется константой c, с целью сокращения возможна запись H'n_i, j' — n_i последовательных строк, начиная со строки j, перемещается в матрицу C ($0<n_i<256$, иначе считается, что E задает константу c_i). Значение j может быть отрицательным. В этом случае знаки у элементов переносимых строк изменяются на противоположные.

Д: НЕКОРРЕКТНОСТЬ МАТЕМАТИЧЕСКОЙ ОПЕРАЦИИ, если $n>$K.

НЕСООТВЕТСТВУЮЩИЕ РАЗМЕРЫ, если N$\neq$L.

НЕВЕРНЫЕ КООРДИНАТЫ, если не выполняется условие $0<(j+n_i-1)<$M.

Перестроение матрицы по столбцам производится аналогично перестроению по строкам с помощью программы PSTO

```
VP    PSTO,(&A,&S),&C
```

Перестроение блочной матрицы A размером M×N в блочную матрицу C осуществляется с помощью CA BPST

```
&FC    SPFOR (&C)
BPST   SASP  1,&A,&S,&FC
```

Здесь & S — имя поля, которое имеет следующий вид:

& S DC A(*+4),F'p, q', H'r_1, k_1', $E_1^1,\ldots,E_{k_1}^1,\ldots$

В зависимости от величины параметров p и q производится перестроение по строкам ($p\neq0$, $q=0$) или столбцам ($p=0$, $q\neq0$). В первом случае размер результирующей блочной матрицы C равен p×N, во втором — M×q. Для построения каждой из p суперстрок (q суперстолбцов) служат k_j элементов $E_1^j,\ldots,E_{k_j}^j$. Количество строк в блоках суперстроки (столбцов в блоках суперстолбца j) равно r_j. Недостающий размер блоков берется из исходной матрицы.

Каждый перестраивающий элемент E_i^j может быть записан в одной из указанных форм: H'a, b, c, d' или H'a, 0', E'c_i^j'. Первая форма расшифровывается следующим образом: взять a строк (столбцов), начиная с b, из суперстроки (суперстолбца) c матрицы A и поместить последовательно в суперстроку (суперстолбец) j, начиная со строки (столбца) d, либо за предыдущими, если $d=0$. Если $b=0$, то последовательно вписываются a строк (столбцов) из константы c_i^j. Вписывание по одной и той же координате d приводит к суммированию элементов.

Стыковка нескольких матриц A1, ..., AN, имеющих одинаковое число строк, т. е. объединение их в одну матрицу C, число столбцов которой равно сумме вторых размеров исходных матриц, осуществляется программой STD

```
&FST   SPFOR (&C)
STD    SASP  1,&S,&FST
```

где &S — имя оператора ADSP, содержащего имена исходных матриц: &S ADSP &A1, &A2, ..., &AN; &FST — имя списка, содержащего фактический результат (&C) выполнения CA STD.

Стыковка нескольких динамических или смешанных статических списков S1, S2, ..., SN, имеющих одинаковый первый размер, т. е. объединение их в один список C, число элементов которого равно сумме элементов в S1, S2, ..., SN, выполняется программой STS

```
&FSTS    SPFOR (&C)
STS      SASP  1,&S,&FSTS
```

где &S — имя оператора ADSP, в качестве подсписков которого могут выступать простые динамические и сложные статические списки, подлежащие стыковке; &FSTS — имя списка, содержащего фактический результат (&C) выполнения CA STS.

Для изменения порядка следования столбцов матрицы на противоположный используется программа PSST

```
         VPL   PSST,,&A
```

где &A — имя матрицы, в которой программа PSST меняет местами столбцы 1 и N, 2 и (N—1), 3 и (N—2) и т. п.

Для сжатия матрицы по строкам, т. е. для перенесения нулевых строк в конец матрицы, используется программа SGM

```
         VPL   SGM,(&REZ),&A
```

где &A — матрица, строки которой переставляются; &REZ — количество ненулевых строк в матрице &A.

Если некоторая ненулевая строка стоит выше другой ненулевой строки до выполнения программы SGM, она останется выше другой и после выполнения программы.

7.7. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О МАТРИЦАХ

Для проверки, является ли матрица нулевой, можно использовать программу PNUL

```
         V     PNUL, (&A),&C
```

где &A — проверяемая матрица; &C — имя поля длиной 1 байт (&C DS X). В результате выполнения программы PNUL &C будет равно X'FF', если &A≠0, или равно X'00', если A=0.

Для определения количества ненулевых блоков в блочных матрицах необходимо использовать оператор

```
         V     PNNB,(&A1,...,&AI,...,&AN),&C
&C       DS    NF
```

где &AI — имена метаматриц (I=1, ..., N); &C — имя поля из N слов, в каждое из которых будет внесено число, равное числу ненулевых блоков для соответствующей метаматрицы.

Для определения абсолютных размеров блочной матрицы &A используется программа ARBM

```
V     ARBM, (&A), &C
```

где &C — имя поля

```
&C     DS    2H
```

в котором в результате выполнения программы ARBM будут сформированы абсолютные размеры &A.

Д: НЕРЕГУЛЯРНАЯ СТРУКТУРА, если один из размеров первой суперстроки или суперстолбца равен нулю.

Для определения максимальных размеров блоков блочной матрицы &A используется программа MRB

```
V     MRB, (&A), &C
```

где &C — имя поля

```
&C     DS    2H
```

в котором в результате выполнения программы MRB будут сформированы размеры наибольшего блока &A вне зависимости от того, существует этот блок реально или нет.

Для нахождения наибольшего или наименьшего значения матрицы используется группа программ, обращение к которым имеет следующий вид:

```
V     &I, (&A), &C
```

где &I — название операции (подпрограммы); &A — исходная матрица; &C — имя поля размером в одно или два двойных слова, в которое программа &I поместит результат своей работы.

Параметр &I может принимать следующие значения: MAX — определение максимального элемента ($C=\max\limits_{i,j} a_{ij}$); MAXA — определение максимального по абсолютной величине элемента ($C=\max\limits_{i,j}|a_{ij}|$); MIM — определение наименьшего и наибольшего элементов ($C1=\min\limits_{i,j} a_{ij}$; $C_2=\max\limits_{i,j} a_{ij}$).

Для проверки совпадения матриц A и B, т. е. для отыскания максимальной по модулю относительной разности соответствующих элементов матриц $\left(C=\max\limits_{i,j}\left|\frac{a_{ij}-b_{ij}}{a_{ij}}\right|\right)$ используется программа MAX1

```
V     MAX1, (&A, &B), &C
```

где &A, &B — исходные матрицы, &C — имя поля в формате двойного слова (&C DS D), в которое программа MAX1 поместит результат своего выполнения.

Для распечатки наименьших и наибольших элементов всех блоков блочной матрицы или простой матрицы используется СА BMIM.

```
BMIM     SASP  1, &A
```

где &A — исходная матрица.

Если & A — простая матрица, то печатается текст НАИМЕНЬШИЙ И НАИБОЛЬШИЙ ЭЛЕМЕНТЫ ПРОСТОЙ МАТРИЦЫ, вслед за которым выводятся значения $\min_{i,j} a_{ij}$ и $\max_{i,j} a_{ij}$. Если & A — блочная матрица, то СА BMIM формирует и печатает две матрицы, размеры которых равны размерам метаматрицы &A. Первая содержит наименьшие элементы блоков & A, вторая — наибольшие. Перед матрицами печатается соответствующий текст.

СА BMIM часто применяется после восстановлений матриц с магнитной ленты или диска, если есть сомнения в правильности чтения информации. Иногда, если из-за сбоя на носителе информация неправильно восстановилась и операционная система не обнаружила ошибку, происходит сдвиг восстанавливаемой информации, и числа будут иметь очень большие и малые порядки. В этом случае ошибка может быть легко обнаружена с помощью СА BMIM.

7.8. ВВОД-ВЫВОД ИНФОРМАЦИИ

С целью сокращения операторов ввода обычных матриц (VV) рекомендуется использовать операцию группового ввода

```
GVV     SASP  &M,&A1,...,&AJ,...,&AM
```

где & M — число вводимых матриц & A1, ..., & AM.

Ввод блочной матрицы A осуществляется с помощью команд

```
BVV     SASP  1,&A,&S
```

Метаматрица A должна иметь фон. Производится определение нулевых блоков в рабочем наборе данных. Выполнение ввода зависит от вида сложного списка & S. Если & S задана как NET, то по суперстрокам вводятся все блоки. Если & S имеет вид

```
&S      ADSP  NET
```

то ввод блоков не производится. Если & S задана в виде

```
&S      ADSP  &S1
&S1     CSP   i1,j1,i2,j2
```

то вводятся блоки с координатами (i_1, j_1), (i_2, j_2), ... и т. д. Определение и ввод всех блоков осуществляется по суперстрокам. Ввод каждого блока производится с помощью оператора VV.

Печать ненулевых блоков блочной матрицы A размером M×N осуществляется с помощью СА BPE.

```
BPE     SASP  1,&A,&S,&F1,&F2
&S      ADSP  &S1
&S1     CSP   k1, l1, i1, j1,...,kq, lq, iq, jq,...,kn, ln, in, jn
```

Для каждой группы элементов шкалы & S1: k_q, l_q, i_q, j_q производится печать по суперстрокам, если $k_q \neq 0$, и по суперстолбцам, если $l_q \neq 0$. Печать всегда начинается с блока i_l, j_l. Если $k_q > (N+1-i_q)$ или $l_q > (M+1-j_q)$, то печатаются блоки следующей суперстроки

или суперстолбца, начиная с j-го суперстолбца или i-й суперстроки. Если &S задано как NET, то производится печать всех ненулевых блоков матрицы.

Параметры &F1, &F2 определяют форматизованную печать блоков (см. оператор PE). Если они заданы как NET, то выполняется неформатизованная печать. Если &F1 и &F2 указывают на поле

```
&F1      DC    H'0,1'
```

будет напечатана структура блоков матрицы &A: нулевые числа будут указаны точками, ненулевые — звездочками.

Для подготовки информации с плавающей точкой к печати используется программа PCPC

```
                    V      PCPC,(&C,&F),&AD
```

где &C — имя поля, содержащего число с плавающей точкой; &F — формат символьного представления информации, задаваемый так же, как в операторе PE; &AD — адрес поля, в котором будет сформирована полученная для печати информация. Он может быть задан в виде символического имени либо в виде имя + число.

Если число не помещается в указанный формат, поле будет заполнено звездочками.

Пример 7.8.1

Пусть нужно напечатать строку СКОРОСТЬ=XX.YYY. Значение скорости находится в поле S. Фрагмент программы может иметь следующий вид:

```
         V        PCPC,(S,FF),SKO+11
         TEKST    SO
         . . . . . . . . . . . . . . . . . . .
S        DS       D
FF       DC       H'5,3'
SKO      DC       CL18    'СКОРОСТЬ='
```

Число будет размещено в поле SKO, начиная с адреса SKO+11. Для его представления потребуется 5 цифр, десятичная точка, возможно, знак числа, т. е. всего 7 позиций. Поэтому поле SKO имеет длину 18 позиций.

Для распечатки содержимого каталога матриц на магнитной ленте после восстановления или сохранения матриц из этого каталога используется программа PEKT

```
V      PEKT,(&K),*
```

где &K — имя каталога.

На печать выводятся: имя каталога, число описываемых им матриц, число записей в наборе данных, который описывает каталог; для каждой матрицы печатаются ее имя, тип (простая или блочная), размеры и начальный ленточный адрес.

Для распечатки элементов списков используется программа PESP, обращение к которой в зависимости от типа списка может иметь следующий вид:

```
VPL    PESP,,&S1
```

где & S1 — простой динамический список;

V PESP, (& S2), & S2

где & S2 — смешанный статический список;

V PESP, (& S3, & S3), & S3

где & S3 — простой статический список;

V PESP, (&I), &S4

где & S4 — сложный список; & I — имя поля в формате H, содержащего номер подсписка, который будет печататься.

7.9. ПРИМЕР ПРОГРАММЫ НА ВХОДНОМ ЯЗЫКЕ ППП СМПО

В качестве примера, иллюстрирующего программирование на СМПО, рассмотрим стандартный алгоритм LANC (см. приложение 4), реализующий метод Ланцоша для решения проблемы (7.19). Алгоритм является упрощенным вариантом СА RDML (см. разд. 7.5).

Входными параметрами алгоритма являются: KN — фактор Холецкого, полученный программой BRXL из матрицы K, вектор-строка B, задающий значение начального вектора; вектор-строка M, содержащий матрицу масс; N — максимально допустимое число итераций; N1 — количество собственных значений, которое нужно получить. Результатами будут матрицы Q, ALFA, BETA, содержащие векторы Ланцоша, числа α_i, β_i.

Выполнение СА начинается с печати текущего времени и названия алгоритма. Затем определяются результирующие и промежуточные матрицы. После задания начальных значений начинается цикл по этапам. На j-м этапе вычисляются β_j и $\mathbf{q}_j$. Далее проверяется: если на этом этапе уже выполнялась ортогонализация, то можно переходить к решению системы уравнений. В противном случае вычисляется $\mathbf{W}_{j-1}$ по формуле (7.24). Если $\max \mathbf{W}_{j-1} \leqslant \varkappa = 10^{-8}$, то можно переходить к решению системы уравнений, в противном случае вектор $\mathbf{r}_j$ ортогонализуется ко всем $\mathbf{q}_i$, $i=1$, $j-1$ и вычисление α_j и $\mathbf{r}_j$ повторяется. После решения системы уравнений вычисляются β_j и $\mathbf{q}_j$, а вектор Ланцоша $\mathbf{q}_j$ передается в рабочий набор данных. Далее с помощью QL-алгоритма определяются собственные значения z_i^j для редуцированной трехдиагональной матрицы, диагональ которой составляют a_i, $i=1, ..., j$, а поддиагональ — β_i, $i=2, ..., j$. Найденные z_i^j сравниваются с полученными на предыдущем этапе z_i^{j-1} с точностью $\varkappa$. Если сошлось не менее N1 форм, то работа алгоритма завершается; иначе начинается выполнение следующего этапа.

Отличительной особенностью описанного алгоритма является то, что он ориентирован на решение больших задач; как фактор Холецкого, так и векторы Ланцоша находятся во внешней памяти и размещаются в ОП для решения системы уравнений или ортогонализации соответственно.

Глава 8. Взаимодействие пользователя СМПО с операционной системой

8.1. БИБЛИОТЕКА ПРОЦЕДУР СМПО

Для связи пользователя СМПО с ОС в ППП имеется набор процедур. Процедура СМПО представляет собой набор операторов языка управления заданиями (ЯУЗ) [28], облегчающий пользователю формирование пакета заданий. Процедуры СМПО могут располагаться как в системной библиотеке процедур (SYS1 . PROCLIB), так и в личной библиотеке процедур СМПО. Во втором случае эта библиотека перед началом работы должна присоединяться к библиотеке SYS1 . PROCLIB с помощью команды оператора LIB или с помощью явного указания ее в процедуре системного ввода.

Процедуры СМПО при вводе задания пользователя интерпретируются программой системного ввода ОС ЕС.

Трансляцию программ СМПО выполняет Макроассемблер ОС ЕС, причем допускается использование любой его версии. Однако для ускорения процесса трансляции рекомендуется использование Ассемблера 2, модули которого содержатся на дистрибутивной магнитной ленте СМПО. К тому же при использовании Ассемблера 2 диагностика компилятора может быть выдана на русском языке. В описываемых ниже процедурах СМПО предполагается использование именно этой версии Ассемблера. В ней по умолчанию включены следующие режимы работы:

DECK, LIST, XREF(FULL), ALIGN, ESD,
RLD, MES(R), LINECOUNT(55), FLAG(0), SYSPARM ( );

— выключены OBJECT, RENT, TEST, BATCH, TERM, LEN. Если предполагается использование другой версии, тексты процедур должны быть изменены.

В процедурах СМПО применяется параметрическое задание режимов компилятора и Редактора связей. Точнее, с помощью символического параметра PASM пользователь может изменить значение одного режима Ассемблера, а с помощью параметра PLKED — одного режима Редактора связей. Изменение большего числа режимов достигается кодированием параметра PARM для соответствующего пункта задания.

Рассмотрим распределение наборов данных СМПО, которые используются процедурами пакета.

Библиотека загрузочных модулей (BZMSMPO) содержит все СПЕ и монитор пакета в загрузочном виде. Библиотека исходных модулей пакета (BIMSMPO) содержит исходные тексты программ СМПО. Макробиблиотека ППП СМПО (MACSMPO) содержит макроопределения операторов входного языка пакета. Если предполагается использование Ассемблера 2 и его программы не помещены в общую библиотеку (SYS1 . LINKLIB), они помещаются в отдельный набор данных (SYS1 . ASMLIB). Тексты процедур предполагают использование набора SYS1 . LINKLIB.

В процедурах предполагается, что все эти наборы располагаются на одном томе. В противном случае тексты процедур должны быть изменены.

Процедуры рассчитаны на использование Редактора связей уровня 44К, 64К или 88К [28]. При использовании Редактора уровня 128К тексты процедур должны быть изменены.

Все наборы печати направляются процедурами СМПО в выходной класс А.

Тексты процедур СМПО даны в приложении 3.

8.2. СОСТАВЛЕНИЕ ПАКЕТА ЗАДАНИЙ В СМПО

Пакет заданий в СМПО составляется в соответствии с общими требованиями ОС ЕС [28]. Каждое задание должно начинаться оператором JOB, заканчиваться пустым оператором и может состоять из нескольких пунктов задания. В дальнейшем мы опускаем операторы JOB и пустые операторы. При этом рекомендуется использование процедур СМПО.

Процедура вызывается оператором EXEC ЯУЗ, имеющим следующий формат:

//*имя* EXEC *имя-проц, операнды*

Здесь *имя* — необязательный параметр, задающий имя пункта оператора EXEC, может быть опущен; *имя-проц* — имя вызываемой процедуры СМПО; *операнды* — необязательные операнды, используемые для замещения стандартных значений, принятых в процедурах по умолчанию.

После оператора EXEC могут следовать DD-операторы ЯУЗ, изменяющие и добавляющие DD-операторы процедуры. Синтаксис ЯУЗ требует, чтобы сначала были указаны изменяющие операторы, причем использовать их нужно в том же порядке, что и в процедуре. Затем в любом порядке могут следовать добавляемые DD-операторы.

Трансляция программ СМПО производится с помощью процедуры SMPOT. По умолчанию в этой процедуре включены режимы ассемблера ESD, LIST, XREF; выключены — OBJECT, RLD, RENT.

Параметр ESD обеспечивает распечатку таблицы внешних имен транслируемой программы, LIST — ее текста, XREF — таблицу перекрестных ссылок. Объектный модуль не формируется (NOOBJECT), словарь перемещений не печатается (NORLD), реентерабельность программы не проверяется (NORENT).

Если такой набор режимов устраивает пользователя, то для трансляции программы во входном потоке (т. е. набитых на перфокарты или набранных на дисплее) он может написать:

```
//EXEC SMPOT
//SYSIN DD *                                    (8.1)
```

текст программы.

Оператор (8.1) генерируется операционной системой и может быть опущен.

Если пользователю нужно изменить значение одного режима Ассемблера, он может указать:

// EXEC SMPOT, PASM = *режим* (8.2)

текст программы.

Здесь режим — один из режимов Ассемблера. Если требуется изменить значение нескольких режимов, можно указать:

// EXEC SMPOT, PARM. ASM = *режим 1, режим 2,*

... *режим* N' (8.3)

текст программы.

В этом случае параметр PARM оператора (40) процедуры SMPOT (см. приложение 3) игнорируется, и включенным оказывается указанный при генерации Ассемблера 2 режим RLD, если в списке режимов (8.2) не присутствует NORLD.

Если программа пользователя записана в библиотеку исходных модулей (БИМ) под именем *«имя 1»*, то для ее трансляции можно указать:

// EXEC SMPOT,N = *имя* 1

Если процедуры СМПО находятся в системной библиотеке процедур или подключены к ней командой оператора LIB, то трансляция программы из библиотеки может быть выполнена с помощью команды оператора START:

S SMPOT, N = *имя* 1 (8.4)

Например, трансляция программы SL, находящейся в БИМ, может быть выполнена командой

S SMPOT, N = SL

Для трансляции, редактирования связей, записи исходного модуля в БИМ и загрузочного в библиотеку загрузочных модулей (БЗМ) (в СМПО такая операция называется «каталогизацией») используется процедура SMPOKATB. Она состоит из трех пунктов. В первом пункте выполняется программа SMPOGENR, являющаяся модифицированным вариантом утилиты IEBGENER [28]. Модификация сводится к переименованию DD-операторов утилиты: вместо оператора SYSIN утилиты используется оператор с именем SYSIN1, а вместо SYSUT1 — SYSIN. Это сделано для того, чтобы не было необходимости в кодировании оператора

// SYSUT1 DD *

поскольку после переименования требуемый вместо него оператор (8.1) необязателен, так как генерируется операционной системой.

Программа SMPOGENR помещает исходный модуль в библиотеку исходных модулей. Она не печатает никаких сообщений (оператор 60 процедуры отменяет печать сообщений утилиты). Пра-

вильность ее выполнения можно контролировать по коду возврата программы, который будет напечатан в листинге. При правильном завершении код возврата будет равен нулю.

Второй пункт процедуры — ассемблирование исходного модуля — выполняется аналогично ассемблированию в процедуре SMPOT. Отличие состоит в том, что в процедуре SMPOKATB по умолчанию выключены режимы XREF и ESD, включенные в процедуре SMPOT. Это отменяет распечатку таблицы перекрестных ссылок (NOXREF) и таблицы внешних имен (NOESD).

Третий пункт процедуры — редактирование связей. В результате его выполнения Редактор запишет сформированный загрузочный модуль в БЗМ. В пункте явно указаны следующие режимы Редактора связей: REUS, SIZE=(120K, 32K), LIST, MAP. Параметр REUS обеспечивает повторную используемость программы, благодаря чему при неоднократном обращении к ней используется та же версия программы; параметр SIZE задает распределение ОП для Редактора связей; параметр LIST обеспечивает печать управляющих операторов Редактора, если они есть; параметр MAP вызывает распечатку плана редактируемого модуля.

Если указанный набор режимов устраивает пользователя, то для каталогизации программы во входном потоке он может указать:

// EXEC SMPOKATB, N = *имя* 1 (8.5)

текст программы (8.6)

Здесь *имя* 1 — имя программы, которое указано в тексте программы в операторе STP, STAL или PROG.

Процедура SMPOKATB записывает в БИМ исходный модуль отдельным разделом под именем «*имя* 1», а в БЗМ — загрузочный модуль под именем, состоящим из префикса «SMPO» и из имени «*имя* 1». Например, программа SL записывается в БЗМ под именем SMPOSL.

Если нужно, для изменения значения одного из режимов ассемблера можно использовать параметр PASM аналогично оператору (8.2), а для изменения одного из режимов Редактора — параметр PLKED.

Например, если при каталогизации программы UM не требуется распечатка текста программы, то вместо (8.5) можно указать:

// EXEC SMPOKATB, N = UM, PASM = NOLIST

Если не нужен план загрузочного модуля, можно использовать оператор

// EXEC SMPOKATB, N = UM, PLKED = NOMAP

Если не требуется ни текст программы, ни план загрузочного модуля, можно указать

// EXEC SMPOKATB, N = UM, PASM = NOLIST, PLKED = NOMAP

Порядок следования параметров N, PASM, PLKED произволен

Если требуется изменить значение нескольких режимов ассемблера, их можно указать через поле PARM, как в операторе (8.3), если же изменяются несколько режимов Редактора связей, можно использовать поле PARM пункта LKED

// EXEC SMPOKATB, N = *имя* 1, PARM. LKED = '*режим* L1..., (8.7)

режим LN'

Здесь *режим* L1, ..., *режим* LN — режимы Редактора связей. В этом случае поле PARM оператора (190) процедуры SMPOKATB игнорируется, и перечисленные в нем параметры более не действительны. Если пользователь желает их сохранить, он их должен указать в (8.7).

Если в одном операторе EXEC используется как PARM.ASM, так и PARM.LKED, первый должен быть указан перед вторым.

Например:

// EXEC SMPOKATB, N = OPR, PARM. ASM = 'RENT, XREF',

// PARM. LKED = 'RENT, NOMAP'

Если программа пользователя уже находится в БИМ и необходимо оттранслировать, отредактировать и записать загрузочный модуль в БЗМ, что часто требуется при возникновении сбоев в БЗМ, то для этого также можно использовать оператор типа (8.5) без (8.6). В этом случае первый пункт процедуры не выполнит никаких действий и завершится с кодом возврата 4. Пункты трансляции и редактирования связей будут выполняться обычным образом. Это же действие можно произвести с помощью команды оператора START при тех же условиях, что и (8.4):

S SMPOKATB, N = *имя* 1

Здесь также можно указать дополнительные параметры (PASM, PLKED, PARM.ASM, PARM.LKED).

Кроме общей БЗМ, в СМПО для отладки программ можно широко использовать личные и временные БЗМ.

Личные БЗМ может иметь каждый пользователь или группа пользователей СМПО. Они являются постоянными, т. е. сохраняются после выполнения, и могут использоваться неоднократно. Временные библиотеки создаются в рамках одного задания и в конце его уничтожаются. Они удобны при отладке программ.

Для трансляции, редактирования связей и записи загрузочного модуля в личную или временную БЗМ используется процедура SMPOKTVR. Процедура состоит из двух пунктов: ассемблирования и редактирования связей, работа которых аналогична процедуре SMPOKATB. Кроме параметров, описанных выше, требует задания параметр BZM (предполагаем, что стандартный набор режимов устраивает пользователя)

// EXEC SMPOKTVR,N = *имя* 1, BZM = *имя б* (8.8)

текст программы (8.9)

Здесь *имя б* — имя личной или временной библиотеки загрузочных модулей. Если параметр BZM опущен, предполагается использование временной библиотеки с именем &BZM. Если библиотека не была создана ранее, то при записи в нее первого модуля после (8.9) требуется указать

// LKED. SYSLMOD DD DISP = (,*дисп*), SPACE = (*ед*, (*к*, *пр*, *огл*)) (8.10)

Здесь *дисп* — { PASS — для временной библиотеки; KEEP — для личной библиотеки;

ед — { CYL — при выделении памяти под библиотеку в цилиндрах; TRK — при выделении в дорожках;

к — количество первоначально выделяемых единиц дисковой памяти; *пр* — количество дополнительно выделяемых на диске единиц памяти при исчерпывании их первоначального количества; *огл* — количество 256-байтовых блоков, выделяемых для оглавления библиотеки (один блок требуется на каждые 6 модулей).

Модуль записывается в БЗМ под именем, состоящим из префикса SMPO и из имени «*имя* 1».

Рассмотрим примеры использования процедуры SMPOKTVR

// EXEC SMPOKTVR, N = KAP, PASM = XREF

текст модуля KAP

// LKED. SYSLMOD DD DISP = (,PASS),

// SPACE = (TRK, (3, 2, 1))

Здесь модуль KAP помещается во вновь создаваемую временну́ю БЗМ под именем SMPOKAP. Имя библиотеки &BZM, ей выделяется 3 дорожки, 1 блок оглавления и 2 дорожки будут добавлены при переполнении. При ассемблировании будет напечатана таблица перекрестных ссылок:

// EXEC SMPOKTVR, BZM = RAZ, N = PTTE

текст модуля PTTE

Модуль PTTE помещается в уже существующую личную библиотеку RAZ под именем SMPOPTTE. Режимы Ассемблера и Редактора связей не изменяются.

Для трансляции, редактирования и выполнения основной программы пользователя используется процедура SMPOTRV.

Процедура состоит из трех пунктов. Первый пункт процедуры — ассемблирование исходного модуля — выполняется так же, как и в процедуре SMPOKATB. Второй пункт процедуры — редактирование связей — выполняется аналогично редактированию в процедуре SMPOKATB. Отличие состоит в том, что в процедуре SMPOTRV отменен режим Редактора REUS, поскольку основная программа не является повторно используемой.

Третий пункт процедуры — выполнение основной программы. Если набор режимов Ассемблера и Редактора устраивает поль-

зователя, обращение к процедуре SMPOTRV будет таким:

// EXEC SMPOTRV [,R = *рол*] [,T = *вр*], [,VEL = *ррп*] (8.11)

// [, PR = *прн*] [,VR = *имя б*]

текст основной программы

Здесь *рол* — размер раздела ОП в Кбайтах, выделенный для решения задачи. По умолчанию равен 150; *вр* — предельное процессорное время выполнения пункта задания в минутах, по окончании которого выполнение будет прекращено. По умолчанию равно 900; *ррн* — размер рабочего набора данных в цилиндрах. По умолчанию равен 5; *прн* — дополнительное количество цилиндров (приращение), выделяемое на диске, если будет исчерпана отведенная под рабочий набор память. Такое дополнительное выделение при наличии неиспользуемой памяти на диске может быть выполнено до 15 раз. По умолчанию равно 5; *имя б* — имя временной или личной БЗМ пользователя, откуда должна быть прочитана часть СПЕ. Правила поиска СПЕ следующие: модуль сначала ищется в библиотеке «*имя б*», и только, если поиск был неудачным, используется общая БЗМ. По умолчанию личная БЗМ не используется.

Во время выполнения используются следующие наборы данных: STEPLIB — БЗМ шага задания. Это может быть личная или временная БЗМ, указанная в (8.11), или общая БЗМ, если личная библиотека не используется; оператор без DD-имени, присоединенный к оператору STEPLIB. Он описывает общую БЗМ; SYSPRINT — набор данных печати; DISKSMPO — рабочий набор данных; DUMP — набор данных для записи дампа при аварийном завершении задания, вызванным программным прерыванием или диагностикой СМПО; SYSIN — входной набор данных на перфокартах. Предполагается, что этот набор пустой, т. е. данные во входном потоке отсутствуют.

Если присутствуют данные во входном потоке, должен быть добавлен оператор

// GO.SYSIN DD * ,DLM = '//'

после которого могут располагаться перфокарты исходных данных.

Для выдачи дампа при аварийном окончании, отличном от программного прерывания и диагностики СМПО, после исходных данных, если они есть, добавляется оператор

// GO.SYSUDUMP DD SYSOUT = A

Рассмотрим примеры использования процедуры

//EXEC SMPOTRV

текст основной программы

Решается относительно небольшая задача. Используются значения, принятые по умолчанию: для расчета выделено 150К ОП и 5 цилиндров на диске. Личные и временные библиотеки не используются, исходные данные отсутствуют:

// MIN EXEC SMPOTRV, R = 7∅, VEL = 2, P ASM = XREF, T = 5

текст основной программы

//GO .SYSIN DD * ,DLM = '//'

исходные данные

Решается маленькая задача, в качестве ресурсов ей выделяется 70К ОП, 2 цилиндра на диске. Во избежание бесконечного зацикливания время решения не будет превышать 5 мин. При трансляции печатается таблица перекрестных ссылок. Имеются исходные данные во входном потоке:

//MAX EXEC SMPOTRV, VEL = 140, PR = 15, R = 700, VR = RAZ

текст основной программы

//GO. SYSIN DD * ,DLM = '//'

исходные данные

//GO. KAT DD DSN = KAT, VOL = SER = LENTA, DISP = OLD,

// UNIT = 5010, LABEL = 3

В примере решается большая задача. В качестве ресурсов ей выделяются 700К ОП, 140 цилиндров на диске плюс по 15 цилиндров будет добавляться при переполнении выделенной области. Используются данные во входном потоке. Вводится информация с магнитной ленты. В качестве личной БЗМ используется библиотека.

Приведем также пример использования временной БЗМ

// EXEC SMPOKTVR, N = DP (8.12)

текст СПЕ DP

/LKED. SYSLMOD DD DISP = (,PASS), SPACE = (TRK,(3, 1, 1)) (8.13)

// EXEC SMPOKTVR, N = UMP

текст СПЕ UMP

// EXEC SMPOTRV, VR = ' &BZM' (8.14)

текст основной программы

//GO. SYSIN DD *, DLM = '//'

исходные данные

В этом примере две СПЕ (DP и UMP) помещаются во вновь созданную временную библиотеку. Ее имя не указано (8.12), (8.13) и по умолчанию ей присваивается имя &BZM. Затем выполняется основная программа, использующая эту библиотеку при отыскании вызываемых СПЕ (8.14). Хотя модули DP и UMP есть в общей БЗМ, будут использованы СПЕ из временной библиотеки, поскольку при вызове любой СПЕ сначала просматривается временная библиотека, и только если в ней отсутствует необходимая СПЕ, поиск выполняется в общей БЗМ.

Если требуется использовать несколько личных и (или) временных библиотек, можно поступить следующим образом. Пусть поиск

СПЕ должен выполняться сначала в библиотеке VR1, затем в VR2, а затем — в общей БЗМ — BZMSMPO

// EXEC SMPOTRV, VR = VR1 (8.15)

текст основной программы

// GO. STEPLIB DD

// DD DSN = VR2, DISP = SHR, VOL = SER = SMPO00, UNIT = SYSDA (8.16)

// DD DSN = BZMSMPO, DISP = SHR, VOL = SER=SMPO00, UNIT=SYSDA

//GO.SYSIN DD * ,DLM = '//' (8.17)

исходные данные

В примере предполагается, что все БЗМ расположены на томе SMPO00 (8.16), (8.17). Библиотека VR1 указывается в (8.15), VR2 — в (8.16), BZMSMPO — в (8.17). Если бы нужно было использовать еще нескольких личных БЗМ, их описание, аналогичное (8.16), появилось бы между (8.16) и (8.17). В операторе (8.15) также могут указываться и другие параметры из (8.11).

Если основная программа «закаталогизирована» в общую или личную БЗМ, то для ее выполнения используется процедура SMPOV. Эта процедура состоит из одного пункта задания, который аналогичен третьему пункту процедуры SMPOTRV:

// EXEC SMPOV, N = *имя*1 [,R = *роп*] [,T = *bp*] [,VEL = *ррн*]

// [,PR = *прн*], [,VR = *имя б*].

Здесь *имя*1 — имя вызываемой основной программы. Остальные параметры необязательны и имеют тот же смысл, что и в (8.11).

Примеры применения процедуры SMPOV:

// EXEC SMPOV, N = GFOR, R = 950, VEL = 190

Вызывается основная программа GFOR. В качестве ресурсов ей выделяются 950К ОП и 190 цилиндров на диске. Временные библиотеки не используются

// EXEC SMPOV, N = FOR4, VR = RAZ

Вызывается основная программа FOR 4. Для отыскания ее и (или) вызываемых ей СПЕ используется личная библиотека RAZ. Ресурсы ей выделяются по умолчанию.

Глава 9. Дополнительные возможности ППП СМПО

9.1. ОРГАНИЗАЦИЯ РАЗНОЯЗЫКОВЫХ МОДУЛЕЙ

В СМПО существует возможность использования подпрограмм, написанных на фортране. Наличие в составе ЕС ЭВМ оптимизирующего компилятора фортрана, дающего хороший объектный код, позволяет широко использовать фортран для реализации неблоч-

ных операций или операций над отдельными блоками блочных матриц. В то же время пользователь должен знать, что даже небольшая фортран-программа требует много ОП из-за подключения к ней на этапе редактирования связей модулей библиотеки фортран. Так как модули в СМПО вызываются динамически, каждая из фортран-программ будет довольно большого размера. Поэтому можно объединять в один загрузочный модуль те фортран-программы, которые вызывают друг друга. В состав такого модуля должен входить модуль-связка на языке СМПО.

Если модуль-связка оформлен в виде СА, после команд вызова фортран-программы

```
L       15, =V (имя-форт)        (9.1)
BALR    14,15                    (9.2)
```

должна стоять команда

```
L       15,0(13)                 (9.3)
```

Здесь *имя-форт* — имя фортран-программы, присутствующее в операторе FUNCTION, SUBROUTINE или ENTRY.

Если модуль-связка оформляется в виде СТП, то в операторе STP должен быть указан регистр с номером от 2 до 12, перед командами (9.1), (9.2) должна быть записана команда

```
L       13,8(13)                 (9.4)
```

обеспечивающая область сохранения для СТП, после команд (9.1), (9.2) необходимо указать

```
L       13,4(13)                 (9.5)
```

восстанавливая в регистре 13 адрес системной (СМПО) области сохранения.

При использовании арифметических подпрограмм библиотеки фортран, а также фортран-программ, не содержащих операторов ввода-вывода и отладки, можно не присоединять к создаваемому загрузочному модулю сервисные программы библиотеки фортран (IBCOM#, IHCERRM и т. п.). Для этого нужно указать режим NCAL при работе Редактора связей и (или) использовать его управляющие операторы INCLUDE и LIBRARY [28]. При этом размер загрузочного модуля сокращается, но диагностику ошибок, которые могут быть в фортран-программе, будет выполнять СМПО. Если модуль IBCOM# подключается, то перед вызовом первой фортран-подпрограммы загрузочного модуля нужно указать:

```
L       15, =V(IBCOM #)          (9.6)
BAL     14,64(15)                (9.7)
```

оформляя их так же, как (9.1), (9.2).

Перед вызовом фортран-программы в модуле-связке нужно сформировать список адресных параметров и загрузить его адрес в регистр 1. Порядок следования параметров в списке должен быть тем же, что и в фортран-операторах SUBROUTINE,

FUNCTION или ENTRY. При передаче матриц нужно придерживаться следующих правил.

1. Матрица должна находиться в ОП. Если связка оформлена в виде СТП, то используемые в ней матрицы уже размещены в ОП монитором при вызове СТП и специальных действий по размещению предпринимать не нужно; если связка оформлена в виде СА, матрицу нужно разместить в ОП оператором ZAPOM.

2. Размеры матрицы, если они не являются постоянными, нужно включить в число параметров и переписать их из формата полуслова (H) в формат полного слова (F).

3. В фортран-программе передаваемая матрица должна быть описана в операторе DOUBLE PRECISION, причем размеры ее должны быть переставлены. Например, если передается матрица A размером M×N, ее нужно описать: DOUBLE PRECISION A (N, M).

В результате этих действий фортран-программе будет доступна матрица, которая является транспонированной по отношению к передаваемой из СМПО. Это вызвано тем, что в фортране матрицы хранятся в ОП по столбцам, а в СМПО — по строкам.

Подпрограммы-функции и библиотечные функции фортрана возвращают результат своей работы в нулевом общем регистре или в нулевом регистре с плавающей точкой в зависимости от типа функции (INTEGER или REAL). Этот результат можно сохранить в модуле-связке сразу после вызова функции.

При входе в фортран-программу содержимое регистров с плавающей точкой не сохраняется. Поэтому при необходимости перед вызовом фортран-программы их можно сохранить, а после вызова — восстановить.

Для автоматического формирования списка параметров и вызова фортран-программы из СА можно использовать оператор FORT:

```
&NAME    FORT  &P1,&P2,...,&PI,..., &PN
```

Здесь &NAME — имя фортран-программы, которое должно присутствовать в фортран-операторе SUBROUTINE, FUNCTION или ENTRY; &PI — фактический параметр, передаваемый в фортранпрограмму.

Если вызывается подпрограмма-функция, то в зависимости от ее типа результат ее выполнения будет в нулевом общем регистре либо в нулевом регистре с плавающей точкой.

Рассмотрим подробнее формат списка. Его элементами могут быть имена матриц, переменных или обозначения размеров матриц. Размеры идентифицируются цифрами 1 или 2 (для первого или второго размеров) перед именами матриц. Имена переменных должны быть взяты в скобки, имена матриц должны быть вне скобок, обозначения размеров могут быть как в скобках, так и вне их. Порядок следования фактических параметров должен соответствовать порядку следования формальных параметров.

Рассмотрим примеры использования оператора FORT.

Пример 9.1.1

Передаются три матрицы: A(M×N), B(K×K), C(L×M):

```
        ZAPOM (A, B, C)
PR1     FORT  A, B, C, 1A, 2A, 2B, 1C                    (9.8)
```

Здесь с помощью записи 1A передается размер M, 2A—N, 2B—K, 1C—L. Оператору СМПО (9.8) соответствуют фортран-операторы

```
SUBROUTINE PR1 (A, B, C, M, N, K, L)
DOUBLE PRECISION A(N, M), B(K, K), C(M, L)
```

Теперь в фортран-программе доступны матрицы A, B и C, транспонированные по отношению к матрицам СМПО.

Пример 9.1.2

Пусть передаются две матрицы: A(3×M), B(1×N), два числа в формате полуслова (H): I и J и три числа в формате с плавающей точкой двойной точности (D): X, Y, Z.

```
PR2     FORT A, B, 2A, 2B, (I, J, X, Y, Z)               (9.9)
```

Здесь предполагается, что A и B запомнены ранее. Фортран-операторы:

```
SUBROUTINE PR2 (A, B, M, N, I, J, X, Y, Z)
DOUBLE PRECISION A(M, 3), B(N), X, Y, Z                  (9.10)
INTEGER*2 I, J
```

Вместо (9.9) можно было указать

```
PR2 FORT A, B, (2A, 2B, I, J, X, Y, Z)
```

Матрица B в (9.10) описана в виде вектора, так как один из ее размеров равен 1. Можно было бы описать ее и в виде прямоугольной матрицы.

При динамическом вызове загрузочного модуля, содержащего фортран-программу (операторы SA, SASP, V, VPL, VP, SOZ), требуется принятие дополнительных мер для правильной работы операторов ввода-вывода фортрана. В противном случае печатаемая фортран-программой информация при ее повторном динамическом вызове перекроет информацию, выведенную при первом обращении, введена будет с перфокарт не информация, находящаяся за введенной ранее, а та же самая. Добиться правильной работы операторов ввода можно путем подсчета числа введенных карт, введения его в список параметров фортран-программы и пропуска этого количества записей при вводе. При печати нужно после последней выводимой записи указать оператор фортрана ENDFILE для набора данных печати. Если фортран-программа вызывается несколько раз как статически (т. е. из модуля-связки), так и динамически (т. е. модуль-связка вызывается операторами SA, SASP, V, VP, VPL, SOZ), вместо оператора ENDFILE в тексте фортран-программы нужно смоделировать его работу в модуле-связке после последнего обращения к фортран-программам

```
L       15, =V (IBCOM#)
CNOP    0,4
```

```
BAL     14,48(15)
DC      F' номер'
```

оформляя их так же, как (9.1), (9.2). Здесь *номер* — номер набора данных печати.

9.2. ПОДСИСТЕМА ФОРТРАН-СМПО

Для облегчения освоения и расширения круга пользователей пакета в его состав включена подсистема фортран-СМПО, благодаря которой пользователь получает возможность непосредственно из своей прикладной программы, написанной на фортране, обращаться ко всем средствам ППП СМПО. Эта возможность обеспечивается модулем интерфейса, который представляет собой программу, написанную на языке ассемблер с использованием макросредств СМПО и служит для связи фортран-программы с монитором пакета. Модуль интерфейса имеет ряд точек входа, каждая из которых реализует определенную функцию. Требуемые действия запрашиваются с помощью оператора

CALL *имя* $(a_1, a_2, \ldots, a_n)$

где *имя* — имя точки входа в модуль интерфейса, реализующей необходимую функцию; a_1, a_2, ..., a_n — список фактических аргументов.

Большинство точек входа реализует функции, аналогичные соответствующим операторам языка СМПО. Их имена и форматы списков фактических параметров совпадают с соответствующими мнемоническими кодами и форматами операторов языка СМПО.

Опишем вкратце алгоритм функционирования подсистемы фортран-СМПО при выполнении задания пользователя. Модуль интерфейса, получив через одну из своих точек входа управление от прикладной программы, выполняет действия, аналогичные тем, которые выполняются макроассемблером при трансляции оператора СМПО. После этого в большинстве случаев управление должно быть передано монитору пакета. Однако при выполнении программы пользователя СМПО следит за правильностью выполнения операций, причем обработка программных прерываний в СМПО отличается от их обработки в среде фортрана. Поэтому модуль интерфейса устанавливает среду СМПО и только после этого передает управление монитору пакета. Выполнив требуемую пользователем функцию, монитор возвращает управление модулю интерфейса, который восстанавливает среду фортрана и, в свою очередь, передает полученные результаты и управление фортран-программе пользователя.

Как уже отмечалось, пользователю, программирующему на фортране, доступны все средства пакета, в том числе и все типы данных СМПО. Вместе с этим модуль интерфейса обеспечивает передачу двумерных массивов фортрана удвоенной точности для обработки программами СМПО. Такая возможность реализуется путем заведения соответствующих дескрипторов для массивов фор-

трана, так что монитор их воспринимает как динамические матрицы, запомненные в оперативной памяти.

В качестве примера программы на процедурно-ориентированном входном языке фортран-СМПО рассмотрим сложение матрицы А, введенной с перфокарт, с единичной матрицей В

```
      REAL*8 A, B, C
С ИНИЦИАЛИЗАЦИЯ ПОДСИСТЕМЫ ФОРТСМПО. ОБЛАСТЬ
ДАННЫХ — 30К.
      CALL    SMPO(30)
      CALL    OPR(A, 10, 10, B, 10, 10, C, 10, 10)
      CALL    VPL('VV', A)
      CALL    SOZ('EM', B)
      CALL    VP('SL', A, B, C)
```

9.3. ВИЗУАЛИЗАЦИЯ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Используя возможности организации связи СМПО-фортран, в ППП СМПО можно задействовать устройства машинной графики. В рамках ППП СУМРАК для визуализации графической информации используются базисные и функциональные подпрограммы (БФП) для графопостроителей [28], которые могут использоваться также для вывода на графический дисплей. Для разных графопостроителей и дисплеев используются различные базисные программы, но форматы обращения к ним одинаковы. Поэтому графические подпрограммы (ГПР) ППП СУМРАК отредактированы без них, а также и без других подпрограмм библиотеки фортран (режим NCAL редактора связей [28]). Обращение к ГПР выполняется с помощью графического диспетчера, который сначала вызывает загрузчик ОС ЕС [28], формирующий из ГПР готовый для выполнения модуль путем подключения соответствующих устройству БФП, а затем передает выполнение этому модулю. Сама ГПР состоит из модуля-связки на языке СМПО и фортран-части, вызывающей БФП.

9.4. РАСПАРАЛЛЕЛИВАНИЕ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

Для более полного использования возможностей современных ЭВМ в СМПО можно организовать параллельные вычисления за счет разбиения заданий на несколько параллельно решаемых ветвей (подзадач) [28]. Такая возможность является особенно ценной для многопроцессорных ЭВМ. Для однопроцессорных машин распараллеливание также может привести к повышению эффективности расчетов за счет совмещения обработки центральным процессором одной подзадачи с вводом-выводом в других подзадачах.

Наличие в ЭВМ, используемой для расчета, достаточного размера ОП, нескольких устройств прямого доступа и операционной системы, имеющей возможность создания подзадач, является не-

обходимым условием для организации параллельных вычислений. Однако этого еще недостаточно. Так, например, пользователь, программирующий на фортране, лишен возможности использовать это средство ОС, так как в фортране нет соответствующих операторов. Непосредственная реализация этой возможности на языке ассемблер достаточно трудоемка, вероятно поэтому в современных ППП возможность распараллеливания не получила должного развития [1, 3, 25, 26].

В соответствии с требованиями ОС из всех подзадач выделяется одна, которая считается главной задачей. Остальные будут созданы этой задачей и получат статус подзадачи.

Основной информационной единицей в параллельном процессе является так называемая стандартная подзадача (СПЗ). СПЗ должна начинаться оператором STPZ. Выполняемая часть стандартной подзадачи должна завершаться оператором KSTPZ. Текст СПЗ должен заканчиваться оператором END.

Создание подзадачи и обращение к ней осуществляется с помощью оператора OBPZ, который должен кодироваться в главной задаче. Новой СПЗ выделяются требуемые для ее работы ресурсы ЭВМ: ОП для области данных и память для рабочего набора данных на диске. ОП по желанию программиста может быть выделена как из области данных главной задачи, так и из области памяти раздела. Подзадача может читать информацию из рабочего набора (РН) главной задачи, формируя результат в собственной РН. Операции ввода-вывода всех выполняемых задач выполняются реентерабельными программами: монитором и программой OPR.

Для передачи в главную задачу промежуточных и (или) окончательных результатов работы подзадачи используются операторы FAKPZ (в подзадаче) и FORPZ (в главной задаче). При выполнении оператора FORPZ решение главной задачи будет остановлено до тех пор, пока в подзадаче не будет выполнен оператор FAKPZ. При совместной работе этих операторов в РН главной задачи будет выделена память и перечисленные в них матрицы будут переписаны из РН подзадачи.

Для уничтожения подзадачи используется оператор UNPZ, кодируемый в главной задаче. Он приостанавливает ее выполнение до тех пор, пока подзадача не завершится.

Для повышения эффективности работы важное значение приобретает способ разделения алгоритма по подзадачам. При этом можно предложить следующие рекомендации.

1. Различные подзадачи должны быть по возможности информационно независимыми от результатов работы других подзадач. В противном случае синхронизация подзадач и обмен информацией между ними потребует дополнительных временных затрат.

2. Для однопроцессорной ЭВМ имеет смысл разделить по разным задачам процессорные и обменные части, дав последним более высокий приоритет. Если этого сделать не удается, то можно распределить работу так, чтобы время решения каждой из параллель-

но решаемых задач было примерно одинаково, и дать им равный приоритет.

3. Желательно для каждой задачи выделить РН на разных накопителях на магнитных дисках и, по возможности, на разных каналах. В противном случае время на перемещение механизма доступа накопителя между РН разных задач может свести на нет полученную за счет распараллеливания экономию времени.

4. Ту часть задачи, результаты выполнения которой наиболее велики по объему, выгодно оформить как главную задачу, чтобы уменьшить потери времени на передачу результатов работы из РН подзадачи в РН главной задачи.

Рассмотрим формат операторов оформления СПЗ. Оператор STPZ, с которого должна начинаться СПЗ, имеет следующий формат:

```
&NAME    STPZ   &SP [,&KB] [,ROB = &ROB]
```

Здесь &NAME — «короткое» имя СПЗ, содержащее не более четырех алфавитно-цифровых символов и начинающееся с буквы; &SP — список формальных параметров СПЗ, заключенный в скобки. Параметры перечисляются через запятые. Формат списка такой же, как в операторе STAL; &KB — число блоков стека дисковой памяти, создаваемого для подзадачи. По умолчанию равно 50; &ROB — размер ОП для области данных СПЗ в килобайтах. Это значение будет использоваться только в том случае, если размер области данных не будет явно указан при создании СПЗ оператором OBPZ.

Оператор KSTPZ, завершающий выполнение СПЗ, имеет следующий формат:

```
KSTPZ
```

Таким образом, СПЗ оформляется аналогично СА, причем вместо оператора STAL используется оператор STPZ, а вместо KSTAL — KSTPZ.

Вызов СПЗ осуществляется оператором OBPZ:

```
& NO OBPZ  &NAME, &ISP [,PRTY = &PRTY] [,ROB = &ROB] [,VD = GL]
```

Здесь &NO — имя, которое программист присваивает оператору OBPZ. Указание является обязательным. Оно должно быть уникальным в пределах СПЕ, содержащей оператор OBPZ; &NAME — имя СПЗ. Это имя должно появиться в операторе STPZ подзадачи; &ISP — имя оператора SPSA, содержащего список параметров для вызова СПЗ. Формат списка точно такой же, как и при вызове СА; &PRTY — число со знаком, которое алгебраически складывается с текущим приоритетом главной задачи. Результат будет текущим приоритетом СПЗ, если он не больше граничного приоритета пункта задания, установленного в операторе EXEC; в противном случае текущим приоритетом будет граничный. Поэтому для того, чтобы СПЗ имела более высокий приоритет, чем главная задача, в последней до оператора OBPZ должна быть выдана макрокоманда супервизора CHAP [28], уменьшающая ее текущий приоритет.

Если параметр опущен, приоритет СПЗ будет равен приоритету главной задачи; & ROB — размер ОП для области данных СПЗ в килобайтах. Если этот параметр опущен, будет выделен такой объем памяти ОП, какой указан в операторе STPZ; VD — тип выделения ОП для области данных СПЗ. Если параметр указан, то память выделяется из области данных главной задачи. Если он опущен, она выделяется из свободной памяти раздела, в котором выполняется пункт задания.

Для передачи результатов работы СПЗ в главную задачу используются операторы FORPZ и FAKPZ. Эти операторы связываются через специальный параметр, передаваемый через список входных параметров СПЗ аналогично тому, как связываются операторы SPFOR и SPFAK при вызове СА.

Формат оператора FORPZ (кодируется в главной задаче)

```
&PAR1    FORPZ (&F1,..., &FI,..., &FN), &NO
```

Здесь & PAR1 — имя фактического параметра, который присутствует в операторе SPSA и служит для передачи результатов из СПЗ. Место для него резервировать не нужно; & FI — фактический результат, который получает значение результата работы СПЗ; & NO — имя, присвоенное оператору OBPZ, который вызывает СПЗ.

Формат оператора FAKPZ (кодируется в СПЗ)

```
FAKPZ (&FO1,..., &FOI,..., &FON), &PAR2
```

Здесь & FOI — формальный параметр результата, значение которого будет присвоено фактическому результату из оператора FORPZ. Количество и порядок формальных и фактических параметров должны быть равны. & PAR2 — имя фактического параметра, который присутствует в списке параметров оператора STPZ и соответствует фактическому параметру & PAR1 оператора FORPZ.

Для уничтожения подзадачи после ее завершения используется оператор UNPZ

```
UNPZ  &NO
```

Здесь &NO — имя, присвоенное оператору OBPZ, создавшему уничтожаемую задачу.

Рассмотрим пример использования подзадачи (пример носит исключительно иллюстративный характер).

Пусть требуется решить две блочные системы уравнений: AX=B и CY=D. Предполагается, что разложение Холецкого для матриц A и C уже выполнено. Будем в главной задаче решать первую систему, в подзадаче — вторую.

Главная задача:

```
. . . . . .
PRIM     OBPZ    PZD1, SPIS, ROB =300, VD = =GL
BSS      SASP    1,A,B
FPZD1    FORPZ   (Y), PRIM
         UNPZ    PRIM
. . . . . .
```

```
SPIS     SPSA    1, C, D, FPZD1
   Подзадача:
PZD1     STPZ    (C, D, FPZD1)
         MVC     Y(8), D
         OPR     Y
BPER     SASP    1, D, Y
BSS      SASP    1, C, Y
         FAKPZ   (Y), FPZD1
         KSTPZ
D        GDS     (C, D)
         END
```

В главной задаче создается подзадача, после чего решают систему AX=B, причем X помещается на место B.

Затем главная задача ждет завершения передачи результатов, после чего подзадача уничтожается. В подзадаче создается матрица Y, в которую переписывается D, поскольку нельзя изменять в подзадаче матрицу D, созданную в главной задаче. Затем решается система CY=D, и результат ее передается в главную задачу. После этого выполнение подзадачи прекращается.

При выполнении задания, содержащего обращение к подзадачам, необходимо добавить DD-операторы для рабочего набора данных СПЗ и набора данных печати СПЗ:

```
//GO. DISKSMP1 DD UNIT=SYSDA, SPACE=(CYL, (ррн1, прн1))
//GO. SYSPRIN1 DD SYSOUT=A
```

Здесь *ррн1* — размер рабочего набора для подзадачи в цилиндрах; *прн1* — величина приращения памяти в цилиндрах для рабочего набора СПЗ при исчерпании первоначального количества.

Если в подзадаче присутствуют исходные данные, то для ввода их можно указать:

```
//GO. SYSIN1     DD *, DLM='//'
```

Если одновременно работает несколько подзадач, для каждой из них должны присутствовать аналогичные DD-операторы: для второй подзадачи — DISKSMP2 и SYSPRIN2, для третьей DISKSMP3 и SYSPRIN3 и т. д. Возможно использование операторов SYSIN2, SYSIN3 и т. д.

9.5. ОТЛАДКА ПРОГРАММ В ППП СМПО

ППП СМПО имеет развитые средства отладки программ. В соответствии с общей методологией отладки неинтерактивных программ, основными средствами отладки в СМПО являются трассировки и аварийные выдачи (дампы).

Трассировки в СМПО выполняются при включенном режиме слежения (см. оператор SLED). Режим слежения дает возможность

проследить порядок выполняемых модулей и (или) получить результаты матричных операций.

Для распечатки промежуточных результатов вычислений используется наиболее универсальная форма печати: бесформатная, при которой каждое число печатается в виде $a \cdot 10^b$, где a — мантисса, а b — порядок. Выборочная печать выполняется оператором ОТРЕ.

СМПО производит контроль машинных и системных операций в процессе выполнения программ для решения конкретной задачи пользователя.

В случае невозможности выполнения какой-либо операции система печатает диагностическое сообщение и прекращает выполнение программы.

Слежение за ходом выполнения машинных команд осуществляется центральным процессором вычислительной машины. Аварийная ситуация приводит к программному прерыванию. Некоторые из прерываний могут быть замаскированы и не произойдут, если соответствующий бит маски в слове состояния программы (биты 36—39) равен 1 (сообщения П.08, П.0А, П.0D, П.0Е). В начале выполнения программы пользователя замаскировано прерывание П.0Е. Маскирование и размаскирование прерываний может быть выполнено с помощью машинной команды SPM. Каждая причина прерывания идентифицируется кодом прерывания. Получив от процессора этот код, СМПО печатает причину сбоя, соответствующую этому коду.

Ниже перечисляются диагностические сообщения для пятнадцати кодов прерывания, характеризующих причину сбоя в работе процессора.

П.01 — НЕСУЩЕСТВУЮЩАЯ ОПЕРАЦИЯ. Процессор пытался выполнить команду с несуществующим кодом операции или отсутствующую на конкретной ЭВМ.

П.02 — ПРИВИЛЕГИРОВАННАЯ КОМАНДА. Команда, которая может выполняться только в состоянии «супервизор», встретилась в режиме «задача».

П.03 — НЕКОРРЕКТНАЯ КОМАНДА. Команда «выполнить», позволяющая выполнить одну команду, не входящую в нормальную последовательность команд, ссылается на другую команду «выполнить».

П.04 — ЗАЩИТА ПАМЯТИ. Процессор пытался обратиться к области памяти, защищенной от этого типа обращения.

П.05 — НЕПРАВИЛЬНАЯ АДРЕСАЦИЯ. Произошло обращение по адресу, значение которого выходит за пределы памяти конкретной ЭВМ.

П.06 — НЕПРАВИЛЬНАЯ СПЕЦИФИКАЦИЯ. Считается, что спецификация неправильна, если: адрес команды не кратен 2; адрес операнда не соответствует целочисленной границе, соблюдение которой требуется при выполнении данной команды; в команде, требующей указания регистра с четным номером, указан нечетный регистр; номер регистра с плавающей точкой не равен 0, 2, 4, 6; в опе-

рациях десятичной арифметики длина множителя или делителя превышает по размерам 15 цифр и знак (8 байтов) или поле первого операнда не больше второго.

П.07 — НЕПРАВИЛЬНЫЕ ДАННЫЕ. Считается, что данные неправильные, если: знак или коды цифр операндов в операциях десятичной арифметики, редактирования, преобразования в двоичную систему являются недопустимыми; в операциях десятичной арифметики неправильно перекрываются поля (младшие байты не совпадают); множимое при десятичном умножении не имеет достаточного количества нулевых старших байтов.

П.08 — ПЕРЕПОЛНЕНИЕ С ФИКСИРОВАННОЙ ТОЧКОЙ. Во время выполнения арифметических операций или арифметического сдвига происходит перенос из старшего бита целой части в знаковый разряд.

П.09 — НЕКОРРЕКТНОСТЬ ДЕЛЕНИЯ С ФИКСИРОВАННОЙ ТОЧКОЙ. Частное от деления превышает размер регистра, или попытка деления на ноль, или результат команды «преобразование в двоичную» превышает 31 бит.

П.0А — ДЕСЯТИЧНОЕ ПЕРЕПОЛНЕНИЕ. При выполнении десятичной операции старшие цифры результата теряются из-за того, что результат не умещается в отведенное ему поле.

П.0В — НЕКОРРЕКТНОСТЬ ДЕСЯТИЧНОГО ДЕЛЕНИЯ. Частное от деления выходит за пределы указанного поля.

П.0С — ПЕРЕПОЛНЕНИЕ ПОРЯДКА. Результат операции с плавающей точкой имеет характеристику, большую 127.

П.0D — ИСЧЕЗНОВЕНИЕ ПОРЯДКА. Результат операции с плавающей точкой имеет характеристику, меньшую 0.

П.0Е — ПОТЕРЯ ЗНАЧИМОСТИ. Результат сложения или вычитания с плавающей точкой имеет нулевую мантиссу и ненулевую характеристику.

П.0F — НЕКОРРЕКТНОСТЬ ДЕЛЕНИЯ С ПЛАВАЮЩЕЙ ТОЧКОЙ. Попытка деления на число с нулевой мантиссой.

При выполнении большинства системных (матричных и сервисных) операций производится проверка правильности условий выполнения. Как правило, такая проверка осуществляется в самой стандартной программной единице, реализующей конкретную операцию.

В случае обнаружения ситуации, которая может привести к неверному выполнению операции, выполняется оператор DIAG, вызывающий печать диагностического сообщения, либо оператор EOJ, немедленно завершающий выполнение задания.

При возникновении программного прерывания или обработке оператора DIAG производится аварийная выдача, заголовок которой имеет следующий вид.

имя программы
ДИАГНОСТИКА СИСТЕМЫ /SMPO/ ВР : *xx*ЧАС *yy*МИН *zz*СЕК
РЕШЕНИЕ ЗАДАЧИ ПО ВИНЕ ПОЛЬЗОВАТЕЛЯ ПРЕКРАЩЕНО!
ошибка

АДРЕС СПИСКА ПАРАМЕТРОВ: *аааааааа*

Здесь *имя программы* — это имя стандартной программной единицы СМПО, которая выполнялась в момент ошибки или завершила выполнение непосредственно перед ней; *xx yy zz* — текущее время обнаружения ошибки; *ошибка* — расшифровывается тип ошибки. При срабатывании оператора DIAG печатается соответствующее коду ошибки сообщение. При возникновении программного прерывания печатается текст ПРОГРАММНОЕ ПРЕРЫВАНИЕ ПО АДРЕСУ *бббббббб*, а в следующей строке — тип программного прерывания. Адрес *бббббббб* указывает на команду, при выполнении которой произошла ошибка; *аааааааа* — адрес списка параметров операции, название которой (имя программы) было указано выше. Этот список формируется макроассемблером при обработке оператора обращения (V, VP, VPL, SOZ, SA, SASP, NOCH, KOCH, OPR, MATR, ZAPOM, UDAL). По этому списку можно найти в дампе параметры операции и проанализировать их.

Рассмотрим подробнее некоторые из сообщений СМПО, приведенные при описании опаратора DIAG и являющиеся общесистемными. ОБЛАСТЬ ДАННЫХ ЗАПОЛНЕНА! — это сообщение может появиться при выполнении любого из операторов обращения ППП СМПО. Оно означает, что в области данных недостаточно места для матриц, необходимых для выполнения операции. Наиболее вероятная причина — малый размер области данных, определяемый параметром R=*pon* оператора EXEC SMPOTRV (EXEC SMPOV) языка управления заданиями и параметром &RZ оператора PROG. Также возможно, что испорчен дескриптор какой-либо матрицы, так что ее размеры оказались очень велики. МАТРИЦА, ОБЪЯВЛЕННАЯ ОПЕРАТОРОМ MATR, НЕ ЗАПОМНЕНА! — сделана попытка работы с матрицей, объявленной оператором MATR, для которой не был выполнен оператор ZAPOM. ОШИБКА ВВОДА-ВЫВОДА! — ошибка при обращении к одному из внешних устройств, доступ к которому обеспечивает СМПО. В этом случае перед первой строкой стандартной диагностики СМПО печатается информация, подготовленная макрокомандой SYNADAF ОС ЕС [28]. В ней указывается DD-имя набора данных, при обращении к которому произошла ошибка, тип ошибки, адрес устройства и дисковый (ленточный) адрес записи, к которой выполнялось обращение (если сбой произошел при обращении к диску или ленте). Ошибки ввода-вывода при обращении к набору данных печати, к входному набору, к наборам для долговременного хранения данных (магнитные ленты и библиотеки) обычно бывают вызваны сбоем в аппаратуре. Ошибка при обращении к рабочему набору данных может быть вызвана как сбоем аппаратуры (наиболее часто встречается DATA CHECK, EQUIP CHECK, SEEK CHECK), так и ошибками в программе (ДИСКОВЫЙ АДРЕС НЕВЕРЕН, PROGRAM CHECK). Некоторые ошибки (NO REC FOUND, WRNG LENGTH REC) могут быть вызваны как сбоем оборудования, так и ошибками в программе. Уточнить это можно, запустив задание повторно. Ошибки в программе могут произойти либо из-за того, что испорчен дескрип-

тор какой-либо матрицы, либо при попытке работы с матрицей, уничтоженной оператором KBL. НЕДОСТАТОЧНЫЙ РАЗМЕР ПАМЯТИ ДЛЯ ОЧЕРЕДИ — размера области данных недостаточно для размещения участвующих в очереди матриц. Ошибка вызвана либо неправильным параметром оператора NOCH, либо малым размером области данных. УДАЛЯЕМАЯ МАТРИЦА НЕ НАЙДЕНА — сделана попытка удалить матрицу с выводом ее на диск, но эта матрица не была запомнена в том блоке запоминания, который подлежит удалению. Как правило, эта ошибка связана с нарушением вложенности блоков ZAPOM — UDAL.

После заголовка аварийной выдачи выводится сам дамп. Печатается текст: СОДЕРЖИМОЕ ОБЩИХ РЕГИСТРОВ 0—15: , вслед за которым выводится содержимое регистров в момент обнаружения ошибки, в шестнадцатеричном формате. Затем печатается содержимое регистров в момент выдачи дампа. Затем печатается заголовок ТЕКСТЫ АКТИВНЫХ ПРОГРАММ, вслед за которыми печатаются шестнадцатеричные тексты загрузочных модулей, которые были активны в момент обнаружения ошибки. Модули называются активными, если они выполняются или прервали свое выполнение, обратившись к подпрограмме. Первым печатается дамп модуля, в котором обнаружена ошибка, затем дамп вызвавшего его модуля и т. д. Последним печатается дамп основной программы.

После текстов активных программ печатается заголовок: СОДЕРЖИМОЕ ОБЛАСТИ ДАННЫХ: , вслед за которым выводится шестнадцатеричный дамп области данных, начиная с первой участвующей в операции матрицы и до конца области. Затем выводится содержимое блоков и таблиц управляющей программы, относящихся к данному заданию.

Выполнение программы диагностики заканчивается макрокомандой ОС

```
ABEND 8,DUMP
```

что означает завершение задания с кодом пользователя 8. Если в пакете задания присутствует оператор

```
// GO. SYSUDUMP DD SYSOUT = A
```

операционная система напечатает дамп, содержащий всю информацию, относящуюся к заданию.

Обычно при отладке нет необходимости выдавать полный дамп (это относится как к дампу СМПО, так и к дампу ОС). Выдачу оставшейся части дампа можно отменить, сняв печать.

Рассмотрим подробнее организацию сегмента программы диагностики, который печатает тексты активных программ. Алгоритм его работы требует знания структуры управляющих блоков ОС ЕС [28] и имеет следующий вид.

1. Определяем адрес таблицы векторов связей (CVT) ОС ЕС. Из CVT извлекаем и запоминаем информацию о режиме ОС (организация таблиц супервизора связей в режиме MFT отличается от их организации в режимах MVT и SVS).

2. В CVT находится адрес блока управления задачей (TCB). Из TCB извлекаем адрес первого элемента очереди активных программ (RB), описывающего первую программу в цепочке активных программ.

3. Если ОС работает в режиме MFT, переходим к п. 6.

4. Из блока RB определяем адрес элемента справочника содержимого (CDE). Из CDE находим имя печатаемой программы и адрес списка экстентов (XL) данного модуля.

5. Извлекаем адрес начала программы и ее длину из блока XL. Переходим к п. 7.

6. Извлекаем имя модуля, адрес его начала и длину из блока RB.

7. Печатаем дамп активной программы и ее имя.

8. Используя блок RB, переходим к следующему блоку RB в цепочке активных программ. Если цепочка закончилась, завершаем работу; в противном случае переходим к п. 3.

Такая организация работы сегмента, печатающего тексты активных программ, уменьшает объем печатаемой информации и облегчает поиск ошибки, давая возможность пользователю сразу определить цепочку активных программ. Так, например, текст монитора никогда не печатается: будучи реентерабельной программой, он не изменяет своего состояния, и печатать его нет необходимости.

9.6. ГЕНЕРАЦИЯ И АДАПТАЦИЯ КОНКРЕТНОЙ ВЕРСИИ ППП СМПО. МОБИЛЬНОСТЬ ППП СМПО

ППП СМПО версии 1.0 поставляется пользователю на стандартно помеченной магнитной ленте SMPO01. Лента содержит пять наборов данных, созданных утилитой IEBCOPY ОС ЕС [28].

Первым набором данных (BZM) выгружена библиотека загрузочных модулей пакета. Библиотека имеет размер блока 3625 байт и может быть загружена на диски любого типа. Для ее размещения требуется 100 блоков оглавления и 8 цилиндров на диске ЕС-5066.

Вторым набором данных (BIM) выгружена библиотека исходных модулей ППП СМПО. Она имеет размер блока 3360 байт и может быть загружена на диски любого типа. Для ее размещения требуется 50 блоков оглавления и 24 цилиндра на диске ЕС-5066.

Третьим набором данных (MAC) выгружена макробиблиотека пакета. Размер ее блока по умолчанию равен 3360 байтам. Если макробиблиотека ОС ЕС SYS1.MACLIB имеет больший размер блока, то при восстановлении макробиблиотеки СМПО ей нужно установить тот же размер блока. Для ее размещения требуется 10 блоков оглавления и 1 цилиндр на диске ЕС-5066.

Четвертым набором данных (PROC) выгружена библиотека процедур пакета. Ее можно загружать либо в отдельную библиотеку, либо в библиотеку процедур ОС ЕС SYS1.PROCLIB. Для ее размещения требуется 1 блок оглавления и 2 дорожки на диске ЕС-5066.

Пятым набором данных (SYS1. ASMLIB) выгружена библиотека, содержащая программы Ассемблера 2, который рекомендуется

использовать при трансляции программ ППП СМПО. Размер блока этой библиотеки программ равен 7294 байтам, поэтому восстановить ее на диски типа ЕС-5050 нельзя. В этом случае можно пользоваться обычной версией ассемблера, входящей в состав ОС ЕС, или сгенерировать Ассемблер 2 с дистрибутивной ленты макроассемблерной системы. Для размещения этой библиотеки требуется 2 блока оглавления и 18 дорожек на диске ЕС-5066.

Подготовка ППП СМПО к работе начинается с восстановления описанных библиотек с магнитной ленты. Затем требуется настроить процедуры СМПО на параметры конкретной вычислительной установки (см. гл. 8). В процессе этой настройки корректируются групповые имена устройств, имена томов, имена и параметры обрабатывающих программ ОС (Ассемблер, Фортран, Редактор связей). Могут быть изменены значения параметров, принятых в СМПО по умолчанию.

Процедуры СМПО могут быть записаны в системную библиотеку процедур ОС ЕС (SYS1.PROCLIB) или подсоединены к ней. Если используется отдельная библиотека, размер ее блока должен быть равен размеру блока SYS1.PROCLIB (по умолчанию он равен 3360). Подсоединение библиотек может быть выполнено командой оператора LIB или с помощью специальной процедуры системного ввода, в которой библиотека процедур СМПО должна быть сцеплена с библиотекой SYS1. PROCLIB.

После настройки библиотек ППП СМПО готов к работе. Монитор пакета сам определяет режим работы операционной системы (MFT, MVT, SVS) и настраивается на него; модуль SMPOEXCP определяет тип накопителя на магнитных дисках и его характеристики (число дорожек в цилиндре, размер дорожки, величина промежутка между физическими записями).

Разработчик ППП СМПО имеет версию пакета, функционирующую в ДОС ЕС. Мобильность пакета в том, что обе версии совместимы на уровне входного языка снизу вверх, т. е. программы, написанные для СМПО-ДОС, работают и в СМПО-ОС. При этом объектные коды, полученные при трансляции программ СМПО в ДОС и ОС, значительно отличаются друг от друга. Тексты проблемных программ пакета одинаковы в общих версиях, а некоторые системные программы (монитор, программа OPR, программы долговременного хранения информации) различны.

Глава 10. Использование ППП СМПО для создания программного обеспечения МКЭ и решения задач, сформулированных в матричном виде

Данная глава носит иллюстративный характер: приведены примеры решения задач из различных областей техники с помощью СМПО. Для этого применен ряд методов строительной конечноэлементной механики, а также иные инженерные методы, где ис-

пользуются матричные формулировки. В большинстве случаев приводятся простые примеры, но подразумевается, что СМПО рассчитана на решение задач практически неограниченной сложности.

10.1. ПРОГРАММНАЯ ПОДСИСТЕМА МЕТОДА СИЛ

Определение напряженно-деформированного состояния подкрепленных оболочек суперэлементным методом сил

Это наиболее испытанная подсистема программного обеспечения. Алгоритм суперэлементного метода сил [35] — один из наиболее сложных в МКЭ, и он стимулировал развитие СМПО. Проверка достоверности получаемых результатов описана в книгах [10, 35]. Приведем блок-схему подсистемы (рис. 10.1), на которой использованы обозначения: ROMS — расчет одного суперэлемента; DWI — получение матриц для сочленения; WJ — формирование и решение систем уравнений для сочленения; BR — корректировка матриц усилий с учетом взаимодействия объединенных суперэлементов; W — вычисление матриц узловых усилий; PRF — учет присоединенного фактора; LM — вычисление матрицы нагружения; BFR — формирование матриц податливости и усилий; BF — объединение матриц и т. п.

Наличие подсистемы метода сил в ППП позволяет рассчитывать конструкции с непосредственным определением напряжений,

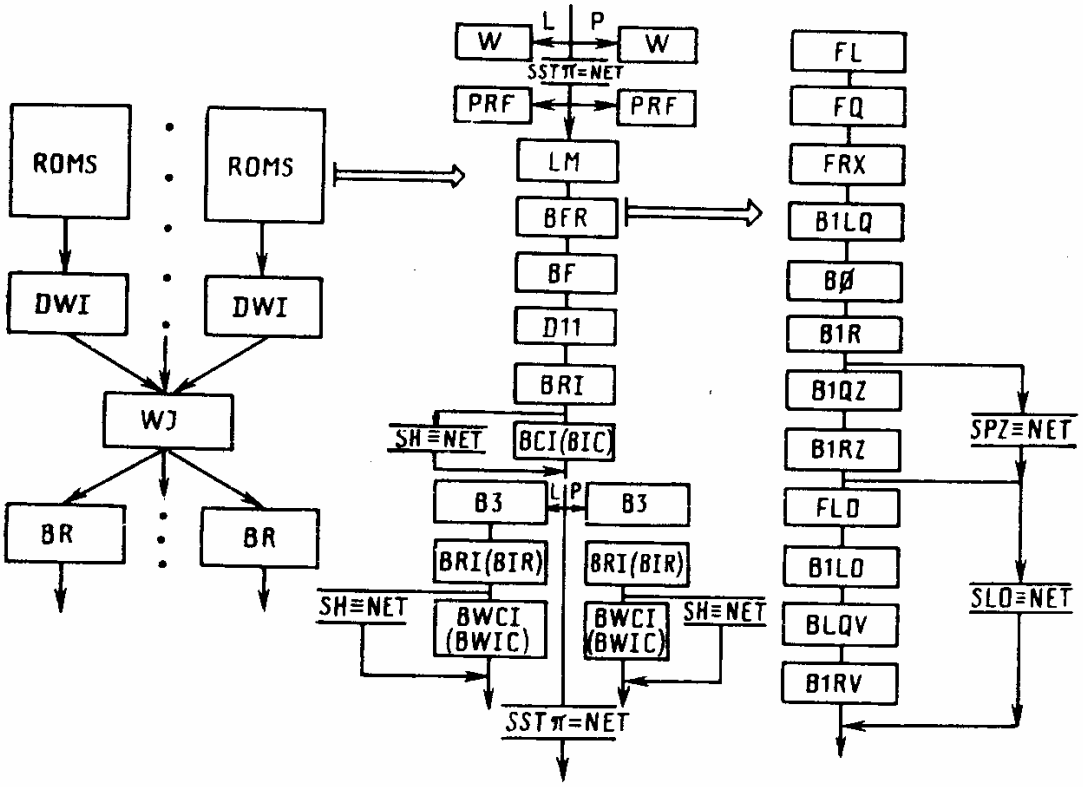


Рис. 10.1. Схема программной реализации суперэлементного метода сил

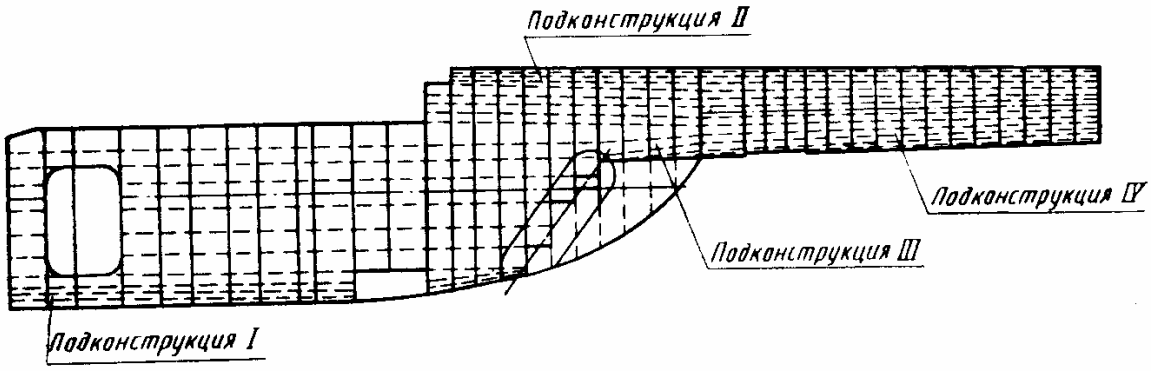


Рис. 10.2. Расчетная схема конструкции

выполнять параллельно с методом перемещений расчеты для сопоставления напряженно-деформированного состояния (НДС) и собственных колебаний, выполнять физически нелинейный расчет для оценки несущей способности подкрепленных оболочек.

На рис. 10.2 показана расчетная схема подкрепленной оболочки, рассчитанной суперэлементным методом сил. На рис. 10.3, 10.4 приведены эпюры нормальных и касательных напряжений на одном из участков конструкции.

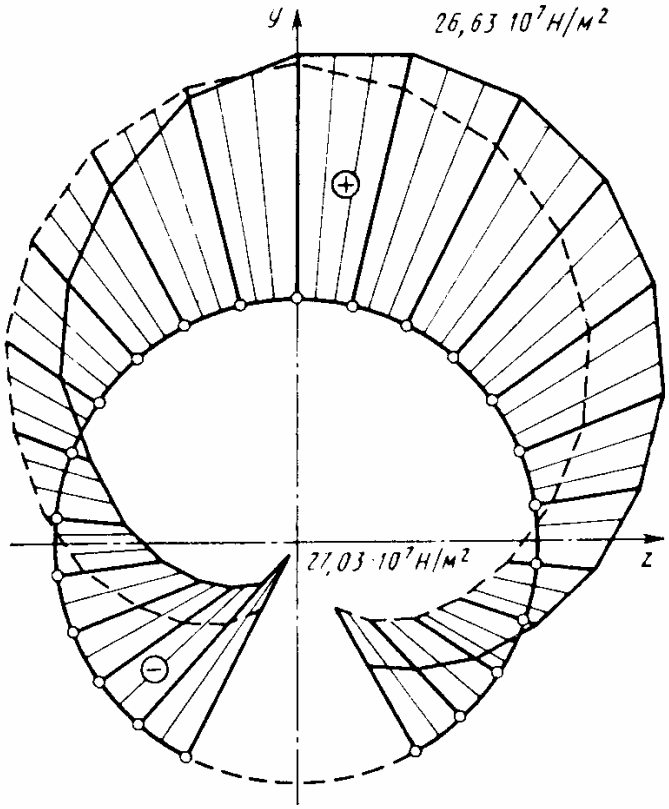


Рис. 10.3. Эпюра нормальных напряжений в продольном наборе в сечении 3 подконструкции IV:

——— случай нагружения 1; — — — случай нагружения 2

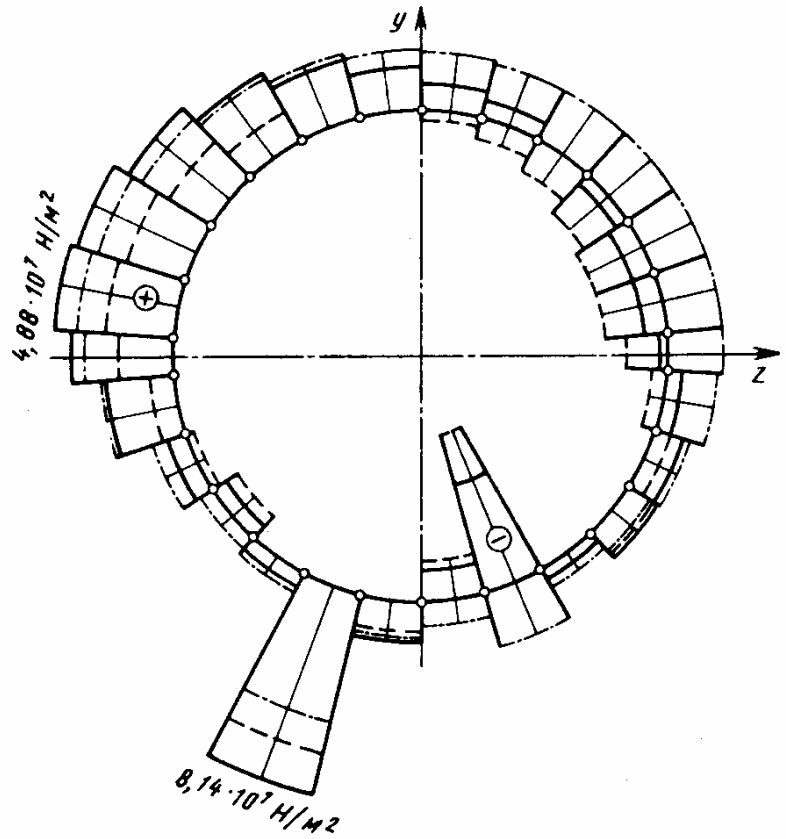


Рис. 10.4. Касательные напряжения в обшивке между сечениями 2 и 3 подконструкции IV:

——— случай нагружения 1; — — — случай нагружения 2

Расчет на собственные и вынужденные колебания подкрепленных оболочек

Конечноэлементный метод сил был применен в работе [35] для расчета на собственные колебания подкрепленной оболочки, там же приведена теория алгоритма, примеры расчета различных конструкций и сравнение с экспериментом; показано хорошее соответствие результатов. Расчеты были проведены на ЭВМ 2-го поколения (М-220). Аналогичные результаты были получены на СМПО, что является доказательством правильности ее работы.

Приведем пример расчета более сложной конструкции с построением матрицы податливости суперэлементным методом сил. Расчетная схема объекта показана на рис. 10.5. Матрица податливости сформирована с помощью суперэлементного метода сил, при этом конструкция расчленялась на 3 суперэлемента; массовая модель сформирована с учетом 130 динамических степеней свободы. На рис. 10.6 показаны собственные формы колебаний конструкции в вертикальной плоскости. Эту же конструкцию рассчитывали на вынужденные колебания от внешнего воздействия гармонической силой. Использовался метод разложения по собственным формам без учета демпфирования по формуле

$$V = -\frac{A_0 \left(A_0^{\tau} M A_0\right)^{-1} A_0^{\tau} \overline{P}}{w^2} + \sum_{i=1}^{n} \frac{\rho_i \rho_i^{\tau} \overline{P}}{\left(\omega_i^2 - W^2\right) \rho_i^{\tau} M \rho_i},$$

где V — форма вынужденных колебаний; W — частота вынуждающей силы; ω_i — частота собственных колебаний; $\overline{P}$ — амплитуда вынуждающей силы; ρ_i — собственная форма; A_0 — матрица равновесия; M — диагональная матрица масс.

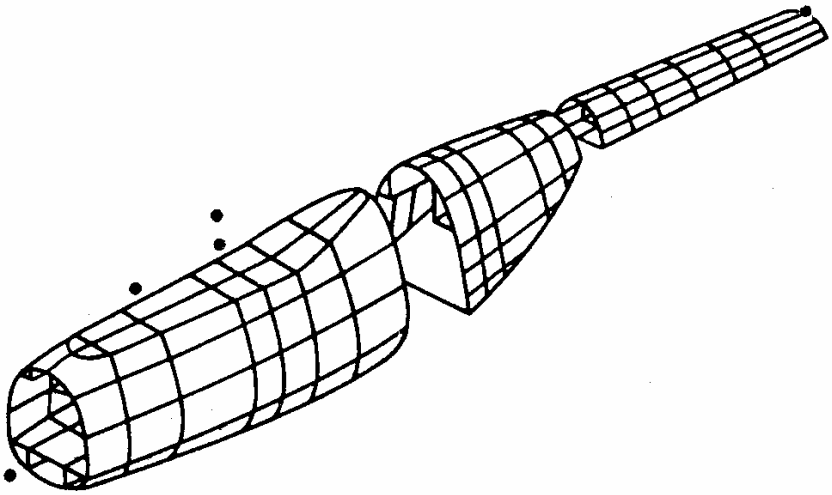


Рис. 10.5. Расчетная схема конструкции (точками показаны сосредоточенные массы)

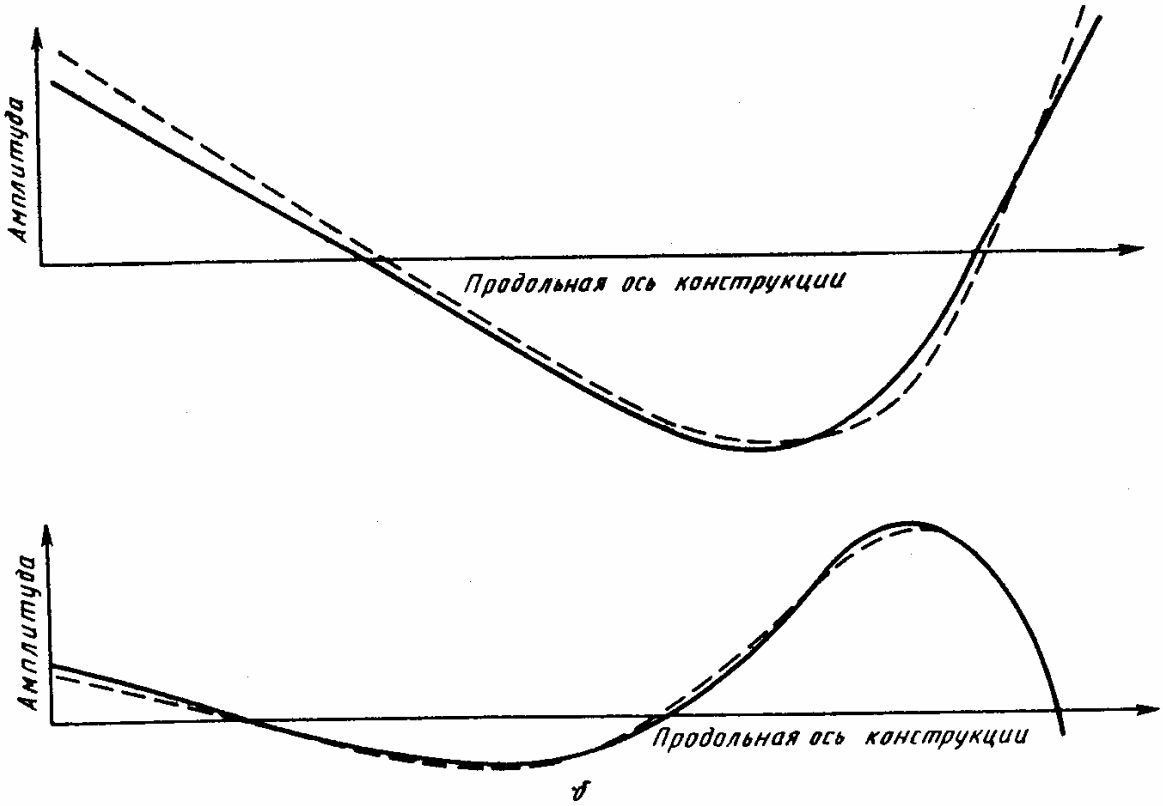


Рис. 10.6. Собственные формы колебаний 1-го (а) и 2-го (б) тонов:
——— эксперимент; — — — расчет

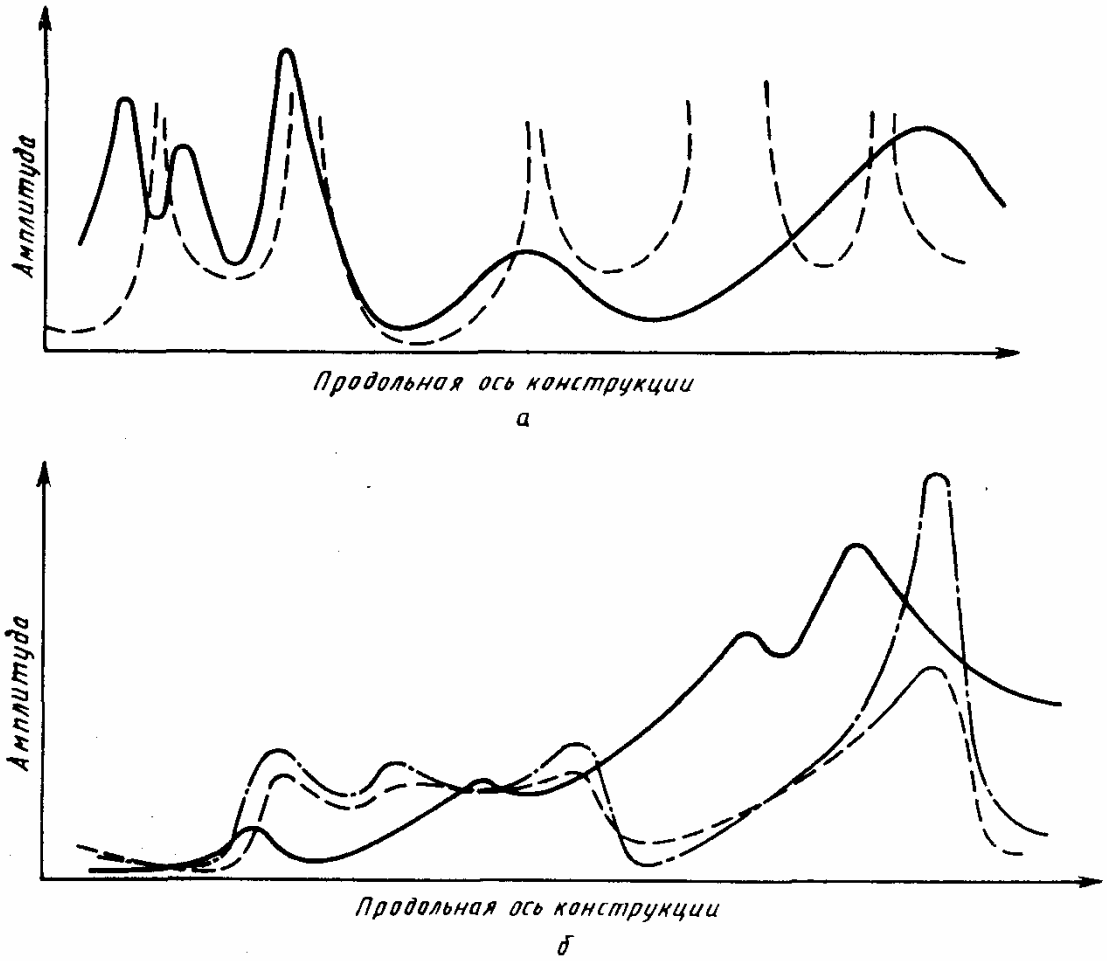


Рис. 10.7. Амплитудно-частотные характеристики:

а — без учета демпфирования: ——— эксперимент; — — — расчет; *б* — с учетом демпфирования: ——— эксперимент; — · — расчет при коэффициенте демпфирования 0,02; — — — расчет при коэффициенте демпфирования 0,05

На рис. 10.7 приведены амплитудно-частотные характеристики для частот 6—21 Гц. Вдали от резонансных пиков согласование с результатами экспериментов удовлетворительное.

Для учета демпфирования при определении динамической реакции конструкций на гармоническое воздействие создан алгоритм, реализующий формулу

$$V = -\frac{\rho_0 \rho_0^T \overline{P}}{W^2} + \sum_{i=1}^{n} \frac{\rho_i \rho_i^T \overline{P}}{\sqrt{(\omega_i^2 - W^2)^2 + (2\xi\omega_i W)^2}},$$

где ρ_0 — формы колебаний конструкции как твердого тела; ρ_i — формы упругих собственных колебаний, нормированных относительно матрицы масс; ξ — коэффициент демпфирования.

Проверка алгоритма на задачах, решенных аналитически, а также прямым методом для систем уравнений динамического равновесия, показала его работоспособность и высокую точность получаемых результатов.

На основе этого алгоритма проведен расчет динамической реакции подкрепленной оболочки большого размера. Расчетная схема содержала $\sim 10^4$ динамических степеней свободы. Результаты расчета сопоставлены с экспериментальными данными. На рис. 10.7, *б* приведена амплитудно-частотная характеристика для одной точки конструкции (по данным расчета и эксперимента). Некоторое расхождение объясняется неполным соответствием конструкции натурного объекта и его расчетной схемы, а также тем, что в расчете оболочка предполагалась свободной (незакрепленной), а в эксперименте оболочка закреплялась с помощью пружин.

При решении задач определения динамической реакции конструкции методом разложения по собственным формам очень важным является вопрос о количестве удерживаемых форм.

Авторами проведены исследования, по результатам которых сделан вывод о том, что собственная частота последней удерживаемой формы должна в 2—4 раза превышать частоту приложенной гармонической нагрузки (в зависимости от особенностей конструкции). Разработана методика, позволяющая для каждого конкретного объекта расчета определять число собственных форм, необходимых для получения достоверных результатов.

Расчет на собственные колебания суперэлементным методом сил или смешанным методом (задача динамического синтеза)

Алгоритмы и программное обеспечение подсистемы расчета на собственные колебания построены так, что позволяют решать задачу динамического синтеза подкрепленных оболочек и стержневых систем как суперэлементным смешанным методом, так и суперэлементным методом сил.

Приведем лишь результаты расчета незакрепленной балки, схема которой показана на рис. 10.8. Рассмотрим поперечные колебания балки постоянной жесткости с десятью сосредоточенными массами. Кратко опишем алгоритм для данного конкретного случая.

Расчленим балку на три суперэлемента: для первого и третьего суперэлементов выберем расчетную схему по методу сил, а для второго — по методу перемещений. Число динамических степеней свободы для суперэлементов, рассчитываемых методом сил, можно

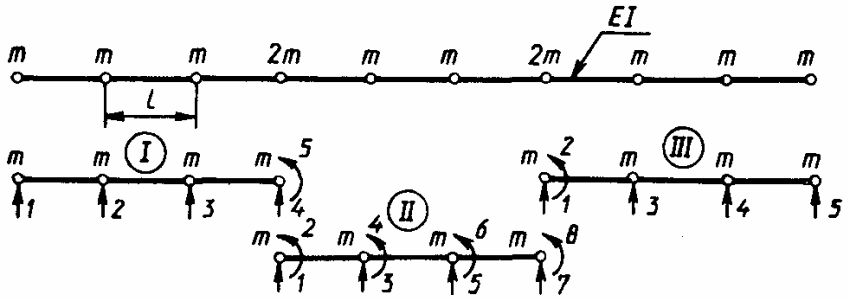


Рис. 10.8. Схема балки, рассчитанной смешанным суперэлементным методом

выбрать равным пяти, а для суперэлемента, рассчитываемого методом перемещений, — восьми. Выполнив статический расчет суперэлементов, построим матрицы податливости крайних суперэлементов, матрицу жесткости среднего суперэлемента, а также их матрицы масс. Решая сокращенную собственную проблему для суперэлементов, определим по одной упругой низшей форме собственных колебаний. Добавляя к ним формы колебаний как твердого тела и учитывая влияние отброшенных форм, по приведенному алгоритму проведем упругое сочленение суперэлементов. Результаты расчетов показали, что найденные суперэлементным методом формы собственных колебаний балки практически полностью совпали с «точным» решением для целой балки, полученным по методу перемещений. Ниже приведены частоты собственных колебаний, полученных без расчленения ($f_{точн}$) и с расчленением ($f_{с.э}$) на суперэлементы.

Номер частоты	$f_{точн}$, Гц	$f_{с.э}$, Гц	Ошибка, %
1	0,0342364	0,0341872	0,14
2	0,0878437	0,0876604	0,20
3	0,1992311	0,1992907	0,03

Результаты свидетельствуют о хорошей точности суперэлементного метода даже при использовании минимального числа форм собственных колебаний суперэлементов.

Определение несущей способности тонкостенных подкрепленных оболочек (физически нелинейный расчет)

В зоне разрушения тонкостенной конструкции силовые элементы работают в условиях физической нелинейности. В работах [4, 22] приведен метод расчета несущей способности таких конструкций.

Разрешающие уравнения метода записываются в виде уравнений в приращениях через касательные модули упругости элементов. Абсолютные величины напряжений и деформаций в любой момент времени определяются интегрированием с учетом исходного состояния конструкции. Метод основан на представлении реальных нелинейных диаграмм внешнего нагружения и диаграмм деформирования элементов конструкции в виде кусочно-линейных диаграмм (рис. 10.9).

Процесс нагружения конструкции разбивается на этапы так, чтобы на протяжении каждого этапа жесткостные характеристики и скорости нагружения оставались бы неизменными, что обеспечи-

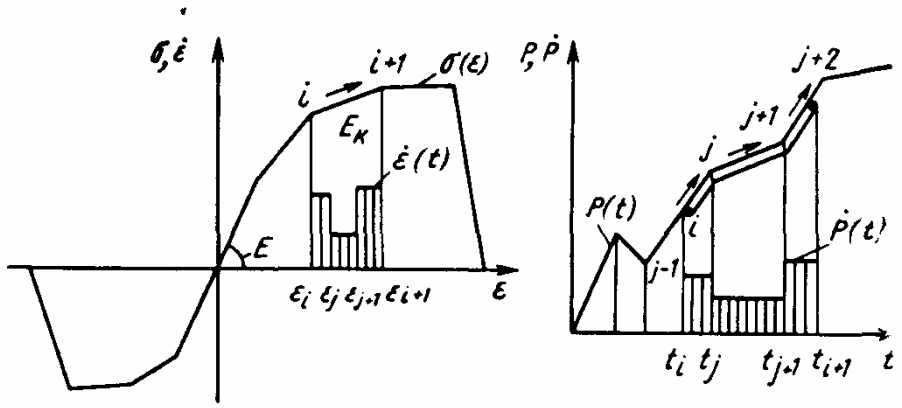


Рис. 10.9. Аппроксимация нелинейных диаграмм деформирования элементов конструкции и внешнего нагружения кусочно-линейными кривыми (E — модуль упругости; P — нагрузка; ε — относительное удлинение; σ — напряжение)

вает постоянную скорость изменения НДС элементов конструкции ($\dot{\sigma}$, $\dot{\varepsilon}$), соответствующую скоростям изменения нагружения на шаге расчета. Длительность шага Δt определяется условием достижения точки излома кусочно-линейной диаграммы одним из элементов конструкции или одним из компонентов нагружения. НДС конструкции на любом шаге ν определяется через НДС предыдущего шага $\nu-1$ по формулам

$$\varepsilon_\nu=\varepsilon_{\nu-1}+\dot{\varepsilon}_\nu\Delta t;\quad \sigma_\nu=\sigma_{\nu-1}+\dot{\sigma}_\nu\Delta t.$$

Применим метод конечных элементов в варианте метода сил [10, 35]. На первом шаге выполняется линейный расчет конструкции от внешней нагрузки; на каждом последующем шаге решается система уравнений значительно меньшего порядка; ее размер определяется числом силовых элементов, перешедших в зону нелинейного деформирования. Это достигается сведением расчета реальной упругопластической конструкции под воздействием внешней нагрузки к расчету некоторой упругой (фиктивной) конструкции, на которую действуют внешние нагрузки, и расчету от действия дополнительной системы сил, определяемой нелинейными деформациями. Считается, что в рассматриваемой фиктивной конструкции имеют место те же деформации элементов, что и в упругопластической конструкции.

В работе [22] получено выражение для скоростей изменения усилий в фиктивной конструкции (на любом шаге расчета) в виде

$$\dot{\mathbf{S}}_\text{ф}=\dot{\bar{\mathbf{S}}}+[\mathbf{E}_h-\mathbf{b}_D\mathbf{b}_{F,h}]\,\mathbf{A}_h^{-1}\dot{\bar{\mathbf{S}}}_h, \tag{10.1}$$

где $\mathbf{E}_h=\left\|\frac{\mathbf{I}_h}{0}\right\|$ — прямоугольная матрица; $\mathbf{I}_h$ — единичная матрица размером h; $\mathbf{b}_D=\mathbf{b}_1\mathbf{D}_{11}^{-1}$; $\mathbf{b}_F=\mathbf{b}_1^T\mathbf{f}$; $\mathbf{b}_{F,h}$, $\dot{\bar{\mathbf{S}}}_h$ — матрицы, полученные в результате выборки соответственно h столбцов из матрицы $\mathbf{b}_F$ и h строк из матрицы $\dot{\bar{\mathbf{S}}}$; h — число силовых элементов в упругопластической конструкции, имеющих пластические деформации

или потерявших устойчивость; $\mathbf{A}_h^{-1}$ — матрица размером h, обратная к матрице

$$\mathbf{A}_h = \mathbf{b}_{D,h}\mathbf{b}_{F,h} + \gamma_h;$$

$\mathbf{b}_{D,h}$ — матрица, полученная в результате выборки h строк из матрицы $\mathbf{b}_D$; $\gamma_h = \frac{\mathbf{E}_k}{\mathbf{E} - \mathbf{E}_k}$ — диагональная матрица параметров изменения касательных модулей силовых элементов; $\mathbf{E}$, $\mathbf{E}_k$ — матрицы-столбцы модулей упругости и касательных модулей линейных участков диаграмм деформирования.

На первом шаге $\nu = 1$ определяются скорости изменения усилий $\dot{\bar{\mathbf{S}}}$ в упругой конструкции от заданного внешнего нагружения. Скорости изменения НДС на этом шаге определяются из следующих выражений:

$$\dot{\varepsilon}_\nu = \frac{\dot{\bar{\mathbf{S}}}}{\mathbf{EF}}; \quad \sigma_\nu = \mathbf{E}\dot{\varepsilon}_\nu,$$

где $\mathbf{EF}$ — матрица-столбец жесткостей необъединенных элементов упругой конструкции.

На каждом последующем шаге изменяется касательный модуль только у одного опорного силового элемента, первым достигшего точки излома своей диаграммы деформирования.

Произведем коррекцию второго слагаемого выражения (10.1) с учетом того, что размеры матриц во втором слагаемом определяются числом силовых элементов, имеющих нелинейные деформации. Скорости изменения НДС в упругопластической конструкции в этом случае определяются скоростями изменения НДС фиктивной конструкции

$$\dot{\varepsilon}_\nu = \frac{\dot{\mathbf{S}}_\Phi}{\mathbf{EF}}, \quad \dot{\sigma}_\nu = \mathbf{E}_k\dot{\varepsilon}_\nu.$$

Для определения фактической несущей способности конструкции по предлагаемой методике разработан комплекс программ на базе СМПО и вычислительного комплекса СУМРАК [35].

Для иллюстрации возможностей метода рассматриваются результаты определения НДС и несущей способности оболочки, которая представляет собой круговую слабоконическую тонкостенную конструкцию, подкрепленную продольным и поперечным силовым набором (рис. 10.10). За расчетную схему работы конструкции принята схема конечно-элементного метода сил [10, 35]. В расчетном случае оболочка будет работать на изгиб в вертикальной плоскости при наличии незначительного крутящего момента. Рассмотрим простое нагружение, когда компоненты внешней нагрузки изменяются пропорционально и выражаются одним параметром — процентом расчетной нагрузки.

Поведение конструкции исследовалось в процессе всего нагружения при изменении напряженно-деформированного состояния за счет нелинейного деформирования отдельных элементов. Реальные

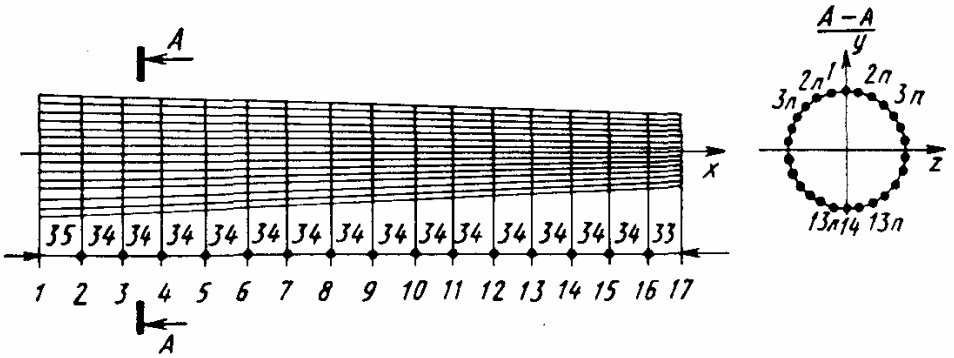


Рис. 10.10. Расчетная схема тонкостенной подкрепленной оболочки

диаграммы деформирования продольных силовых и оболочечных элементов аппроксимировались кусочно-линейными диаграммами (рис. 10.11). Было сделано допущение, что поперечные силовые элементы деформируются упруго в процессе всего нагружения.

В расчете для достижения нагрузки разрушения, составляющей 125,3% расчетной, потребовалось сделать 76 шагов. На рис. 10.12 показаны кривые распределения фиктивной несущей способности по длине балки, определенные как минимальные запасы прочности наиболее нагруженных силовых элементов сечений на разных уровнях нагрузки. Запас прочности определялся линейной экстраполяцией результатов расчета, как отношение предельно допустимой деформации элемента к достигнутой на данном уровне нагрузки. Тогда несущая способность сечения равна

$$P_{н.с} = \frac{\varepsilon_{пр}}{\varepsilon_{max}} P.$$

В приведенных результатах за предельно допустимую деформацию при работе элементов на сжатие принята максимальная расчетная деформация элементов конструкции $\varepsilon_{пр} = 1{,}0\%$ при нагрузке $P = 125{,}3\%\ P_{расч}$. Считается, что разрушение тонкостенной конструкции при изгибе происходит при достижении относительных деформаций элементов в сжатой зоне величины 0,8—1,0% [4].

Из рис. 10.12 видно, что оценка несущей способности по результатам расчета в упругой постановке дает сильно завышенные ре-

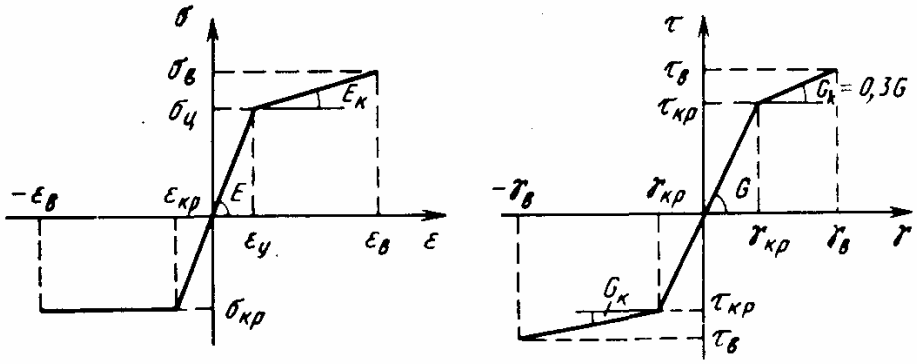


Рис. 10.11. Расчетные диаграммы напряжений, возникающих при деформировании элементов конструкции

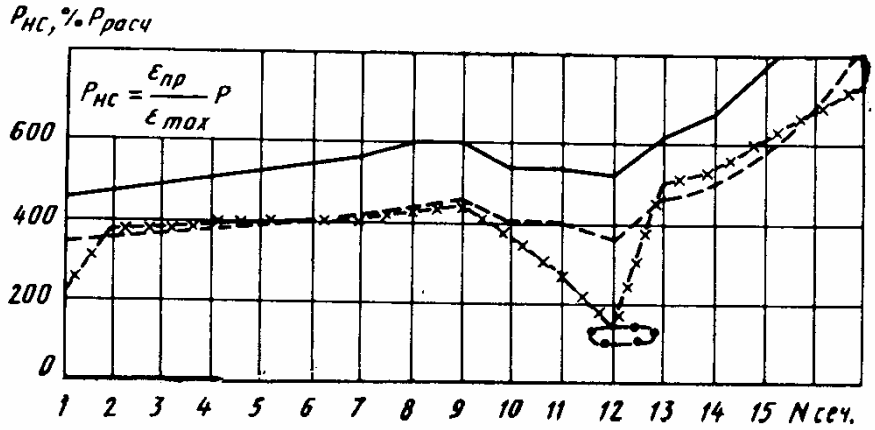


Рис. 10.12. Распределение несущей способности в расчетных сечениях по длине конструкции:

P_1=67% $P_{расч}$; — — — P_2=100% $P_{расч}$; — x — P_3=125,3% $P_{расч}$; — · — зона разрушения в эксперименте

зультаты, а расчет шаговым методом в физически нелинейной постановке позволяет достаточно точно определить фактическую прочность и слабые места конструкции (сечение № 12), а кроме этого выявить зоны локализации больших пластических деформаций элементов в слабых местах конструкции по сечениям 1 и 12.

Для сравнения на рис. 10.12 показана зона фактического разрушения тонкостенной подкрепленной оболочки при статических испытаниях.

Разрушение оболочки в эксперименте произошло при нагрузке, составляющей 126,3% расчетной, при этом разрушились продольные силовые элементы нижних панелей между сечениями 11—13, что хорошо согласуется с результатами расчета.

Таким образом, шаговый метод расчета тонкостенных конструкций по МКЭ в физически нелинейной постановке позволяет достаточно точно определять НДС конструкции с учетом изменения напряженно-деформированного состояния, выявлять слабые места конструкции в виде зон концентрации пластических деформаций и определять несущую способность конструкции в целом.

Пример использования метода сил с автоматизированным выбором основной системы и лишних неизвестных

Данное программное обеспечение для реализации конечноэлементного метода сил по сравнению с предыдущим является по идее более универсальным, оно предназначено для расчетов широкой разновидности конструкций при использовании различных конечных элементов. Алгоритм метода описан в разд. 1.2.

На рис. 10.13 показаны конструкции, на которых отлаживалось программное обеспечение с использованием СМПО, предназначенное для реализации указанного метода.

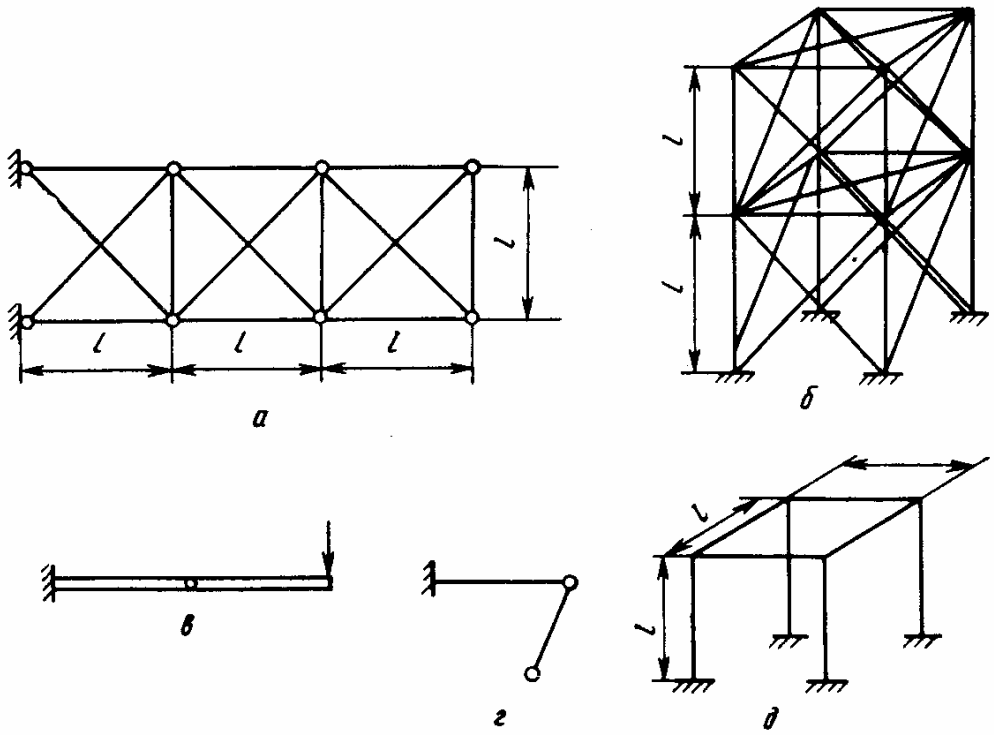


Рис. 10.13. Тестовые конструкции:

а — плоская ферма; *б* — пространственная ферма; *в* — балка; г — плоская рама; *д* — пространственная рама

10.2. ПРОГРАММНАЯ ПОДСИСТЕМА МЕТОДА ПЕРЕМЕЩЕНИЙ

Определение напряженно-деформированного состояния подкрепленных оболочек суперэлементным методом перемещений

Разработанное программное обеспечение, подсистемой которого является конечно-элементный метод перемещений (МП), испытано на разнообразных конструкциях. Подсистема МП является наиболее автоматизированной, снабжена средствами контроля, диагностики и визуализации. Приведем ряд простых примеров. На рис. 10.14 показана расчетная схема подкрепленной оболочки, которая разбита на 3 суперэлемента, и приведены эпюры напряжений в некоторых сечениях для случая, когда оболочка нагружена на свободном торце крутящим моментом $M = 10$ кН·м. Можно отметить практически приемлемое согласование результатов, полученных расчетным путем и экспериментально.

Расчет на собственные и вынужденные колебания машин, конструкций и сооружений

Алгоритм расчета на собственные колебания приведен в гл. 1. Остановимся здесь лишь на некоторых практических сторонах расчета на собственные колебания комбинированных конструкций. На рис. 10.15 показан алгоритм расчета.

Основные этапы решения в этом случае те же, что и при анализе напряженно-деформированного состояния. Дополнительно необ-

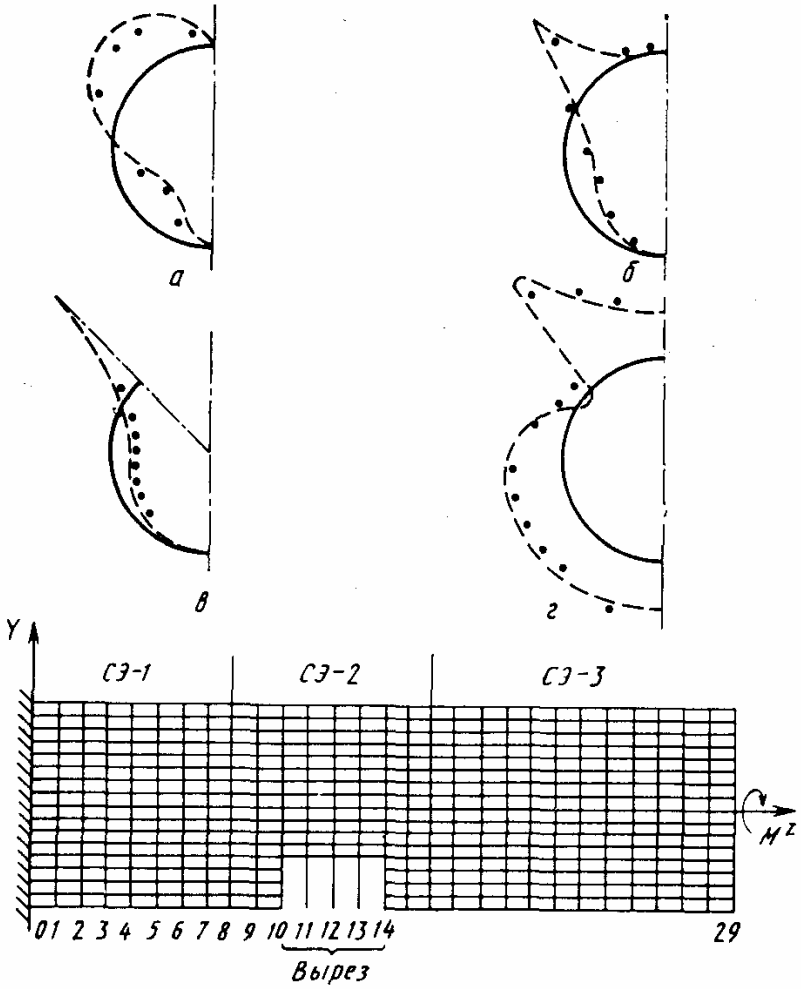


Рис. 10.14. Расчетная схема подкрепленной оболочки, нагруженной на свободном торце крутящим моментом $M=10$ кН·м:

а, б, в — распределение нормальных напряжений в продольных элементах между сечениями 9—10 и 10—11 соответственно; *г* — распределение касательных напряжений в обшивке между сечениями 9—10; — · — расчет; · — эксперимент

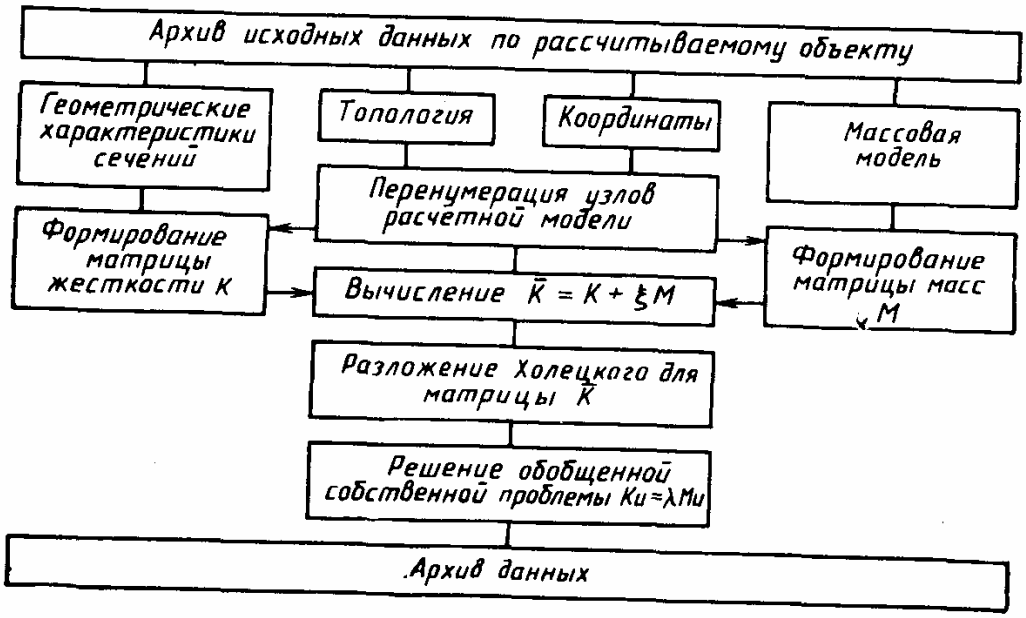


Рис. 10.15. Алгоритм расчета конструкции на собственные колебания методом перемещений

ходимо сформировать матрицу масс для полной конструкции с учетом сосредоточенных масс и решить обобщенную собственную проблему высокого порядка.

1. Формирование матрицы масс. Подсистема анализа собственных колебаний методом перемещений включает в себя построение как согласованной матрицы масс, так и матрицы сосредоточенных масс. Применение согласованной матрицы масс позволяет учесть инерционные свойства конечных элементов, что особенно важно при анализе гладких конструкций (при идеализации одним типом конечных элементов (КЭ)).

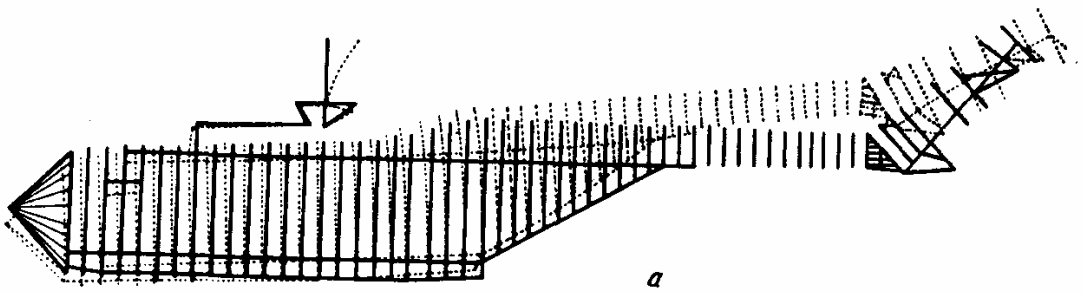
При анализе подкрепленных оболочек с крупными сосредоточенными массами влияние последних на собственные частоты и формы колебаний является более существенным по сравнению с инерционными свойствами КЭ (особенно для балочных тонов) и применение в этом случае матрицы сосредоточенных масс оправдано. Дополнительным преимуществом в этом случае является необходимость хранения лишь диагонали матрицы масс.

Для каждого типа конечных элементов написаны стандартные алгоритмы, позволяющие учитывать вклады от отдельных КЭ в полную матрицу масс. Помимо собственной массы оболочки необходимо включать в матрицу масс сосредоточенные массы. Сосредоточенные массы подразделяем на две группы: массы, совпадающие с узлами расчетной модели, и массы, лежащие вне ее узлов (вынесенные массы).

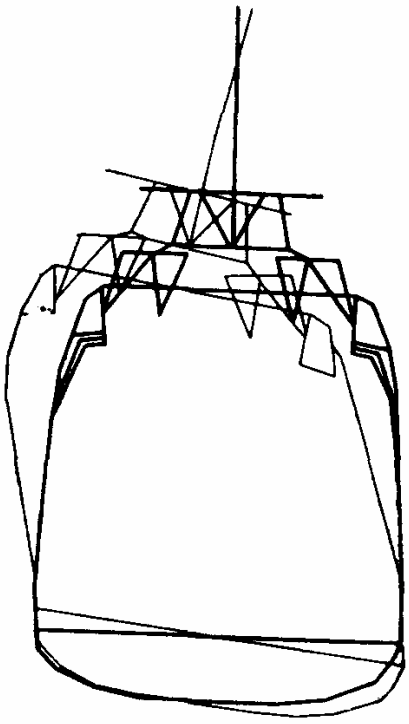
Массы первой группы дают вклад в матрицу масс по всем степеням свободы, соответствующим узлу, в котором они находятся. Массы второй группы включаются в полную матрицу масс иначе — расчетная модель объекта увеличивается путем включения в нее дополнительных узлов, соответствующих вынесенным массам. Дополнительные узлы присоединяются к существующим при помощи «абсолютно жестких» балок.

2. Решение обобщенной собственной проблемы. Обобщенная собственная проблема для полной конструкции имеет для реальных конструкций высокий порядок. Для ее решения используется метод одновременных итераций (см. гл. 1). При анализе свободной конструкции матрица жесткости является вырожденной и перед применением метода необходимо применить сдвиг, т. е. вычислить матрицу $\bar{\mathbf{K}}=\mathbf{K}+\xi\mathbf{M}$, где величина сдвига принимается порядка $\xi=\omega^2$, для устранения вырожденности матрицы жесткости. При этом собственные формы не изменяются, а собственные числа увеличиваются на величину ξ. В случае закрепленной конструкции величина ξ равна 0.

По описанной выше схеме можно рассчитывать реальные конструкции с практически любым числом узлов. Так были рассчитаны различные подкрепленные оболочки. В расчетной схеме были учтены все основные особенности конструкции и подробная массовая модель (1606 сосредоточенных масс). Решалась проблема собственных значений для динамической матрицы размером $10^4\times10^4$. Некоторые из найденных форм колебаний приведены на рис. 10.16.



а



б

Рис. 10.16. Формы колебаний № 11 (а) и № 12 (б)

Согласование расчетных и экспериментальных данных практически приемлемое. Таким же образом конструкция рассчитывалась на вынужденные колебания различными методами (точным и приближенными: разложением по собственным формам и по векторам Ланцоша), которые дали практически одинаковые результаты.

Расчет на собственные колебания и устойчивость суперэлементным методом перемещений (задача динамического синтеза)

В разд. 1.3 изложен алгоритм суперэлементного метода перемещений для расчета конструкций на собственные колебания и устойчивость. Для проверки эффективности приведенного алгоритма модельная ребристая оболочка (рис. 10.17) была рассчитана на собственные колебания с применением подконструкций, и результаты сравнивались с результатами расчета нерасчлененной оболочки.

Оболочка содержит шесть поперечных и шесть продольных силовых элементов. При моделировании оболочки применялись конечные элементы плоского бруса (для идеализации поперечных элементов), элементы фермы (для продольных элементов) и прямоугольные мембранные элементы (обшивка). Общее число степеней свободы при расчете полной конструкции равнялось 144. Назовем параметры составляющих оболочку элементов: площадь поперечного сечения продольных элементов $A_s=5\cdot10^{-5}$ м^2; площадь и момент инерции поперечного сечения поперечных элементов $A=6\cdot10^{-4}$ м^2, $J=18\cdot10^{-4}$; толщина обшивки $t=3\cdot10^{-3}$ м. Физические характеристики материала оболочки: модуль Юнга $E=7{,}2\cdot10^{10}$ Н/м^2; коэффициент Пуассона $\mu=0{,}333$; плотность $\rho=2{,}8\cdot10^{-3}$ кг/м^3. При расчете суперэлементным методом оболочка расчленялась на две подконст-

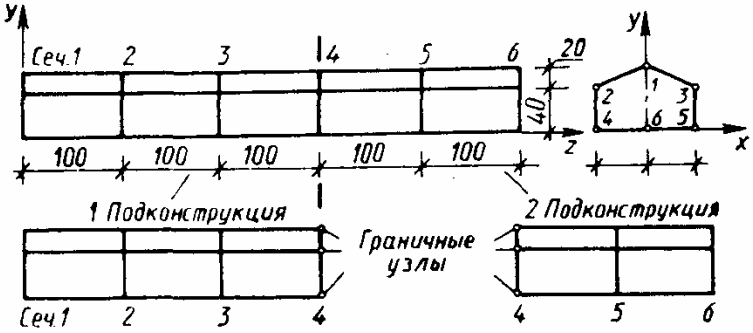


Рис. 10.17. Расчетная схема оболочки, рассчитанная суперэлементным методом перемещений

рукции по плоскости четвертого сечения. При этом жесткость 4-го поперечного элемента делилась пополам. Число степеней свободы первой подконструкции равно 96, второй — 72.

В результате расчета были определены 18 частот и форм колебаний для нерасчлененной конструкции и для конструкции, расчлененной на суперэлементы. Результаты, полученные обоими методами, совпали.

Ниже приведены частоты колебаний в Гц для 16-ти форм:

Номер формы	1...6	7	8	9	10	11	12	13	14	15	16
Частота, Гц	. . . 0	2,35	2,57	2,59	2,62	2,64	2,75	3,05	3,15	3,83	4,04

10.3. ПРИМЕНЕНИЕ СМПО К РАСЧЕТУ РАЗЛИЧНЫХ КОНСТРУКЦИЙ

Применение СМПО к расчету конструкции автомобиля

Метод конечных элементов стал к настоящему времени основным средством оценки прочностных и динамических характеристик конструкции автомобиля на стадии проектирования.

Сфера применения матричных методов и МКЭ для решения задач динамики и прочности автомобиля весьма широка; она включает в себя определение НДС несущей системы, ходовой части и деталей двигателя, оптимизацию масс, определение частот и форм собственных колебаний, амплитудно-частотных характеристик, численное моделирование деформирования автомобилей при авариях, вопросы расчетной оценки ресурса важнейших узлов и деталей и т. д.

Большой объем и качественное разнообразие задач строительной механики автомобиля требуют создания развитых специализированных программных средств МКЭ либо адекватной адаптации пакетов программ общего назначения, к которым относится и ППП СМПО и СУМРАК. Путь адаптации имеющихся программных систем представляется более целесообразным в силу наличия в них проблемно-независимых средств, обеспечивающих управление процессом счета, выполнение сервисных функций и алгебраических операций. Для указанного ППП такая адаптация сводится к введению в библиотеку необходимых конечных элементов, а также некоторых проблемно-ориентированных матричных процедур, легко реализуемых средствами СМПО; отдельные алгоритмы, для которых матричный язык не является естественным, могут быть написаны на языке фортран-IV и подключены к ППП СУМРАК.

Для иллюстрации применения СМПО к расчету автомобильных конструкций рассмотрим решение МКЭ задачи определения общего напряженно-деформированного состояния автомобиля-самосвала КамАЗ с задней разгрузкой.

В литературе описаны различные расчетные модели самосвальных платформ: 1) тонкостенный стержень с диафрагмой (передний борт) и подкреплениями (обвязка бортов); 2) система последовательно рассчитываемых лонжеронов основания и пластин днища и бортов; 3) тонкостенная подкрепленная оболочка. При выполнении настоящей работы принята последняя модель как наиболее адекватная.

Подкрепления моделировались с помощью прямолинейных пространственных конечных элементов балки. В качестве КЭ оболочки был принят плоский треугольный элемент, изгибной составляющей которого является КЭ *DKT* (Dicrete Kirchoff Triangle), мембранной составляющей — широко применяемый элемент с постоянной деформацией. Узловыми степенями свободы являются здесь 6 минимально допустимых «инженерных» степеней свободы — 3 перемещения и 3 угла поворота. Выбор КЭ *DKT* в качестве изгибной составляющей обусловлен быстрой сходимостью результатов по перемещениям и напряжениям, а также его другими важными для практического применения свойствами: слабой зависимостью от ориентации и отношений длин сторон КЭ и возможностью вычисления матрицы жесткости по явным формулам без применения численного интегрирования. Обширные сопоставительные исследования [42] показали, что КЭ *DKT* является предпочтительным среди треугольных КЭ с тремя узлами и девятью изгибными степенями свободы.

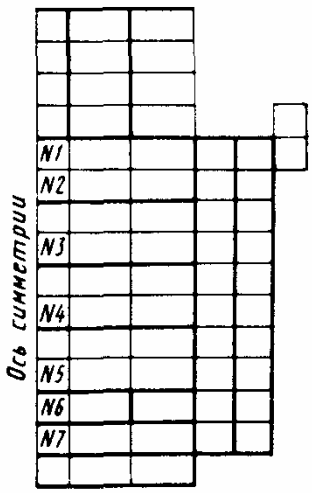
Отметим также, что применение сложных КЭ с большим числом степеней свободы для расчета нерегулярных подкрепленных оболочек рассматриваемого класса нерационально, поскольку частое расположение подкреплений не позволяет моделировать конструкцию малым числом КЭ, а вычислительные затраты в этом случае существенно выше, чем для простых КЭ.

Соотношения для КЭ оболочки *DKT* запрограммированы на языке фортран-IV; КЭ *DKT* включен в библиотеку КЭ ППП СУМРАК.

Выполнению расчета для реальной конструкции — платформы самосвала КамАЗ предшествовала серия сопоставительных расчетов конструкции-прототипа, близкой к реальной по геометрическим и жесткостным характеристикам (рис. 10.18), с использованием КЭ оболочки двух различных типов и при различных сетках. Конструкция-прототип рассматривалась шарнирно закрепленной в трех точках: точке упора гидроцилиндра подъемника и двух точках на оси опрокидывания. Такое закрепление соответствует моменту начала подъема платформы и определяет один из основных расчетных случаев. Рассматривались два случая нагружения: 1) гидростатическое давление; 2) нагружение сыпучим грузом (величина нагрузки обратно пропорциональна расстоянию от центра ее прило-

Рис. 10.18. Развертка расчетной схемы платформы-прототипа (симметричная часть):

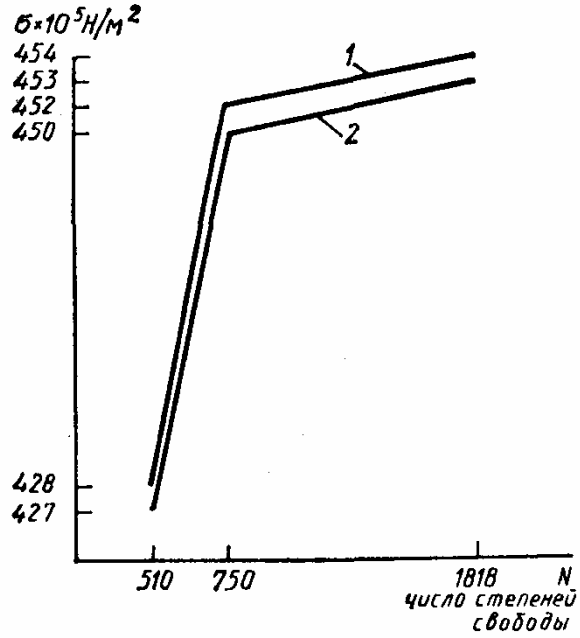
——— ребра жесткости



жения). Расчет проводился для части конструкции, симметричной относительно ее продольной оси.

Оболочка конструкции-прототипа моделировалась последовательно с помощью двух типов элементов: 1) прямоугольного гибридного КЭ [35] (его однослойного варианта), построенного на основе обобщенного функционала Кастильяно, и классической теории оболочек Кирхгофа — Лява; 2) треугольного КЭ *DKT* в перемещениях, построенного на основе функционала Лагранжа и теории оболочек типа Тимошенко. Расчеты прототипа проводились на трех сетках КЭ для обоих типов КЭ оболочки (для каждого вида нагружения выполнено 6 расчетов).

На рис. 10.19 показан график сходимости значений напряжений в одном из балочных КЭ подкрепления № 3 днища в зависимости от типа КЭ оболочки и числа степеней свободы модели (случай гидростатического давления); аналогичная картина сходимости наблюдается и для других подкреплений.



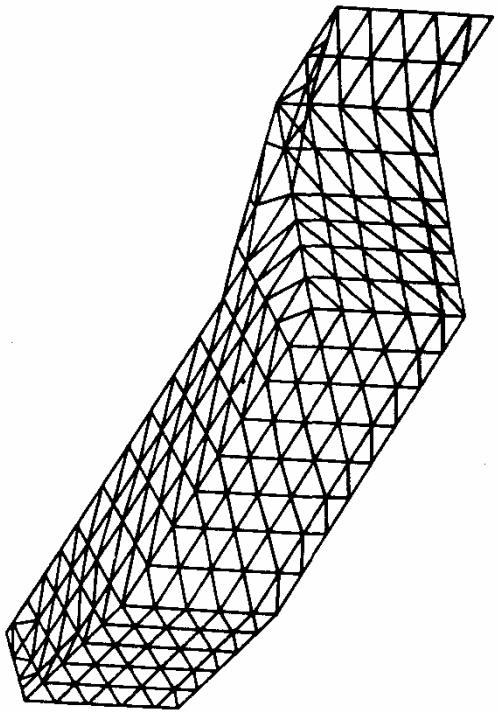


Рис. 10.19. Зависимости напряжений в одном из КЭ ребра жесткости № 3 днища платформы-прототипа от числа степеней свободы модели и типа КЭ оболочки: *1* — КЭ DKT; *2* — гибридный прямоугольный КЭ

Рис. 10.20. Сетка треугольных КЭ для расчета конструкции платформы автомобиля-самосвала КамАЗ-5511

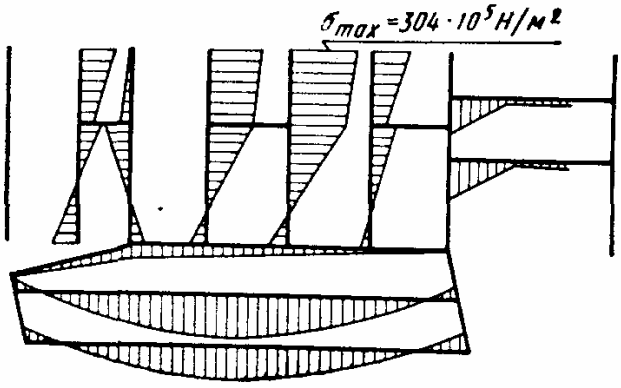


Рис. 10.21. Эпюры нормальных напряжений в подкрепляющих элементах днища и бортов платформы автомобиля-самосвала КамАЗ-5511

Анализ результатов расчета прототипа позволяет сделать вывод о сходимости решений в рамках принятой рсчетной модели и выбрать сетку КЭ для оценки общего НДС реальной конструкции. Отметим, что аналогичный опыт подбора расчетной сетки для конструкции кабины грузового автомобиля описан в статье [60].

Сетка КЭ реальной конструкции платформы автомобиля-самосвала КамАЗ (рис. 10.20) содержала 192 узла, 330 треугольных КЭ оболочки, 114 пространственных КЭ балки; порядок разрешающей системы уравнений равен 1152.

Подготовка данных по геометрии, топологии, жесткостям КЭ оболочки и балок, нагружению, а также оптимизация первоначальной нумерации узлов сетки были проведены с использованием соответствующего программного обеспечения ППП СУМРАК. Построенная сетка визуально контролировалась с применением графической подсистемы пакета.

Днище и борта платформы представляют собой стальные листы толщиной 8 и 4 мм соответственно; подкрепления — балки швеллерного сечения. В частности, продольная балка бокового борта имеет следующие размеры поперечного сечения: высота стенки — 145 мм, ширина полок — 65 мм, толщина — 3 мм.

Величина давления на днище платформы считалась равной высоте столба перевозимого груза в соответствующей точке, умноженной на его плотность; давление на борта платформы вычислялось по соответствующим формулам механики грунтов, применяемым для расчета давления грунта на подпорные стенки.

Проведенный расчет следует рассматривать как сравнительно простой пример использования СМПО в анализе напряженно-деформированного состояния автомобильных конструкций. Описанное программное обеспечение позволяет также решать задачи определения НДС, собственных и вынужденных колебаний, связанные с рассмотрением автомобиля как единого целого (используя суперэлементный вариант МКЭ).

На рис. 10.21 показаны эпюры нормальных напряжений, возникающих в подкрепляющих элементах платформы-самосвала КамАЗ-55П.

Расчет различных строительных конструкций

Программное обеспечение располагает необходимыми средствами для прочностных расчетов широкого класса конструкций, в том числе и строительных. Приведем несколько примеров расчета.

1. Расчет мостового перегружателя пролетом 76,2 м и грузоподъемностью 30 т. Расчетная схема крана показана на рис. 10.22. Число узлов расчетной модели составило 618. Конструкция моделировалась 1618 балочными конечными элементами. Произведенный расчет позволил выявить наиболее нагруженные элементы металлоконструкций перегружателя.

2. Расчет облегченных арочных конструкций сельскохозяйственных сооружений. Производился расчет полной арки пролетом 16,29 м, высотой 8 м (расчетная схема показана на рис. 10.23) для определения усилий в элементах; при этом пояса моделировались балочными элементами, а раскосы — элементами фермы. Далее производился расчет типового узла нижнего пояса на действие вычисленных усилий (расчетная модель узла показана на рис. 10.24).

Для идеализации узла были использованы треугольные элементы оболочки *DKT* (см. разд. 10.9). Число узлов расчетной модели равно 104, число элементов — 154.

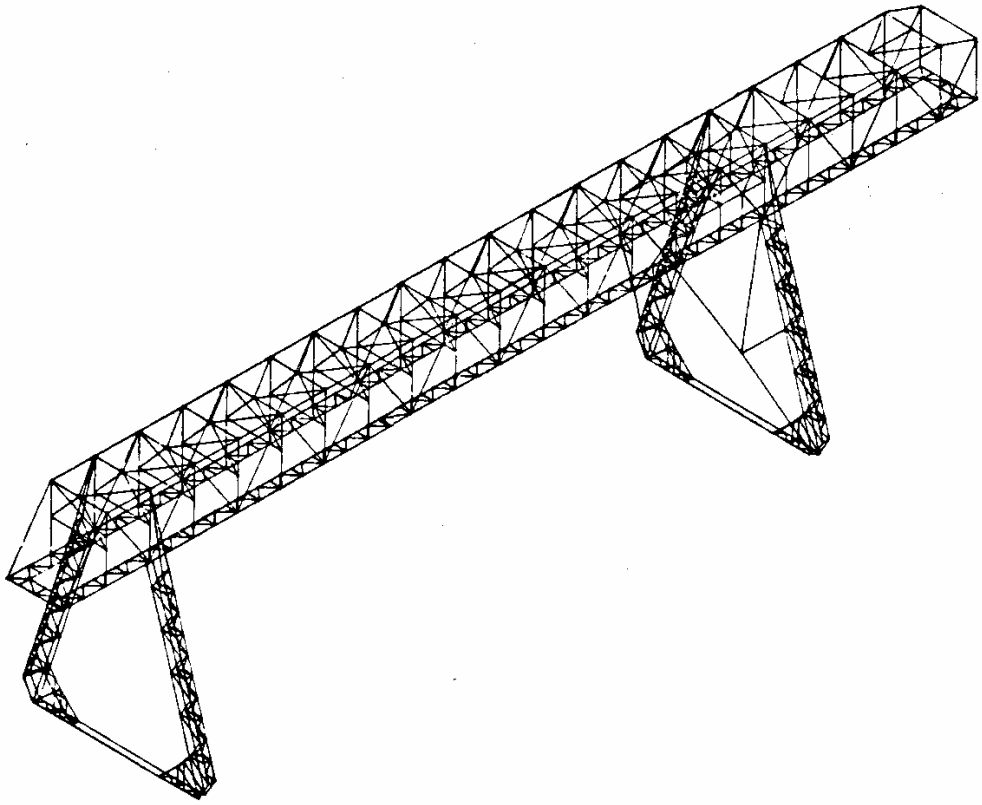


Рис. 10.22. **Расчетная схема мостового крана-перегружателя**

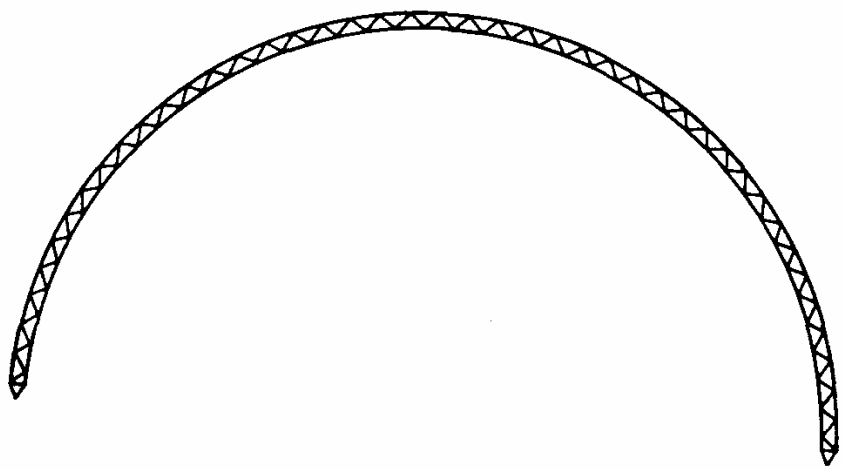


Рис. 10.23. Расчетная схема арки

Расчет пространственной конструкции из условий улучшения ее виброизоляции

В работе [31] решена задача виброзащиты пространственной фермы, на которой установлены оптические приборы высокой точности. Известно, что податливость конструкции влияет на поведение приборов с большим оптическим плечом.

Объектом виброзащиты явилась пространственная ферма длиной 15 м, высотой 0,42 м и шириной 0,72 м с установленными на ней оптическими приборами; число виброизолирующих опор — 22 — соответствует числу узлов нижних поясов фермы. Ферма подвергалась внешнему возбуждению в виде вертикального смещения осно-

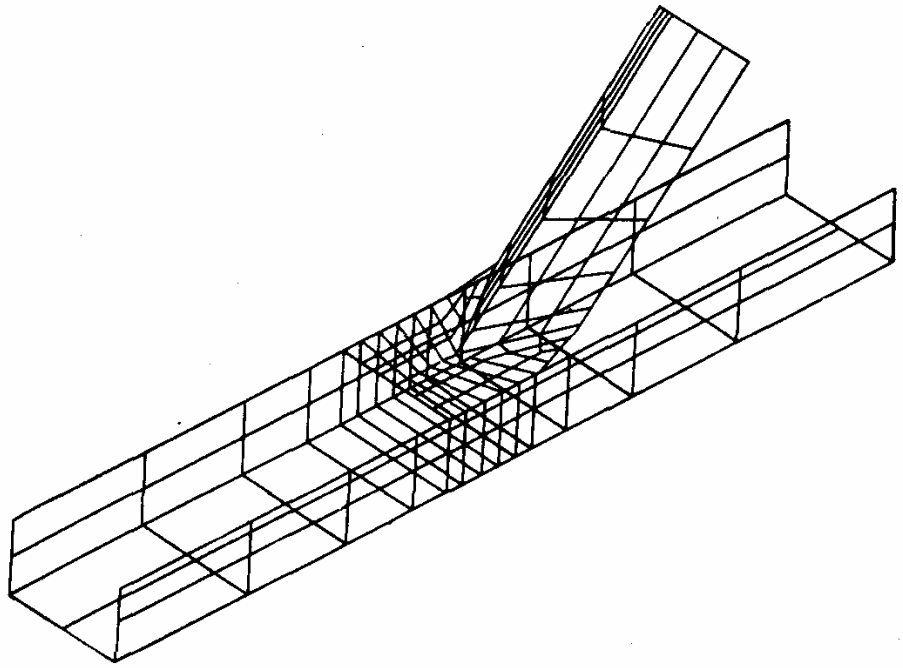


Рис. 10.24. Расчетная схема типового узла нижнего пояса арки

вания по гармоническому закону

$$\mathbf{X}_0 = \overline{\mathbf{X}} \sin \omega t. \tag{10.2}$$

Рассмотрим следующее уравнение колебаний системы с учетом демпфирования:

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{C}\dot{\mathbf{X}} + \mathbf{K}\mathbf{X} = 0, \tag{10.3}$$

где $\mathbf{M}$ — матрица масс; $\mathbf{K}$ — матрица жесткости; $\mathbf{C}$ — матрица коэффициентов вязкого сопротивления демпферов; $\mathbf{X}$ — вектор узловых смещений. Так как смещения основания заданы, то уравнение (10.3) примет вид

$$\mathbf{M}_1\ddot{\mathbf{X}}_1\mathbf{C}_1\mathbf{X}_1 + \mathbf{K}_1\mathbf{X}_1 = \mathbf{M}_0\ddot{\mathbf{X}}_0 + \mathbf{C}_0\dot{\mathbf{X}}_0 + \mathbf{K}_0\mathbf{X}_0,$$

где $\mathbf{M}_1$ и $\mathbf{K}_1$ — квадратные матрицы порядка $n + N_0$ или $N_{пр} + 2n + N_0$ при учете приведенных масс виброизоляторов; n, $N_{пр}$ и N_0 соответственно число виброизоляторов, узлов приведенной модели фундамента и частей оборудования, расположенного на фундаменте.

Учитывая (10.2) и то, что $\mathbf{X} = \mathbf{A} \sin \omega t + \mathbf{B} \cos \omega t$, получим систему уравнений относительно неизвестных векторов $\mathbf{A}$ и $\mathbf{B}$

$$\begin{gathered} (\mathbf{K}_1 - \omega^2 \mathbf{M}_1)\mathbf{A} - \omega \mathbf{C}\mathbf{B} = (\mathbf{K}_0 - \omega^2 \mathbf{M}_0)\overline{\mathbf{X}}; \\ \omega \mathbf{C}\mathbf{A} + (\mathbf{K}_1 - \omega^2 \mathbf{M}_1)\mathbf{B} = \omega \mathbf{C}_0\mathbf{X}_0. \end{gathered} \tag{10.4}$$

Решение задачи проводилось на ППП СУМРАК; находились пути уменьшения амплитуд изгибных колебаний рассматриваемой системы.

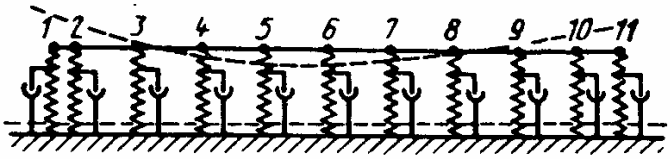
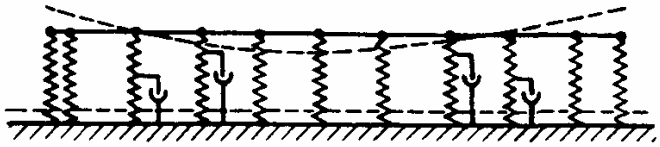
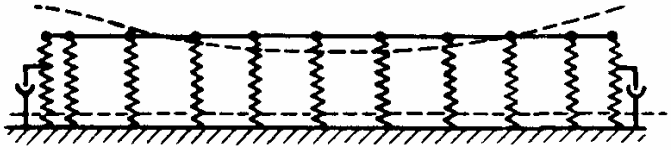
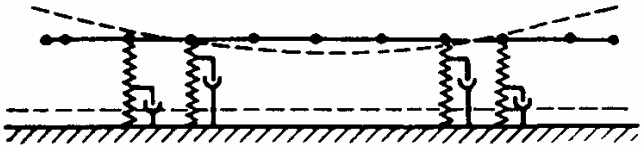
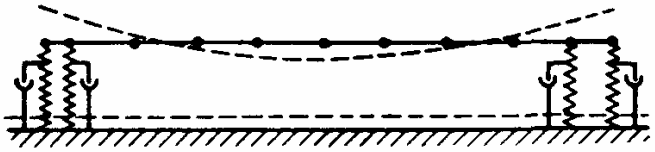
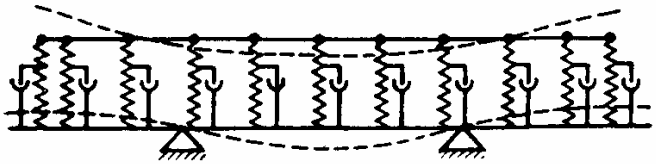
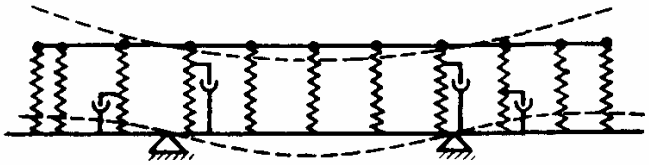
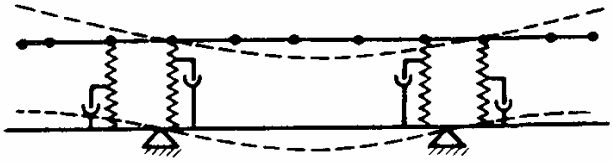
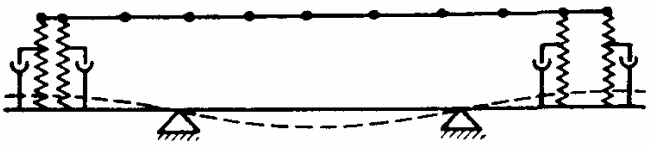
За критерий оценки принималась разность $\Delta\mathbf{X}$ между наибольшим и наименьшим значениями перемещения точек фундамента или масс расположенного на нем оборудования при наиболее неблагоприятных с точки зрения работы точного оборудования изгибных формах колебаний.

В табл. 10.1 приведены результаты оптимизации размещения упругих и демпфирующих элементов по длине фундамента по принятому критерию для первой изгибной формы колебаний. Пунктиром на схемах показана первая балочная форма колебаний и форма колебаний перекрытия.

Первая и шестая схемы характеризуют начальные этапы поиска. Минимальное относительное смещение точек фундамента обеспечивается при расположении упругих элементов в соответствии с распределением масс виброизолируемого объекта, а демпферов — в узловых точках собственных форм колебаний фундамента так, чтобы равнодействующая их неупругих реакций проходила через узлы формы (схемы 2, 4, 7, 8). Неэффективному размещению демпферов соответствуют схемы 3, 5 и 9.

В схемах 4 и 8 упругие и неупругие характеристики виброизоляторов (например, резиновых) неразделимы.

Таблица 10.1

Схема расположения упругих и демпфирующих элементов виброизоляции протяженного фундамента	ΔX, мм при C=2100 кг/с
	0,1174
	0,0485
	0,2987
	0,0643
	0,2760
	0,0228
	0,0096
	0,0115
	0,0758

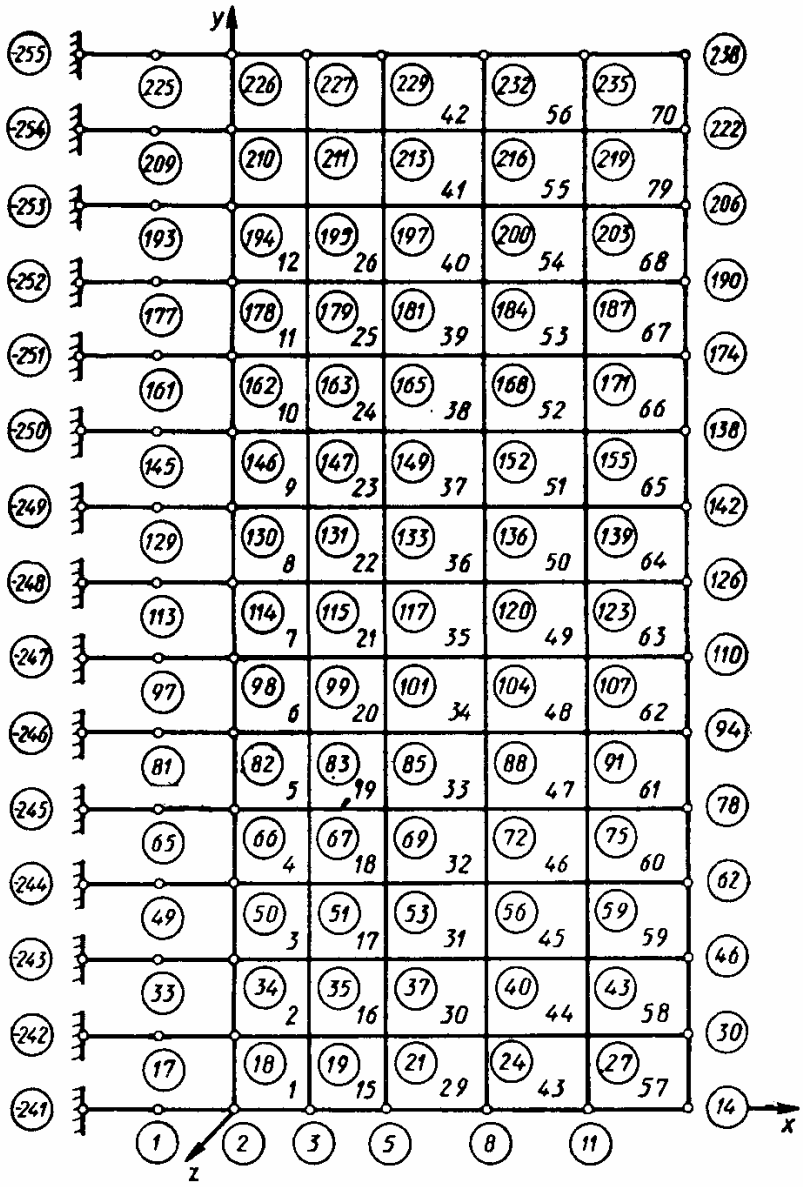


Рис. 10.25. Расчетная схема левой створки шлюзовых ворот (числа в кружках — номера узлов; без кружков — номера элементов)

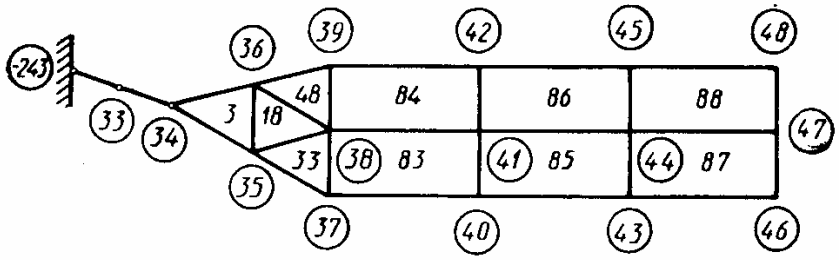


Рис. 10.26. Расчетная схема одного из ригелей

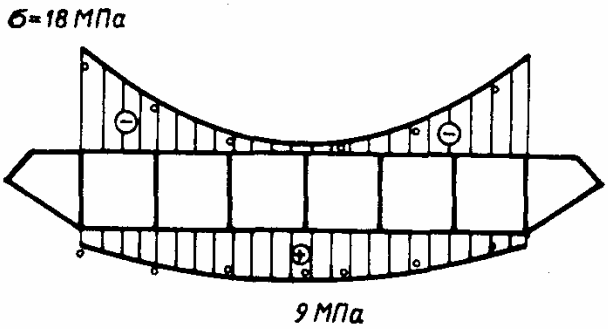


Рис. 10.27. Распределение продольных напряжений по сечению третьего ригеля.

——— МКЭ; • — эксперимент

Расчет шлюзовых ворот

Створка шлюзовых ворот представляет собой металлическую сварную конструкцию, состоящую из обшивки и подкрепляющего перекрестного набора ребер — ригелей и диафрагм. Расчетная схема ворот и одного из ригелей показана на рис. 10.25 и 10.26. В силу симметрии рассматривается половина створки. В расчете использованы следующие конечные элементы: 294 прямоугольных гибридных, 60 плоских треугольных, 102 пространственных стержневых; число степеней свободы расчетной схемы 1440. На рис. 10.27 показано распределение напряжений растяжения — сжатия по сечению третьего ригеля, найденное расчетным и экспериментальным путем; наблюдается удовлетворительное согласование результатов.

10.4. СМЕШАННЫЙ МЕТОД. СОПОСТАВЛЕНИЕ МЕТОДОВ

Определение напряженно-деформированного состояния комбинированных конструкций

Смешанный суперэлементный метод расчета комбинированных конструкций, описанный в работах [9, 35] и нашедший свое дальнейшее теоретическое обобщение в данной книге, реализован с помощью СМПО. Целесообразность использования метода проявляется в случае, когда отдельные части сложной большеразмерной конструкции могут быть рассчитаны методом сил (что даст возможность в указанном месте на более грубой расчетной сетке получить достоверную картину напряжений).

В работе [9] приведены результаты расчета подкрепленной оболочки, а также некоторые эпюры расчетных напряжений и данные эксперимента, которые согласуются. Рассчитывался также еще ряд сложных конструкций.

Правильность работы алгоритма проверялась с помощью ряда специальных тестов и показано, что программное обеспечение СМПО позволяет получать результаты, близкие к тем, которые могут быть получены однородными методами (или только МП, или только МС).

Расчет на собственные колебания суперэлементным смешанным методом

В книге [35] дан алгоритм построения матрицы податливости конструкции суперэлементным смешанным методом. Здесь приводится пример использования указанного алгоритма для расчета подкрепленной оболочки. Взята расчетная схема конструкции, состоящая из пяти суперэлементов (три из которых рассчитывались методом сил); размер динамической матрицы 394×394. Имеет место удовлетворительное согласование расчетных и экспериментальных данных для низших форм колебаний (рис. 10.28).

Сопоставительный расчет тонкостенных подкрепленных оболочек конечно-элементными методами сил и перемещений

Представляется целесообразным производить прочностной расчет одной и той же конструкции параллельно двумя методами. Программное обеспечение представляет пользователю такую возможность. Методика реализована для тонкостенных подкрепленных оболочек, для которых напряженное состояние удовлетворительно определяется путем использования простейших конечных элементов: стержня, балки, оболочки, работающей на чистый сдвиг, в виде плоского закрученного четырехугольника.

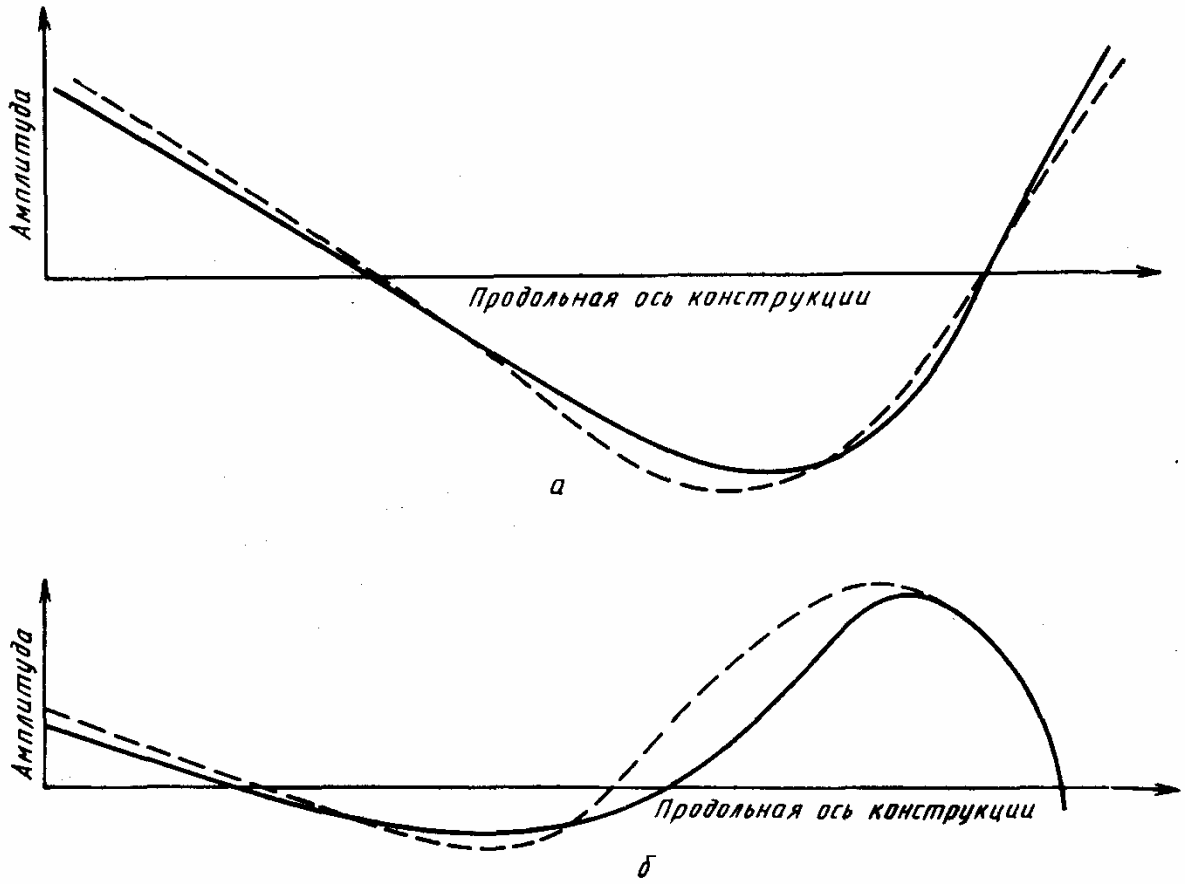


Рис. 10.28. Собственная форма 1-го (а) и 2-го (б) тонов

——— эксперимент; — — — расчет

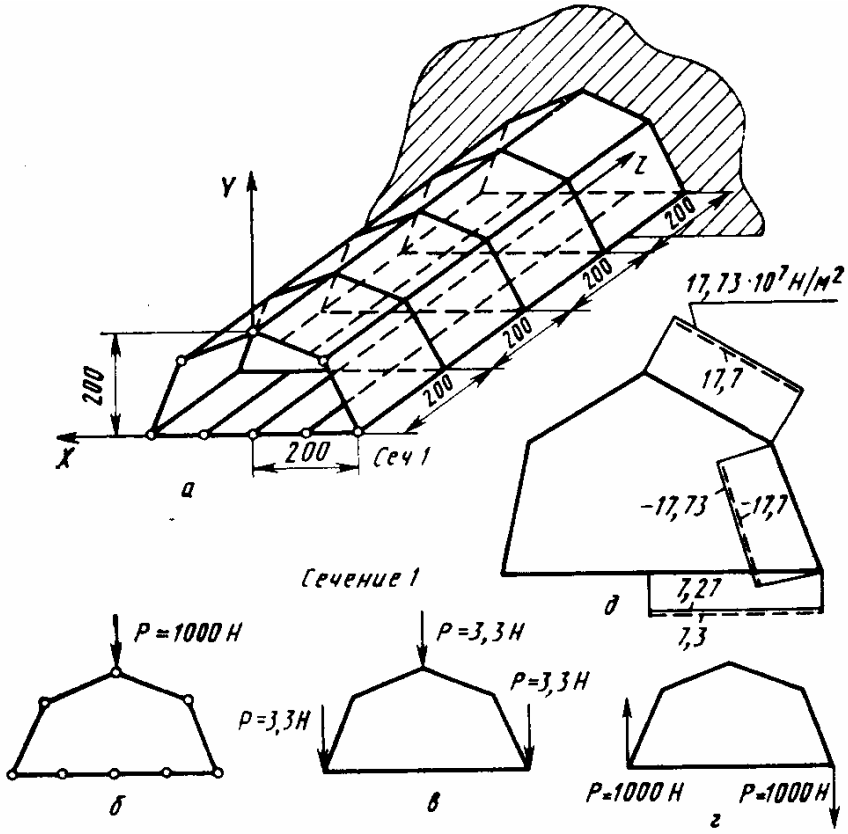


Рис. 10.29. Цилиндрическая тонкостенная подкрепленная оболочка:

а — расчетная схема; *б, в, г* — соответственно расчетные случаи нагружения; *д* — эпюра касательных напряжений в обшивке между сечениями 1, 2 для расчетного случая *б* (эпюра — антисимметрична); ——— метод перемещений; — — — метод сил

Ниже рассмотрены простые примеры, на которых легко проследить эффективность сопоставительных расчетов.

1. Определение НДС. На рис. 10.29 показана расчетная схема жестко закрепленной на 5-ом сечении оболочки, рассчитанной при трех видах нагружения. Исходные расчетные данные оболочки: $t=0{,}05$ см — толщина обшивки; $B=2$ см^2 — площадь сечения продольных элементов; $A=4$ см^2 — площадь сечения поперечных элементов; $E=2{,}1\cdot10^{11}$ Н/см^2 — модуль упругости материала конструкции; $\nu=0{,}3$ — коэффициент Пуассона; $W=2{,}67$ см^3 — момент сопротивления поперечного элемента.

Одна из эпюр напряжений по результатам расчета различными методами дана на рис. 10.29, д; такое же хорошее совпадение результатов наблюдается для всей конструкции.

Производилось также сопоставление НДС, полученного для подкрепленной оболочки различными методами [35]. Расчетная сетка в методе сил грубая, в методе перемещений (см. разд. 10.6) — мелкая (число конечных и конструктивных элементов здесь совпадает). Тем не менее в обоих случаях согласование результатов расчета и эксперимента одинаково хорошее. Это свидетельствует о преимуществе метода сил перед методом перемещений в плане получения более достоверного напряженного состояния.

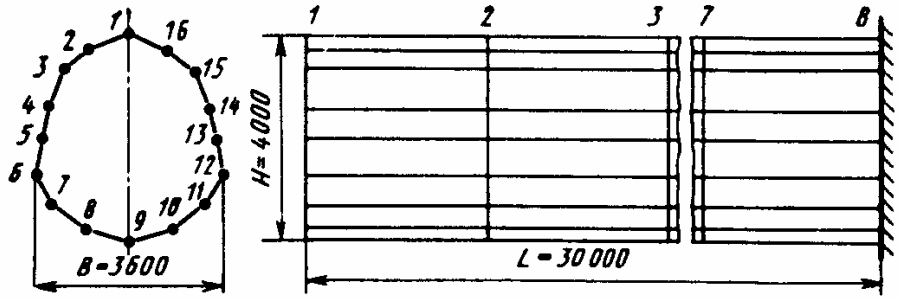


Рис. 10.30. Расчетная схема подкрепленной оболочки для сопоставительного расчета конечноэлементными методами сил и перемещений

2. Расчет на собственные колебания. Алгоритм расчета методом сил описан в работе [35], алгоритм расчета на собственные колебания методом перемещений приведен в гл. 1.

На рис. 10.30 показана оболочка, имеющая 16 продольных и 8 поперечных силовых элементов. Частоты собственных колебаний в герцах, полученные в результате расчета различными методами, приведены ниже:

Номер частоты	1	2	3	4	5
Метод сил	$4{,}1\cdot10^{-2}$	$3{,}7\cdot10^{-2}$	$1{,}4\cdot10^{-3}$	$1{,}3\cdot10^{-3}$	$1{,}2\cdot10^{-3}$
Метод перемещений	$4{,}0\cdot10^{-2}$	$3{,}9\cdot10^{-2}$	$1{,}4\cdot10^{-3}$	$1{,}3\cdot10^{-3}$	$1{,}2\cdot10^{-3}$
Номер частоты	6	7	8	9	10
Метод сил	$5{,}0\cdot10^{-4}$	$4{,}6\cdot10^{-4}$	$4{,}6\cdot10^{-4}$	$4{,}4\cdot10^{-5}$	$4{,}0\cdot10^{-5}$
Метод перемещений	$5{,}0\cdot10^{-4}$	$4{,}6\cdot10^{-4}$	$4{,}5\cdot10^{-4}$	$4{,}3\cdot10^{-5}$	$4{,}0\cdot10^{-5}$

10.5. ПРИМЕНЕНИЕ СМПО ДЛЯ РЕШЕНИЯ НЕКОТОРЫХ ТЕХНИЧЕСКИХ ЗАДАЧ В МАТРИЧНОЙ ФОРМЕ

Расчет висячих мостов

Для определения напряженно-деформированного состояния получена следующая система нелинейных уравнений, прогибов балки, равновесия тросов, неразрывности перемещений тросовых ферм и балки и уравнение для определения распора в тросовых фермах.

$$\eta = \mathbf{L}_{10}\mathbf{T}; \tag{10.5}$$

$$\mathbf{P}_{\xi} = \mathbf{L}_2 H(\bar{\xi}^0 - \Delta\bar{\xi}); \tag{10.6}$$

$$\Delta\bar{\xi} = \theta\bar{\eta}; \tag{10.7}$$

$$(\mathbf{P}^{\mathrm{T}}_{\xi_0}\Delta\bar{\xi}) = \frac{(H - H_0)H_0 L_s}{EF}, \tag{10.8}$$

где $\mathbf{T}$ — нагрузка с учетом усилий от тросов; $\mathbf{P}_{\xi}$ — усилия в подвесках тросов; H — распор тросовой фермы; $\bar{\xi}$ — обобщенная координата; θ — диагональная матрица связи, $\mathbf{P}_{\xi 0}$ — нагрузка на тросовые фермы в начальном состоянии, $\Delta\bar{\xi}$ — проекции прогибов тросовой фермы, H_0 — распор в начальном состоянии; L_s — приведенная дли-

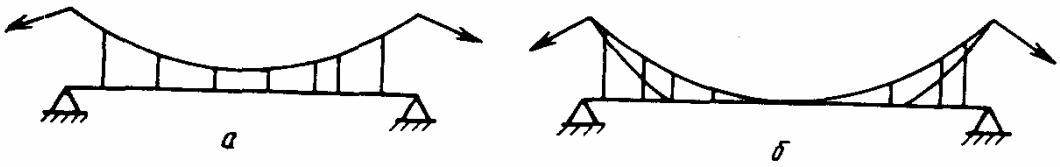


Рис. 10.31. Поперечные сечения висячих мостов:

а — однопоясная система; *б* — однопоясная система с присоединением тросов к балке

на троса, η — вектор прогибов балки жесткости, $\mathbf{L}_{10}$ — матрица влияния прогибов балки, EF — жесткость на растяжения. Изменением матрицы $\mathbf{L}_2$ учитывается соединение тросов с балкой жесткости.

Таким способом рассчитаны конструкции, поперечные сечения которых показаны на рис. 10.31. Решение уравнений проведено методом Ньютона — Рафсона с использованием СМПО.

Следует отметить, что если натяжение в подвеске нижнего яруса тросовых ферм рассматривать как дополнительное нагружение (интенсивность которого зависит от деформации системы в целом), то удается свести расчет двухъярусных систем (рис. 10.32) к расчету одноярусных (рис. 10.33). Система уравнений в этом случае распадается, так как уравнения для нижнего яруса используются как вспомогательные. Учет конкретных особенностей висячих мостов дает наиболее быстросходящиеся решения, которые ищутся путем минимизации функции вида

$$F(\mathbf{x})=\sum_{i=1}^{n} f_i^2(\mathbf{x});$$

$$f_i(\mathbf{x})=\mathbf{C}_i^{\tau}\eta-\mathbf{x}_i=\mathbf{C}_i^{\tau}(E+\mathbf{L}_{10}\mathbf{A}^*)^{-1}(\mathbf{B}^*\mathbf{T}_R)-\mathbf{x}_i;$$

$$\mathbf{C}_i=\frac{EF}{L_s}\left[(\mathbf{L}_2\mathbf{y}_{0i})^{\tau}\mathbf{A}+(\mathbf{L}_2\mathbf{z}_{0i})^{\tau}\mathbf{B}\right];$$

$$\mathbf{B}^*=\mathbf{L}_{10}\mathbf{Q},\quad \mathbf{T}_R=\mathbf{L}_{10}\left(\sum_{1}^{4}\mathbf{L}_2\mathbf{y}_{0i}\mathbf{x}_i\right),$$

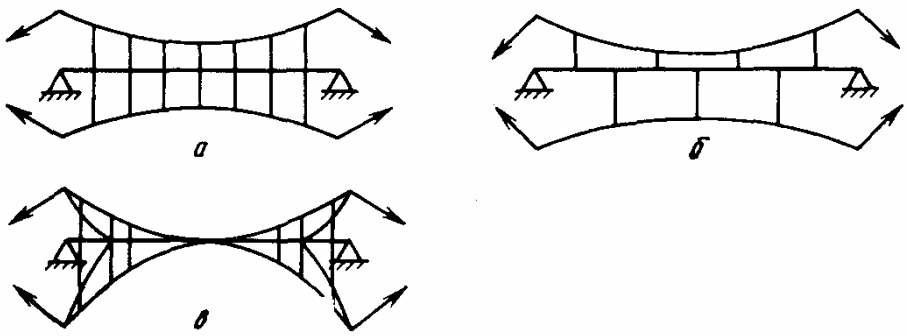


Рис. 10.32. Предварительно напряженная двухпоясная система висячего моста:

а — с одинаковым шагом подвесок по верхнему и нижнему поясам; *б* — с разным шагом подвесок; *в* — с непосредственным соединением тросов к балке

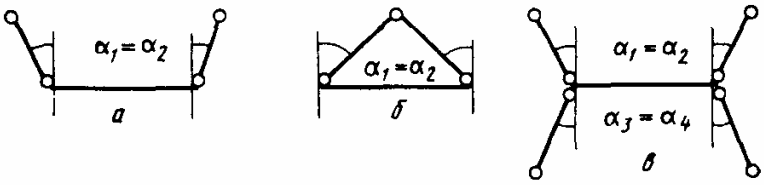


Рис. 10.33. Типовые сечения:

а — однопоясная двухкабельная система; *б* — монокабельная однопоясная система; *в* — предварительно напряженная двухпоясная система

где $\mathbf{x}_i$ — i-е изменение распора от временной нагрузки; $\mathbf{y}_{0i}$ и $\mathbf{z}_{0i}$ — начальные координаты тросов; **A**, **B** — матрицы связи; **Q** — вектор временно́й нагрузки; $\mathbf{B}^*$ — вектор «влияния» балки жесткости; $\mathbf{T}_R$ — вектор «влияния» тросовой части; $\mathbf{A}^*$ — матрица, учитывающая пространственность статического воздействия.

Для систем, имеющих одинаковый шаг подвесок по верхнему и нижнему поясу (см. рис. 10.34) изменение распора от временной нагрузки нижнего пояса (например x_3), можно выразить через усилие верхнего пояса (x_1)

$$x_3 = -(\mathbf{C}_3^*, \eta^*) + k\left[-x_1 + (\mathbf{C}_1^*, \eta^*)\right]. \qquad (10.9)$$

Это уравнение позволяет при симметричной расчетной схеме и симметричной в поперечном сечении нагрузке свести расчет к одной тросовой ферме

$$x_1 = x_2;\ x_3 = x_4 \text{ — с учетом симметрии;}$$

$$x_3 = f(x_1)$$

$$x_1 = [\mathbf{C}_1^{\mathrm{T}}(\mathbf{E} + \mathbf{L}_{10}\mathbf{A}^*)^{-1}(\mathbf{B}^* - \mathbf{T}_R).$$

Варианты расчета комбинированных висячих систем на вертикальную нагрузку рассмотрены в [19].

Применение СМПО для решения задач строительной механики с использованием интегрирующих матриц

Программное обеспечение для численной реализации матричных алгоритмов СМПО может быть эффективно применено для решения задач из различных областей техники, записанных в матричной форме. В работе [13] описан метод конечных сумм (интегрирующих матриц), примененный для решения задач судостроения.

В СМПО предусмотрены подпрограммы формирования интегрирующих матриц различных типов, которые в совокупности с остальными нужными программами ППП и обеспечивают решение разнообразных задач.

Не теряя общности в оценке возможностей СМПО в плане применения аппарата интегрирующих матриц, ограничимся рассмотрением примера расчета на собственные крутильные колебания стержня постоянного (или переменного) сечения по длине l.

Крутильные колебания стержня описываются дифференциальным уравнением вида

$$\frac{\partial}{\partial y}\left(\mathbf{GJ}_p \frac{\partial\theta}{\partial y}\right)+\mathbf{J}_п \frac{\partial^2\theta}{\partial\tau^2}=0, \qquad (10.10)$$

где $\theta(y, \tau)$ — угол поворота сечения с координатой y в момент времени τ.

Для консольного стержня должны выполняться следующие граничные условия:

при $y=0$ $\theta=0$; при $y=l$ $\mathbf{GJ}_p \frac{d\theta}{dy}=0$. Численно интегрируя уравнение (10.10) с помощью интегрирующих матриц, придем к уравнению собственных колебаний

$$\det(\mathbf{C}_q-\lambda\mathbf{E})=0, \qquad (10.11)$$

где $\mathbf{C}_q=\mathbf{J}_2\mathbf{G}\mathbf{J}_1\tilde{\mathbf{J}}$ — динамическая матрица системы; $\mathbf{J}_1$, $\mathbf{J}_2$ — интегрирующие матрицы 1-го и 2-го типа соответственно; $\mathbf{G}$, $\tilde{\mathbf{J}}$ — диагональные матрицы жесткостей на кручение и погонных моментов инерции соответственно.

К этому же уравнению можно придти, применяя метод аналогий в кручении (подобный методу Мора), предложенный в работе [11]. Следовательно, с помощью интегрирующих матриц формируется динамическая матрица C_q, решение проблемы собственных значений которой и позволяет рассчитать рассматриваемую систему на собственные колебания.

В случае стержня постоянного сечения $C_q=\frac{J}{GJ_{кр}}J_2J_1$, где $J_{кр}$ — момент инерции стержня на кручение. Рассчитываемый стержень имеет следующие характеристики:

$$GJ_{кр}=7{,}95\cdot10^6\ Н\cdot м^2;\ J=700\cdot10^{-4}\ кг\cdot м;\ l=5{,}16\ м.$$

Расчет произведен при разбиении длины l на 3, 5 и 10 равных участков соответственно.

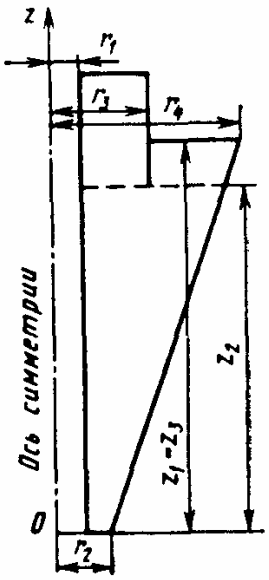
Приводим результаты сравнения численного ($P_г$) и точного ($P_т$) решений для низших тонов:

Число участков n	Частота тона, 1/с					
	1-го		2-го		3-го	
	$P_г$	$P_т$	$P_г$	$P_т$	$P_г$	$P_т$
3	102,51	102,5	308	307	—	—
5	102,5	102,5	307	308	537	538
10	102,5	102,5	308	307	515	512,5

Расчет течений жидкости в гидроциклонах

Гидроциклоны широко применяются в химической, горнодобывающей, керамической, угледобывающей, строительной, целлюлоз-

Рис. 10.34. Схема продольного разреза гидроциклона



но-бумажной, нефтедобывающей и нефтеперерабатывающей промышленности, в технике очистки сточных вод (рис. 10.34).

Как известно, винтовой поток несжимаемой жидкости в гидроциклонах описывается уравнениями

$$\operatorname{rot}\vec{V}=\lambda\vec{V}; \qquad (10.12)$$

$$\frac{\partial}{\partial q_1}(H_2H_3V_1)+\frac{\partial}{\partial q_2}(H_3H_1V_2)+\frac{\partial}{\partial q_3}(H_1H_2V_3)=0, \qquad (10.13)$$

где $\lambda=\vec{I}\cdot\operatorname{rot}\vec{I}$, $I=\vec{V}/V$, H_i — коэффициенты Ляме.

От уравнения (10.13) приходим к системе

$$H_2H_3V_1=\frac{\partial\psi}{\partial q_2},\ H_3H_1V_2=\frac{\partial\psi}{\partial q_1},\ H_3V_3=\Phi(\psi),\ \lambda=\Phi'\psi;$$

$$\frac{\partial}{\partial q_1}\left(\frac{H_2}{H_3H_1}\frac{\partial\psi}{\partial q_1}\right)+\frac{\partial}{\partial q_2}\left(\frac{H_1}{H_2H_3}\frac{\partial\varphi}{\partial q_2}\right)+\frac{H_1H_2}{H_3}\Phi(\psi)\Phi'(\psi)=0, \qquad (10.14)$$

где ψ — функция тока, $\Phi(\psi)$ — неизвестная функция; $q_3=\varphi$.

Применяя цилиндрическую систему координат ($q_1=r$, $q_2=z$, $q_3=\varphi$) и вводя функцию $\Phi(\psi)=K\psi+C$, получим

$$\frac{\partial^2\psi}{\partial z^2}+r\frac{\partial}{\partial r}\left(\frac{1}{r}+\frac{\partial\psi}{\partial r}\right)+K^2\psi=-KC;$$

$$V_r r=\frac{\partial\psi}{\partial z},\ V_z(-r)=-\frac{\partial\psi}{\partial r},$$

$$V_\varphi r=K\psi+C.$$

Уравнение (10.15) решалось конечно-разностным методом с использованием СМПО при следующих граничных условиях:

$$\psi(r_1z_3)=\psi_1=\frac{Q_{сл}}{2\pi};\ r_3\leqslant r\leqslant r_4;$$

$$\psi(r_3,\ z)=\psi_1=\frac{Q_{сл}}{2\pi};\ z_2\leqslant z\leqslant z_3;$$

$$\frac{d\psi}{\partial z}(r_1z_2)=0;\ r_1<r<r_3;$$

$$\psi(r_1z)=0;\ 0\leqslant z\leqslant z_2;$$

$$\frac{\partial\psi}{\partial z}(r_10)=0;\ r_1<r<r_2;$$

$$\psi(r_1z)=\psi_2=-\frac{Q_{цил}}{2\pi};\ r=\frac{(r_4-r_2)z}{z_1}+r_2;\ 0\leqslant z\leqslant z_1,$$

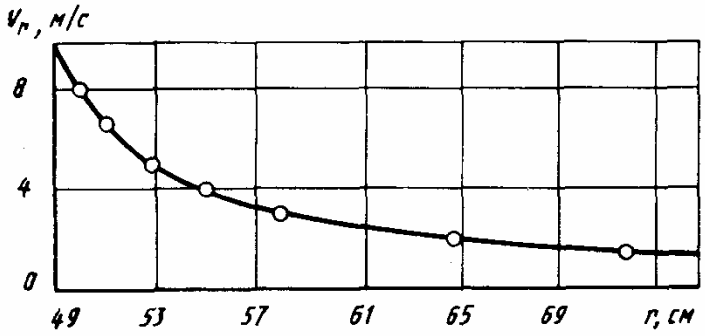


Рис. 10.35. Распределение (расчетное) тангенциальной составляющей скорости V_r вдоль радиуса r в сечении $z=480$ мм гидроциклона

где $Q_{сл}$ — расход жидкости через верхний слив; $Q_{шл}$ — расход через нижнее (шламовое) отверстие; r_1 — радиус воздушного столбца; r_2 — радиус шламового отверстия; r_3 — радиус верхнего слива; r_4 — радиус гидроциклона; z_3 — высота конической части гидроциклона.

Расчеты проводились для четырех типов гидроциклонов с различными радиусами r_3 верхнего слива ($r_1=2...5$ мм, $r_2=5$ мм, $r_3=5$, 9, 13 мм, $r_4=40$ мм, $z_3=700$ мм).

В результате решения задачи получены функция тока и три компоненты скорости; на рис. 10.35 приведен один из графиков тангенциальной составляющей скорости V_r в полости гидроциклона.

10.6. ПЛОСКИЙ ЭФФЕКТИВНЫЙ ТРЕУГОЛЬНЫЙ КОНЕЧНЫЙ ЭЛЕМЕНТ ТОНКОЙ ОБОЛОЧКИ В РАСЧЕТАХ РАЗЛИЧНЫХ КОНСТРУКЦИЙ

Плоский треугольный конечный элемент тонкой оболочки является одним из основных средств решения задач о напряженно-деформированном состоянии оболочечных конструкций. Однако для многих используемых в настоящее время плоских КЭ оболочек характерна достаточно медленная сходимость результатов, которую можно отнести за счет погрешностей в приближении формы оболочки и недостаточной степени аппроксимации тангенциальных перемещений. Низкая степень аппроксимации тангенциальных перемещений не соответствует большой величине вклада энергии мембранных деформаций в общую энергию деформации оболочки и может считаться основной причиной медленной сходимости решений, получаемых при достаточно мелкой сетке плоских КЭ, когда неучет естественной кривизны оболочки уже не является существенным.

В настоящей работе рассматриваются построение и результаты тестирования плоского треугольного трехузлового КЭ тонкой оболочки с 6 степенями свободы в узле.

1. Изгибная составляющая. В качестве изгибной составляющей КЭ оболочки принят КЭ пластины, основанный на дискретной теории Кирхгофа — *DKT* [42], который оценен как предпочтительный среди треугольных КЭ пластин с 9-ю изгибными степенями свободы. Формулировка КЭ DKT основана на теории пластин типа Тимошенко при введении гипотезы о равенстве нулю угла сдвига в отдельных точках элемента, а также с учетом пренебрежимой ма-

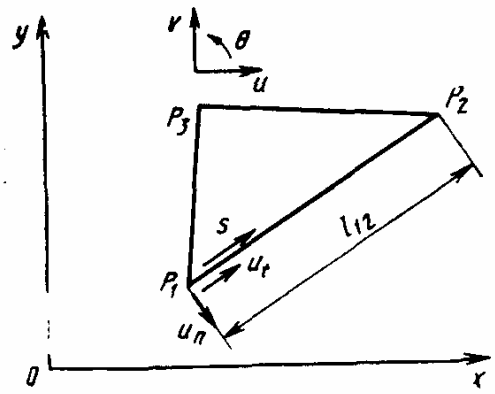


Рис. 10.36. Формулировка мембранной составляющей КЭ оболочки

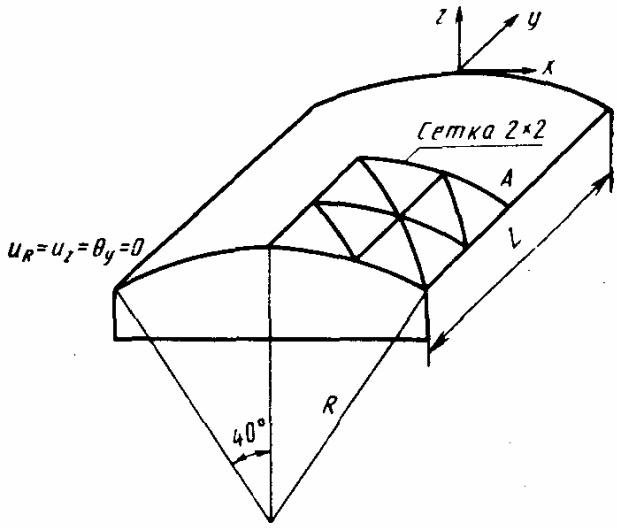


Рис. 10.37. Цилиндрическая «крыша» ($L=0{,}5$ м; $R=0{,}25$ м; $E=4{,}32\cdot10^{12}$ Н/м^2; $\nu=0$; давление по поверхности оболочки равно 900 кПа)

лости энергии деформации сдвига по сравнению с энергией изгиба. Выражение для энергии изгиба содержит лишь первые производные углов поворота β_x, β_y нормали к недеформированной срединной плоскости и не содержит функции прогиба W; поэтому для формулировки соотношений КЭ относительно «инженерных» степеней свободы (W, W_x, W_y) должна быть установлена связь следующего вида:

$$\beta_x=\mathbf{H}_x^{\mathrm{T}}(\xi,\ h)\,\mathbf{U},\ \beta_y=\mathbf{H}_y^{\mathrm{T}}(\xi,\ h)\,\mathbf{U},$$

где $\mathbf{U}=[W_1,\ \theta_{x_1},\ \theta_{y_1},\ W_2,\ \theta_{x_2},\ \theta_{y_2},\ W_3,\ \theta_{x_3},\ \theta_{y_3}]^{\mathrm{T}}$ — степени свободы КЭ; $\mathbf{H}_x$, $\mathbf{H}_y$ — девятикомпонентные функции формы; ξ, h — координаты площади треугольника.

2. Мембранная составляющая. В качестве мембранной составляющей КЭ оболочки принят совместный треугольный трехузловой элемент [39] с квадратичным полем перемещений, обладающий поступательными u, v и вращательной θ_z степенью свободы в узле. Элемент выдерживает кусочное тестирование, что с учетом совместности гарантирует сходимость к точному решению. Показано, что скорость сходимости результатов для элемента существенно выше, чем для КЭ с постоянной деформацией и близка к скорости сходимости для элемента *LST* с линейным полем деформаций, хотя последний обладает 12-ю степенями свободы. Формулировку соотношений КЭ не приводим; как достоинство формулировки [39] отметим, что она допускает вывод матрицы жесткости элемента в явной форме (рис. 10.36).

3. Результаты тестирования. Результаты тестирования построенного КЭ приведены ниже в сопоставлении с данными для другого плоского КЭ оболочки, в котором сочетаются элемент пластины

DKT и мембранный элемент с постоянной деформацией *CST*; для сравнения приведены также результаты, полученные с использованием криволинейного элемента *DKT+CST*.

Цилиндрическая панель («крыша») (рис. 10.37)

В табл. 10.2 приведены значения вертикального перемещения точки А, отнесенные к точному значению 0,3024 см.

Таблица 10.2

Сетка	Тип конечного элемента		
	DKT+CST (пл.)	*DKT*+КЭ [39] (пл.)	*DKT+CST* [59] (кр.)
2×2	0,974	1,416	0,863
4×4	0,707	1,007	0,644
6×6	0,802	0,986	0,768
8×8	0,866	0,997	0,845

Цилиндрическая оболочка, опертая на диафрагмы, нагруженная сосредоточенными силами (рис. 10.38).

Значения прогиба под сосредоточенной силой, приведенные в табл. 10.3, отнесены к точному значению $0{,}18248\times10^{-6}$ см.

Видно, что ошибка, вносимая неучетом естественной кривизны в формулировке плоского КЭ, уменьшается при использовании в качестве мембранной составляющей КЭ [39].

Таблица 10.3

Сетка	Тип конечного элемента		
	DKT+CST (пл.)	*DKT*+КЭ [39] (пл.)	*KT+CST* [59] (кр.)
2×2	0,054	0,084	1,324
4×4	0,462	0,777	0,5317
6×6	0,727	0,869	0,8202
8×8	0,860	0,9284	0,948
10×10	0,930	0,986	0,988

Консольная арка (рис. 10.39)

В табл. 10.4 приведены значения прогиба в точке *A*, отнесенные к точному значению, равному $0{,}37699\times10^{-3}$ м.

Таблица 10.4

Сетка	Тип конечного элемента	
	DKT+CST (пл.)	*DKT*+КЭ [40] (пл.)
3×1	0,9469	0,9469
6×1	0,989	0,989

Из этой таблицы видно, что наличие вращательной мембранной степени свободы и соответствующей жесткости не препятствует

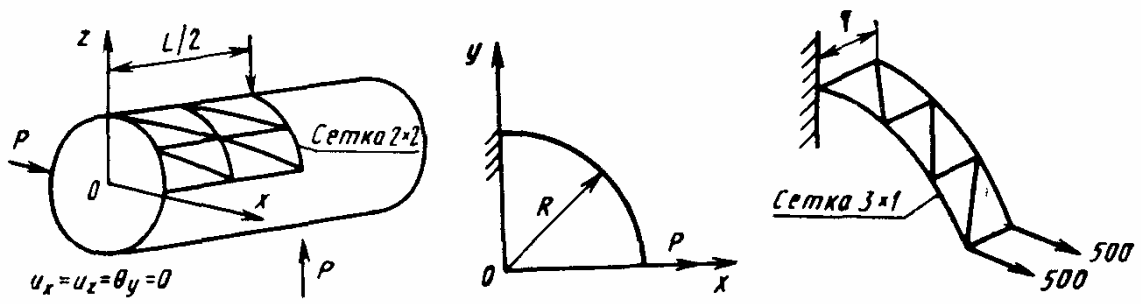


Рис. 10.38. Замкнутая цилиндрическая оболочка, опертая на диафрагмы ($R=3$ м; $L=6$ м, $h=0{,}03$ м, $E=3\cdot10^{10}$ Н/м^2; $\nu=0{,}3$, $P=1$ Н)

Рис. 10.39. Расчетная схема консольной арки ($R=2$ м, $h=0{,}1$ м; $P=10^3$ Н; $E=2\cdot10^{11}$ Н/м^2; $\nu=0$)

корректному воспроизведению чисто изгибной формы и смещений элементов конструкции как твердого целого.

Таким образом, показано, что построенный КЭ превосходит некоторые известные элементы по быстроте сходимости, и решение сходится в рассмотренных случаях к точному решению для непологих оболочек. Соотношения описанного КЭ запрограммированы на языке ФОРТРАН-IV и подключены в ППП СУМРАК средствами СМПО.

10.7. ОПРЕДЕЛЕНИЕ ДИНАМИЧЕСКОЙ РЕАКЦИИ КОНСТРУКЦИЙ С ПОМОЩЬЮ ВЕКТОРОВ ЛАНЦОША

Рассмотрим уравнения движения конструкции в форме метода перемещений

$$\mathbf{M}\ddot{\mathbf{u}}+\mathbf{K}\mathbf{u}=\mathbf{r}\,\varepsilon(t), \tag{10.15}$$

где $\mathbf{M}$ и $\mathbf{K}$ — матрицы масс и жесткости размерами $N\times N$ соответственно; $\mathbf{r}$ — вектор формы нагружения, не зависящий от времени; $\varepsilon(t)$ — скалярная функция времени.

Последовательность Крылова для данной задачи будет иметь следующий вид:

$$\{\mathbf{r},\ \mathbf{K}^{-1}\mathbf{M}\mathbf{r},\ (\mathbf{K}^{-1}\mathbf{M})^2\mathbf{r},\ldots,(\mathbf{K}^{-1}\mathbf{M})^m\mathbf{r}\}. \tag{10.16}$$

Подпространство Крылова, натянутое на векторы этой последовательности, обладает прекрасными аппроксимационными свойствами [29]. В работе [52] предлагается использовать это пространство непосредственно для определения динамической реакции (10.15), не определяя собственных форм. Для этого сначала определяется базис $\mathbf{Q}_m$ последовательности (10.16), процедура построения которого известна как процесс Ланцоша. Уравнение (10.15) в этом базисе примет вид

$$\mathbf{T}_m\ddot{\mathbf{x}}+\mathbf{x}=\beta_1\mathbf{e}_1\,\varepsilon(t), \tag{10.17}$$

где $\mathbf{T}_m$ — трехдиагональная матрица, образованная диагональными α_i, $i=1, \ldots, m$ и поддиагональными β_i, $i=2, \ldots, m$ коэффициента-

ми, полученными в процессе Ланцоша (см. разд. 7.5), $\mathbf{e}_1=\{1, 0, 0, ..., 0\}^{\mathrm{T}}$. Задачу (10.17), размерность которой $m \ll N$, можно решать как с помощью численного интегрирования, например, методом Ньюмарка, так и путем определения собственных векторов $\mathbf{T}_m$ и отклика, полученного суперпозицией несвязанных уравнений.

Если матрица $\mathbf{K}$ вырождена, что соответствует свободной (не имеющей связей) конструкции, можно применить процесс Ланцоша к паре матриц $(\mathbf{M}, \mathbf{K}-\sigma\mathbf{M})$. В базисе Ланцоша для матриц $(\mathbf{M}, \mathbf{K}-\sigma\mathbf{M})$ задача (10.15) примет вид

$$\tilde{\mathbf{T}}_m\ddot{\mathbf{x}}+(\sigma\tilde{\mathbf{T}}_m+\mathbf{E})\,\mathbf{x}=\tilde{\beta}_1\mathbf{e}_1\,\varepsilon(t), \tag{10.18}$$

где $\tilde{\mathbf{T}}_m$ и $\tilde{\beta}_1$ построены в результате процесса Ланцоша для $(\mathbf{M}, \mathbf{K}-\sigma\mathbf{M})$. Задача (10.18) не сложнее для решения, чем задача (10.17). Исследуя свободные колебания задачи (10.18), получим (индекс m опущен)

$$(\sigma\tilde{\mathbf{T}}+\mathbf{E})\,\mathbf{z}=\omega_i^2\tilde{\mathbf{T}}\mathbf{z}.$$

Отсюда $\tilde{\mathbf{T}}\mathbf{z}=\frac{1}{\omega_i^2-\sigma}\mathbf{z}$. Обозначая $\lambda_i=\frac{1}{\omega_i^2-\sigma}$, получим

$$\tilde{\mathbf{T}}\mathbf{Z}=\mathbf{Z}\Lambda. \tag{10.19}$$

Из (10.19) определяем собственные значения Λ и векторы $\mathbf{Z}$, а затем ω_i — собственные значения (10.18). Если, например, $\varepsilon(t)=\sin\omega t$, то

$$\mathbf{x}=\beta_1\mathbf{Z}\Lambda^{-1}[\Lambda^{-1}+(\sigma-\omega^2)\mathbf{E}]^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{e}_1\sin\omega t.$$

Поскольку $\mathbf{u}=\mathbf{Q}_m\mathbf{x}$,

$$\mathbf{u}=\beta_1\mathbf{Q}\mathbf{Z}\Lambda^{-1}[\Lambda^{-1}+(\sigma-\omega^2)\mathbf{E}]^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{e}_1\sin\omega t,$$

причем эта формула включает как упругие деформации точек тела, так и движения их как твердого тела.

Сдвиг σ можно применять и при анализе поведения закрепленных конструкций. Выбирая его близким к ω^2, мы уменьшим число векторов Ланцоша, необходимых для получения хорошего решения.

Описанная методика реализована в ППП СУМРАК. Численные эксперименты показывают хорошую сходимость процесса. Для получения результата требуется меньшее число векторов Ланцоша, чем форм при решении методом суперпозиции собственных форм. Так как векторы Ланцоша определяются примерно в 2—3 раза быстрее, чем собственные формы, метод весьма эффективен, особенно когда требуется определить реакцию при одном $\mathbf{r}$ для разных $\varepsilon(t)$. Параллельно метод дает возможность определить собственные формы и частоты.

Метод также позволяет учесть демпфирование конструкции; с его помощью была рассчитана натурная конструкция (некоторые результаты расчета показаны на рис. 10.16).

Глава 11. Применение СМПО в системе автоматизированного проектирования (САПР) машиностроительных конструкций

Важную роль в расчете и проектировании конструкций может сыграть метод конечных элементов (МКЭ), который имеет большие возможности как по объему, так и по содержанию задач.

Однако МКЭ в повседневной практике проектирования применяется пока еще недостаточно широко. Это объясняется рядом причин: нестабильностью результатов, полученных в результате применения данного метода, из-за неумелого или недостаточного использования типов конечных элементов; трудоемкостью организации расчета по МКЭ в связи с переработкой огромного объема конечно-элементных данных на стадиях подготовки и обработки информации и т. п.

Наиболее эффективным средством устранения этих недостатков является минимизация объема входных для программ МКЭ данных и задание их для ЭВМ на проблемно-ориентированном языке, понятном для инженера-проектировщика, а также организация в течение счета непрерывного диагностирования конечно-элементной информации. Программным средством решения данной проблемы может служить лишь такой язык программирования, который обеспечивает основные функции базы данных и создан специально для обслуживания задач МКЭ. СМПО — именно тот инструмент, с помощью которого достигается максимальная простота и надежность применения программного комплекса МКЭ.

Изложению основных разработок в области автоматизации расчета и проектирования конструкций типа подкрепленной ребристой оболочки и посвящена данная глава.

11.1. ФОРМУЛИРОВКА ПРОБЛЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ КОНСТРУКТИВНО-СИЛОВОЙ МОДЕЛИ ОБЪЕКТА; РАЗРАБОТКА АЛГОРИТМОВ И МЕТОДОВ ГЕНЕРАЦИИ ДАННЫХ ДЛЯ КОНЕЧНО-ЭЛЕМЕНТНЫХ МОДЕЛЕЙ МАШИНОСТРОИТЕЛЬНЫХ КОНСТРУКЦИЙ

Неотъемлемая часть проектирования — расчет конструкций на прочность, который в настоящее время может быть выполнен лишь МКЭ с применением мощных ЭВМ. Развитие расчета на прочность прошло большой путь от частных методов до универсальных типа МКЭ. Однако применение МКЭ к расчету таких конструкций, как, например фюзеляж летательного аппарата, корпус судна, автомобиля и т. п., невозможно без создания проблемно-ориентированного программного обеспечения. Наиболее крупными из таких программных подсистем являются: РИПАК, ЛИРА, СИСТЕМА-4, ДИАНА, КАСКАД, СПРИНТ, ПРОЧНОСТЬ, ФРОНТ, NASTRAN, ASKA, MARC и многие другие. Для таких специализированных систем расчета и проектирования, которые предназначены для численного анализа прочностных свойств сложных конструкций и имеют дело с задачами, где свыше тысячи неизвестных, наиболее остро стоит

цип задания минимума сведений для ЭВМ на проблемно-ориентированном языке. Достичь этого можно лишь благодаря формализации операций, связанных с вычислениями любого рода исходных данных. Другим важнейшим аспектом является построение эффективных алгоритмов для реализации этих вычислений на ЭВМ.

Немаловажным средством решения этой задачи является наличие процедурно-ориентированного языка программирования, обеспечивающего важнейшие функции базы данных, а именно: специальных операций над простыми и блочными матрицами, модульного принципа, удобства операций ввода-вывода, символического обращения к информации, автоматической настройки структур данных, иерархической структуры и т. п. Использование таких возможностей позволит разработчику обеспечить высокое быстродействие работы его подсистемы и разместить сервисное программное обеспечение в сравнительно небольшом объеме памяти. Что касается СМПО, то этому способствует наличие возможности программирования на двух разноплановых языках ассемблер и фортран.

11.2. АРХИТЕКТУРА ПОДСИСТЕМЫ СЕРВИС — В/В — МКЭ

Структуру программного комплекса условно делят на препроцессор (ввод), процессор (решение) и постпроцессор (вывод) — три составные части, которые должны быть тесно взаимосвязаны.

От организации обслуживающих центральную часть комплекса подсистем во многом зависят надежность, качество и быстрота процессов подготовки и обработки конечно-элементной информации. Рассмотрим структуру этих подсистем на примере создания подсистемы СЕРВИС — В/В — МКЭ в рамках ППП СУМРАК.

Препроцессор (подсистема ВВОД) пакета прикладных программ СУМРАК представляет собой набор программ, в состав которого входят:

транслятор со специализированного языка;

библиотека специализированных генераторов;

библиотеки специализированных справочных данных;

исполнительные программы со средствами контроля данных от алгоритмических ошибок:

пакет программ графического отображения информации.

Схема подсистемы СЕРВИС — В/В — МКЭ изображена на рис. 11.1. Параметры рассчитываемого объекта (геометрия, геометрические и упругие характеристики поперечных сечений конечных элементов, внешние воздействия и т. п.) описываются на специализированном языке ОДА (описание данных). Параметры модели расчета задаются в структуре формально-логических моделей, каждая из которых максимально приближена к чертежу и представляет собой топологическую развертку поверхности. Исходные данные подготавливаются на специальных таблицах, снабженных пояснениями, которые облегчают их заполнение. В ЭВМ данные вводятся с различных носителей и далее произво-

цип задания минимума сведений для ЭВМ на проблемно-ориентированном языке. Достичь этого можно лишь благодаря формализации операций, связанных с вычислениями любого рода исходных данных. Другим важнейшим аспектом является построение эффективных алгоритмов для реализации этих вычислений на ЭВМ.

Немаловажным средством решения этой задачи является наличие процедурно-ориентированного языка программирования, обеспечивающего важнейшие функции базы данных, а именно: специальных операций над простыми и блочными матрицами, модульного принципа, удобства операций ввода-вывода, символического обращения к информации, автоматической настройки структур данных, иерархической структуры и т. п. Использование таких возможностей позволит разработчику обеспечить высокое быстродействие работы его подсистемы и разместить сервисное программное обеспечение в сравнительно небольшом объеме памяти. Что касается СМПО, то этому способствует наличие возможности программирования на двух разноплановых языках ассемблер и фортран.

11.2. АРХИТЕКТУРА ПОДСИСТЕМЫ СЕРВИС — В/В — МКЭ

Структуру программного комплекса условно делят на препроцессор (ввод), процессор (решение) и постпроцессор (вывод) — три составные части, которые должны быть тесно взаимосвязаны.

От организации обслуживающих центральную часть комплекса подсистем во многом зависят надежность, качество и быстрота процессов подготовки и обработки конечно-элементной информации. Рассмотрим структуру этих подсистем на примере создания подсистемы СЕРВИС — В/В — МКЭ в рамках ППП СУМРАК.

Препроцессор (подсистема ВВОД) пакета прикладных программ СУМРАК представляет собой набор программ, в состав которого входят:

транслятор со специализированного языка;

библиотека специализированных генераторов;

библиотеки специализированных справочных данных;

исполнительные программы со средствами контроля данных от алгоритмических ошибок:

пакет программ графического отображения информации.

Схема подсистемы СЕРВИС — В/В — МКЭ изображена на рис. 11.1. Параметры рассчитываемого объекта (геометрия, геометрические и упругие характеристики поперечных сечений конечных элементов, внешние воздействия и т. п.) описываются на специализированном языке ОДА (описание данных). Параметры модели расчета задаются в структуре формально-логических моделей, каждая из которых максимально приближена к чертежу и представляет собой топологическую развертку поверхности. Исходные данные подготавливаются на специальных таблицах, снабженных пояснениями, которые облегчают их заполнение. В ЭВМ данные вводятся с различных носителей и далее произво-

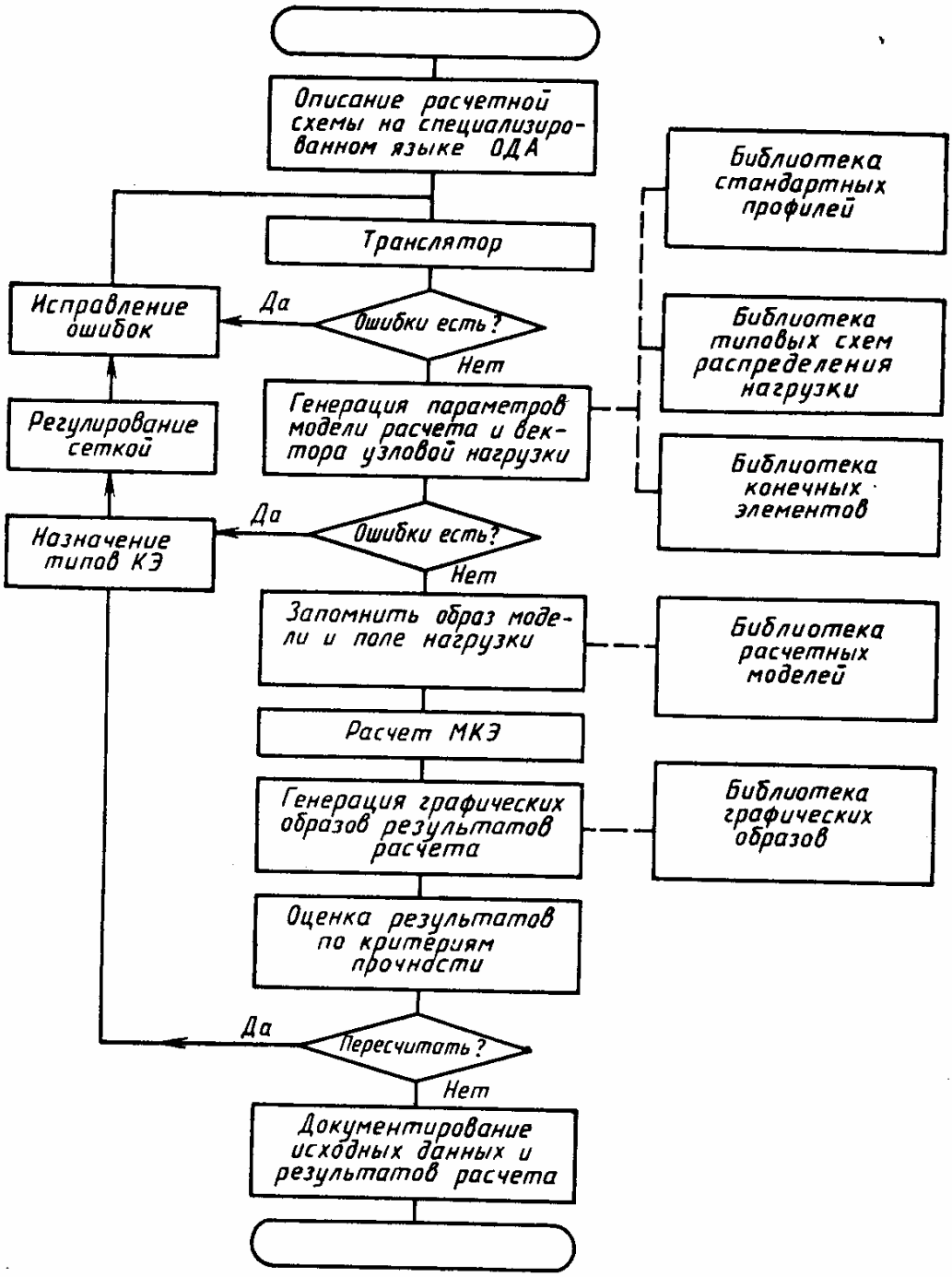


Рис. 11.1. Схема подсистемы СЕРВИС — В/В — МКЭ

дится обработка и контроль информации: перевод языковых включений и развертка массивов, автоматическая настройка размеров матриц, проверка семантических и синтаксических ограничений и т. п. Иными словами, создается исходный файл данных, который обрабатывается однопроходным транслятором интерпретирующего типа. Предложения исходного файла просматриваются последовательно, осуществляется их синтаксический контроль и преобразование во внешний файл данных. Форма представления данных в файле, полученном в результате работы транслятора, — табличная и блочная. В этот файл входят:

таблицы (матрицы) топологической связи номеров узлов модели с конечными элементами, координат узлов модели, геометрических и упругих характеристик;

блочная таблица (вектор) узловой нагрузки;

Таблица 11.1

Этап	Возможность	Этап	Возможность
Подготовка данных для расчета на прочность	**Язык описания данных:** координат узлов в локальной и глобальной системе координат; форм и типов конечных элементов; граничных условий; внешнего воздействия **Генерация данных: информационное обеспечение:** библиотека стандартных профилей; библиотека типовых схем распределения внешнего воздействия; библиотека конечных элементов; **программные средства:** генераторы сеток; генератор граничных условий; генератор узловой нагрузки; генератор топожесткостных характеристик сечений конечных элементов; генератор слияния топожесткостных данных двух моделей в единую модель. **Методическое обеспечение:** технология описания модели на топологических развертках; технология проведения расчетов посредством «жестких форматов»; технология автоматической визуализации исходной и результирующей информации на графопостроителе	Контроль и визуализация информации	**Технические средства:** ЕС ЭВМ (ОС; MFT, SVS, MVT); графопостроитель **Программные средства контроля:** проверка семантики и синтаксиса языка описания модели в структуре топологических разверток; диагностика алгоритмических отказов с указанием места и причины ошибки на развертке; исключение случая вырожденности матрицы жесткости. **Графические средства контроля:** изображение исходного и деформированного состояния модели расчета в пространстве и на плоскости под любым углом зрения; изображение модели описания (развертки) с указанием желаемой информации (нумерация узлов и т. п.); построение эпюр напряжений, усилий, полей внешних воздействий; изображение форм колебаний контура; построение балочных эпюр внешнего воздействия
		Редактирование данных	Автономное или комплексное

таблицы графической информации для визуализации параметров модели расчета и полей внешних воздействий на периферийных устройствах.

Проверенная программными и графическими средствами конечно-элементная информация передается в программы МКЭ.

Генерация данных для программ МКЭ в подсистеме СЕРВИС—В/В—МКЭ происходит в режиме диалога. Передача данных для программ МКЭ, написанных на универсальных языках программирования, происходит с помощью файлов. Информация в этих файлах представлена в символьном виде во внешнем формате языка фортран.

Реализованные в подсистеме СЕРВИС—В/В—МКЭ (версия 1.0) средства основаны на применении специализированных программных средств, ориентированных на решение задач МКЭ (табл. 11.1).

11.3. ВХОДНОЙ ЯЗЫК ПОДСИСТЕМЫ

Среди средств автоматизации подготовки данных важное место занимает входной язык, с помощью которого осуществляется описание рассчитываемого объекта и параметры модели расчета. Допустимыми символами языка являются все буквы русского алфавита, все буквы латинского алфавита, цифры 0...9, а также знаки + — / , : . = ( ) ∗.

Структура всех предложений языка одинакова и имеет вид

⟨предложение⟩ : : = ⟨бланк⟩|⟨список — шкала⟩|⟨матрица⟩
⟨бланк⟩ : : = ⟨балочные КЭ⟩|⟨пояса⟩|⟨обшивка⟩|
⟨продольные балки⟩|⟨формы сечения⟩|
⟨массовая модель⟩ ⟨внешние силы⟩
⟨внешние моменты⟩|⟨разнос по связям⟩|
⟨граничные условия⟩|⟨типы КЭ⟩|
⟨упругие характеристики⟩
⟨список — шкала⟩ : : = ⟨число списков⟩ ⟨список⟩|
⟨список — шкала⟩ ⟨список⟩
⟨число списков⟩ : : = ⟨целое⟩ : ⟨целое⟩
⟨список⟩ : : = ⟨целое⟩|⟨список⟩ ⟨целое⟩
⟨матрица⟩ : : = ⟨размеры матрицы⟩/∗⟨значения⟩
⟨размеры матрицы⟩ : : = ⟨целое⟩ : ⟨целое⟩
⟨значения⟩ : : = ⟨вещественное⟩|⟨значения⟩ ⟨вещественное⟩
⟨БАЛОЧНЫЕ КЭ⟩ : : = ⟨строка БКЭ⟩|⟨балочные КЭ⟩ ⟨строка БКЭ⟩
⟨строка БКЭ⟩ : : = ⟨зона РИС⟩ ⟨параметры⟩
⟨зона РИС⟩ : : = ⟨целое⟩ ⟨целое⟩ ⟨целое⟩ ⟨целое⟩
⟨параметры⟩ : : = ⟨*H*⟩ ⟨*T*⟩ ⟨Б⟩
⟨*H*⟩ : : = ⟨*T*⟩ : : = ⟨вещественное положительное⟩
⟨Б⟩ : : = ⟨знак числа⟩ ⟨целое⟩
⟨ПОЯСА⟩ : : = ⟨строка п⟩|⟨пояса⟩ ⟨строка п⟩
⟨строка п⟩ : : = ⟨зона РИС⟩ ⟨БП⟩
⟨БП⟩ : : = ⟨целое⟩|— ⟨число⟩
⟨ОБШИВКА⟩ : : = ⟨строка о⟩|⟨обшивка⟩ ⟨строка о⟩
⟨строка о⟩ : : = ⟨зона РИС⟩ ⟨Т⟩
⟨ПРОДОЛЬНЫЕ БАЛКИ⟩ : : = ⟨строка Л⟩|⟨продольные балки⟩
⟨строка Л⟩
⟨строка Л⟩ : : = ⟨строка БКЭ⟩
⟨ФОРМЫ СЕЧЕНИЙ⟩ : : = ⟨группа⟩|⟨формы сечений⟩ ⟨группа⟩
⟨группа⟩ : : = ⟨начало⟩ ⟨габариты⟩ ⟨описание⟩ ⟨конец⟩
⟨начало⟩ : : = ⟨целое⟩ ⟨о⟩ ⟨о⟩ ⟨вещественное⟩

⟨габариты⟩ : : = ⟨*T*⟩ ⟨*H*⟩ ⟨Л⟩ ⟨П⟩
⟨описание⟩ : : = ⟨элемент⟩ | ⟨описание⟩ ⟨элемент⟩
⟨элемент⟩ : : = ⟨*Y**⟩ ⟨*Z**⟩ ⟨*H**⟩ ⟨*B**⟩
⟨*Y**⟩ : : = ⟨*Z**⟩ : : = ⟨*H**⟩ : : = ⟨B*⟩ : : = ⟨Л⟩ : : = ⟨П⟩ : : = ⟨число⟩
⟨конец⟩ : : = ⟨о⟩ ⟨о⟩ ⟨о⟩ ⟨о⟩
⟨о⟩ : : = 0
⟨МАССОВАЯ МОДЕЛЬ⟩ : : = ⟨строка ММ⟩ | ⟨массовая модель⟩
⟨строка ММ⟩
⟨строка ММ⟩ : : = ⟨координаты⟩ ⟨масса⟩
⟨координаты⟩ : : = ⟨*X*⟩ ⟨*Y*⟩ ⟨*Z*⟩
⟨масса⟩ : : = ⟨*X*⟩ : : = ⟨*Y*⟩ : : = ⟨*Z*⟩ : : = ⟨вещественное⟩
⟨ВНЕШНИЕ СИЛЫ⟩ : : = ⟨строка ВС⟩ | ⟨внешние силы⟩
⟨строка ВС⟩
⟨строка ВС⟩ : : = ⟨координаты⟩ ⟨сила⟩ |
⟨строка ВС⟩ ⟨сила⟩
⟨сила⟩ : : = ⟨*PX*⟩ ⟨*PY*⟩ ⟨*PZ*⟩
⟨*PX*⟩ : : = ⟨*PY*⟩ : : = ⟨*PZ*⟩ : : = ⟨вещественное⟩
⟨ВНЕШНИЕ МОМЕНТЫ⟩ : : = ⟨строка ВМ⟩ |
⟨внешние моменты⟩ ⟨строка ВМ⟩
⟨строка ВМ⟩ : : = ⟨координаты⟩ ⟨момент⟩ |
⟨строка ВМ⟩ ⟨момент⟩
⟨момент⟩ : : = ⟨*MX*⟩ ⟨*MY*⟩ ⟨*MZ*⟩
⟨*MX*⟩ : : = ⟨*MY*⟩ : : = ⟨*MZ*⟩ : : = ⟨вещественное⟩
⟨РАЗНОС ПО СВЯЗЯМ⟩ : : = ⟨СТРОКА Р⟩ | ⟨РАЗНОС ПО СВЯЗЯМ⟩
⟨СТРОКА Р⟩
⟨строка Р⟩ : : = ⟨*X*⟩ ⟨*Y*⟩ ⟨*Z*⟩ ⟨*U*⟩
⟨ГРАНИЧНЫЕ УСЛОВИЯ⟩ : : = ⟨строка ГУ⟩ |
⟨граничные условия⟩ ⟨строка ГУ⟩
⟨строка ГУ⟩ : : = ⟨зона РИС⟩ | ⟨строка ГУ⟩ ⟨зона РИС⟩
⟨ТИПЫ КЭ⟩ : : = ⟨строка Т⟩ | ⟨типы КЭ:⟩ ⟨строка Т⟩
⟨строка Т⟩ : : = ⟨зона РИС⟩
⟨параметр⟩ : : = ⟨численное значение⟩ | ⟨литерал⟩ |
⟨задание на генерацию⟩ | ⟨список⟩
⟨численное значение⟩ : : = ⟨знак числа⟩ ⟨число⟩
⟨знак числа⟩ : : = + | —
⟨число⟩ : : = ⟨целое⟩ | ⟨дробное⟩
⟨целое⟩ : : = ⟨последовательность цифр⟩
⟨дробное⟩ : : = ⟨последовательность цифр⟩ · ⟨последовательность цифр⟩
⟨литерал⟩ : : = ⟨последовательность допустимых знаков⟩
⟨задание на генерацию⟩ : : = (⟨начальное значение, конечное значение⟩) | [⟨начальное значение, конечное значение⟩ | ⟨количество повторений⟩ ∗ ⟨список⟩
⟨начальное значение⟩ : : = ⟨число⟩
⟨конечное значение⟩ : : = ⟨число⟩
⟨список⟩ : : = ⟨целое⟩ , ⟨список⟩ | ⟨целое⟩
⟨количество повторений⟩ : : = ⟨число⟩

Типы предложений условно могут быть разделены на три группы.

В первую группу входят предложения по заполнению стандартных бланков: «Балочные элементы», «Пояса», «Обшивка», «Продольные балки», «Формы сечений», «Массовая модель», «Внешние силы», «Внешние моменты», «Разнос по связям», «Граничные условия», «Типы КЭ», «Упругие характеристики», «Координаты узлов РИС». Бланки имеют структуру таблицы простого вида, могут содержать конечное число строк (не более 4095). Каждая строка бланка несет в себе однозначную информацию для описания геометрических, упругих характеристик, поля внешнего воздействия и т. п. Например, строка бланка «Балочные элементы» имеет вид СЕЧЕНИЕ СЕЧЕНИЕ ПОЯС ПОЯС ВЫСОТА ТОЛЩИНА НО-

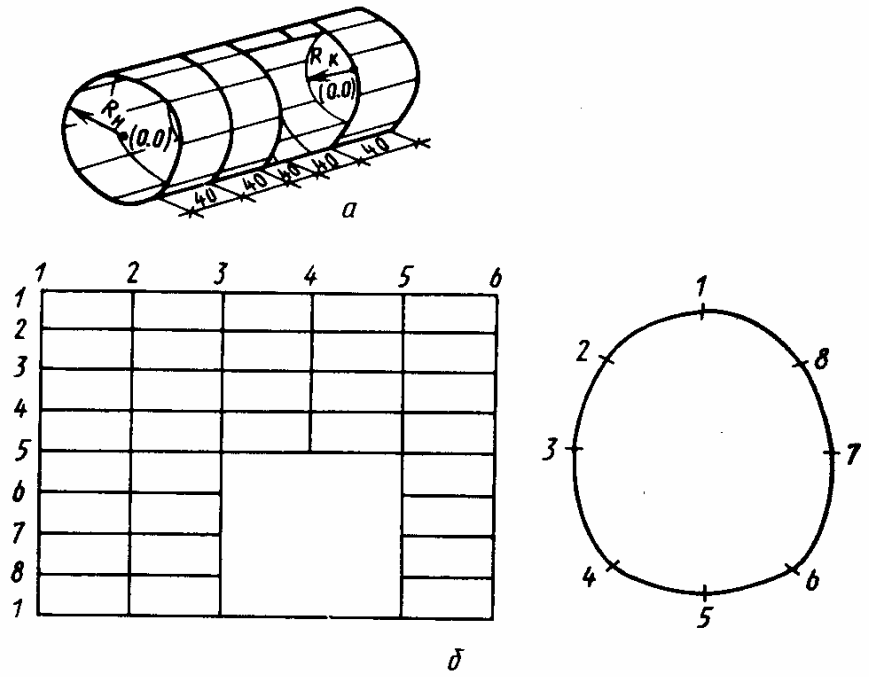


Рис. 11.2. Модель описания — регуляризованная идеализированная схема (РИС): *а* — оболочка; *б* — развертка $T=8$ (горизонтали), $P=6$ (вертикали), $Q=8$ (поля), $R_н=40$ мм, $R_к=60$ мм

МЕР, где сечение — это элемент поперечного, а пояс — продольного силового набора, и позволяет описывать поперечное сечение двухузловых балочных (изгибных) конечных элементов через НОМЕР типового профиля по библиотеке ФОРМЫ СЕЧЕНИЙ, численные значения высоты H и толщины T стенки профиля и номера узлов модели расчета, заданные в нумерации формально-логической модели описания РИС (регуляризованная идеализированная схема).

Модель РИС (рис. 11.2) занимает в методическом обеспечении подсистемы СЕРВИС—В/В—МКЭ центральное место. Она максимально приближена к структуре чертежа и описание элементов расчетной схемы в ее нумерации наиболее естественно. Кроме того, представление и обработка результатов расчета посредством применения математического аппарата теории графов однозначным образом привязаны к таким формально-логическим моделям, что исключает появление случайных ошибок в исходных данных и, следовательно, повышает их математическую достоверность.

Рассмотрим структуру строк остальных бланков описания. Строка бланка «Пояса» имеет вид СЕЧЕНИЕ СЕЧЕНИЕ ПОЯС ПОЯС НОМЕР и позволяет описывать площадь поперечного сечения стержневых элементов, работающих в схеме расчета лишь на растяжение — сжатие. Строка бланка «Обшивка» имеет вид: СЕЧЕНИЕ СЕЧЕНИЕ ПОЛЕ ПОЛЕ ТОЛЩИНА и указывает толщину листа обшивки через информацию: с какого по какое сечение, с какого по какое поле располагается лист оболочки (обшивки) толщиной ТОЛЩИНА.

Структура бланка «Продольные балки» совпадает со структурой бланка «Балочные элементы». Бланки предназначены для назначения стержневых элементов, описанных в бланке «Пояса», балочными, т. е. элементами, которые работают в модели расчета на все виды нагрузки (растяжение, сжатие, кручение и изгиб). Строки бланков «Граничные условия», «Типы КЭ», «Упругие характеристики» имеют одинаковую структуру СЕЧЕНИЕ СЕЧЕНИЕ ПОЯС ПОЯС ТИП и позволяют описать соответственно степени свободы в узлах: типы применяемых КЭ, модули упругости E (или модуль сдвига G), коэффициент Пуассона.

Строка бланка «Массовая модель» имеет структуру АБСЦИССА ОРДИНАТА АППЛИКАТА МАССА и позволяет описать массовую модель рассчитываемого объекта. Строка «Внешние силы» имеет структуру АБСЦИССА ОРДИНАТА АППЛИКАТА ПРОЕКЦИЯ ПРОЕКЦИЯ ПРОЕКЦИЯ и позволяет описать воздействие на конструкцию сосредоточенной силы посредством задания численных значений координат точки ее приложения и ее проекций на оси координат. Бланк «Внешние моменты» имеет структуру бланка «Внешние силы» и позволяет описывать внешнее воздействие через сосредоточенные моменты.

Бланки «Формы сечений» и «Разнос по связям» имеют структуру четырехстолбцовой таблицы простого вида, могут содержать конечное число строк, которые объединены в группы описаний. Бланк «Формы сечений» предназначен для описания формы профиля балки сложного сечения с целью вычисления его геометрических характеристик. Бланк «Разнос по связям» предназначен для описания геометрии расчетной схемы конструкции типового агрегата, не включенного в глобальную систему, с целью определения реакций в узлах крепления его на рассчитываемой конструкции.

Во вторую группу предложений входят предложения список-шкала по описанию топологии узлов и элементов объекта расчета. Например, последовательность номеров РИС типового сечения модели описания, изображенной на рис. 11.2, *б*, запишется в виде: 1 : 9 1 ,8 1 и по ней сгенерируется последовательность 1 : 9 1 2 3 4 5 6 7 8 1.

В третью группу предложений относятся предложения МАТРИЦА. Сюда входит, например, матрица координат узлов РИС, которая в зависимости от формы конструкции (симметрия, круговое сечение и т. п.) может быть представлена информацией, достаточной для генерирования полной таблицы. Например, для конструкции (см. рис. 11.2, *а*) матрица координат узлов имеет вид

```
1 : 2 горизонталей вертикалей
/*
абсцисса ордината радиус
/*
абсцисса ордината радиус
/*
аппликаты узлов
/*
1:2 8 6
```

```
/*
0 0 40
/*
0 0 60
/*
5      0      40      80      120      160      200
/*
```

Данной информации достаточно, чтобы по радиусам крайних сечений, координатам центров круговых крайних сечений и числу разбиений (8) сгенерировать матрицу координат узлов РИС.

Задание координат узлов модели расчета не всегда удобно осуществлять в одной системе координат. Поэтому представляется возможность отдельные части моделируемой расчетной схемы описывать в локальных системах координат, в которых они описываются наиболее естественно. Для перехода из локальной системы координат в глобальную имеются программы преобразования, использующие поворот, сдвиг и т. п. Кроме того, такие программы предполагают топологическую сборку жесткостных характеристик в единое целое для глобальной системы. Для этого достаточно описать номера узлов «склеивания» в нумерации их локальных систем.

11.4. ПРИМЕР ОСНОВНОЙ ПРОГРАММЫ ПОЛЬЗОВАТЕЛЯ-ПРОЕКТИРОВЩИКА

Модульный принцип построения пакета, предполагающий создание многоуровневых стандартных программных единиц и сокращение объема входной информации, которое стало возможным благодаря разработанным и реализованным алгоритмам и методам генерации данных, позволили приблизиться к таким программам, в которых пользователю достаточно задавать три-пять операторов языка управления задания, а основное внимание уделять описанию расчетной модели исследуемой конструкции.

Приведем для наглядности пример одной из многоуровневых основных программ пакета. На рис. 11.3 приведена программа формирования геометрических и жесткостных характеристик конечных элементов модели конструкции, изображенной на рис. 11.2.

11.5. КОНТРОЛЬ КОНЕЧНО-ЭЛЕМЕНТНЫХ ДАННЫХ

Одним из основных средств контроля исходных данных для расчета на прочность является графическое изображение параметров внешнего воздействия и конечно-элементной модели на периферийных устройствах, включая графопостроитель. Однако немаловажным средством контроля является диагностирование данных посредством отображения на экране алфавитно-цифрового дисплея пояснительных сообщений о месте, типе и причине алгоритмических отказов. Кроме того, к числу таких сообщений можно отнести информацию о конструктивных особенностях рассчитываемого объекта — вырезах, способах подкрепления, узлах приложения нагруз-

ки и ее происхождения и т. п. Последнего рода информация может выдаваться по просьбе пользователя, она формируется в программах и по умолчанию не выводится.

Эффективной является форма отображения конечно-элементных данных на устройствах графической визуализации. Для класса конструкций типа ребристой подкрепленной оболочки (фюзеляжи, автомобили, купола, вагоны и т. п.) наиболее наглядной структурой отображения данных является развертка поверхностей объекта на

```
// SERVIS JOB
// EXEC SMPOV, VR=MARAT, N=GF1
// GO.SYSIN DD*, DLM='//'
  РЕЖИМ РАБОТЫ :
1:3 1 Ø /*
1:2 8 6 /*
3*5+6/*
6 Ø 3*25 Ø /*
6 2*25 Ø 2*-25 /*
5 Ø 4Ø 8Ø 12Ø 16Ø 2ØØ /*
1:8 1, 8 /*
1:4 4 4 6 8 /*
1 3 1 8 1
5 6 1 8 1
3 5 1 4 1
/*
1 3 1 8 -0.25
5 6 1 8 -0.25
3 5 1 5 -0.25
/*
3 5 9 5 5 -5Ø
3 5 1Ø 1Ø 5 5 -5Ø
1 3 1 8 1Ø 1Ø -51
5 6 1 8 1Ø 1Ø -51
4 4 1 5 1Ø 1Ø -51
/*
5Ø Ø Ø Ø
5 5 Ø 2005
1005 2005 1005 2005
Ø Ø Ø Ø
51 Ø Ø Ø
1Ø 1Ø Ø 2Ø1Ø
1Ø1Ø 2Ø1Ø 1Ø1Ø 2Ø1Ø
Ø Ø Ø Ø
/*
1:9 1,8 1 /*
1:8 3 5 1 1 3 5 5 5 /*
//
```

Рис. 11.3. Пример программы внешнего пользователя

плоскости. Такая форма исключает графические наложения и максимально приближена к форме представления конструкции инженером, т. е. чертежу.

Наличие таких средств превращает пакет прикладных программ расчета на прочность в эффективный инструмент инженера-проектировщика и научного исследователя, освобождает их от рутинной работы, позволяет использовать свои силы для творческого процесса проектирования или научного исследования.

11.6. ОТОБРАЖЕНИЕ КОНЕЧНО-ЭЛЕМЕНТНОЙ ИНФОРМАЦИИ

При подготовке данных для конечно-элементных расчетов появление ошибок неизбежно. Возможностей автоматизированного поиска ошибок довольно много. Опыт эксплуатации различных программ общего назначения и программных комплексов дает убедительные доказательства эффективности машинного отображения расчетной информации и машинного формального анализа в обнаружении ошибок, допущенных при описании свойств конструкции. В связи с этим разработка графической программной подсистемы и снабжение рабочих программ алгоритмами диагностики ошибок является эффективным средством повышения достоверности расчетной информации и сокращения затрат труда. Кроме того, ориентация рабочих программ генерации исходных данных на решение определенного круга задач позволяет значительно расширить возможности машинного формального анализа, распределив его на содержательную часть вводимой информации.

Автоматизация процессов подготовки данных и отображения результатов конечно-элементных расчетов неразрывно связана с проблемой визуализации информации. Отображение информации может быть оперативного характера, а также может носить характер документа отчетности об исследованиях.

Для фиксации конечных результатов расчета, выявления ошибок, не поддающихся формальному анализу, предназначен графопостроитель. В связи с большим объемом разнообразных данных, вычисляемых в процессе расчетов, и необходимостью максимального упрощения процесса визуализации, требующейся для оценки расчета информации с помощью графопостроителя, перед разработчиками подсистемы САПР встают задачи разработки форм отображения информации в виде рисунков, эпюр, графиков и диаграмм. Разработка форм отображения происходит в тесном контакте с проектировщиками, пользователями предметной области. Следует отметить, что форма отображения информации может со временем измениться в сторону упрощения или усложнения операций, необходимых для ее программной реализации. В настоящее время существует немало пакетов графических программ, обладающих развитым набором процедур визуализации информации. Однако они имеют либо предметную ориентацию, либо их трудно адаптировать к имеющейся подсистеме. Практика же реального проектирования настоятельно требовала, не дожидаясь фундаментальных

разработок в области создания универсальных пакетов графических программ, на основе имеющихся средств разработать специализированное математическое обеспечение для визуализации информации конечно-элементных расчетов.

На основе базисных и функциональных программ для графопостроителя в рамках ППП СУМРАК построена проблемно-ориентированная графическая подсистема. Она написана на фортране с использованием ассемблера. Высокая модульность программ графической подсистемы позволяет постоянно расширять ее возможности, охватывая все новые области применения.

В настоящее время подсистема САПР расчета прочности позволяет выдать следующую информацию в графической форме:

расчетную модель исходной конструкции;

деформированное состояние конструкции;

эпюры нормальных напряжений в балочных элементах расчетных сечений;

эпюры нормальных напряжений в поясах расчетных панелей;

эпюры касательных напряжений в элементах обшивки в расчетных панелях;

эпюры внешнего воздействия по длине конструкции;

диаграммы геометрических характеристик поперечных сечений конечных элементов;

схемы для анализа НДС, потери устойчивости (рис. 11.4).

Исходное и деформированное состояния конструкции могут быть изображены не только в изометрии, но и в проекции трехмерного предмета на одну из плоскостей.

Посредством трехмерного изображения исходной конструкции пользователь выявляет ошибки, неточности моделирования объекта расчетной схемой. С помощью эпюр напряжений проектировщик обосновывает математическую достоверность полученных результатов расчета напряженно-деформированного состояния. Трехмерное изображение деформированного состояния, а также построение перемещений в проекциях позволяет анализировать результаты расчетов определения частот и векторов собственных и вынужденных колебаний. Эпюры внешнего нагружения по длине конструкции подкрепленной оболочки способствуют выявлению наиболее опасных сочетаний внешней нагрузки, действующей на проектируемую конструкцию, а также позволяют наглядно представить характер внешнего воздействия.

Большое внимание при расчетных исследованиях уделяется управлению печатью информации в процессе выполнения программ генерации данных. В подсистеме предусмотрены режимы подробной и краткой печати, а также режим работы программ без вывода на АЦПУ. Массивы и таблицы снабжены пояснениями о их содержательной части. Подробное описание эксплуатации подсистемы приведено в инструкциях по ее применению. Предлагаемая методология подготовки информации для расчетов предполагает представление результатов в удобной для анализа форме. Рассмотрим структуру основных таблиц результатов расчета.

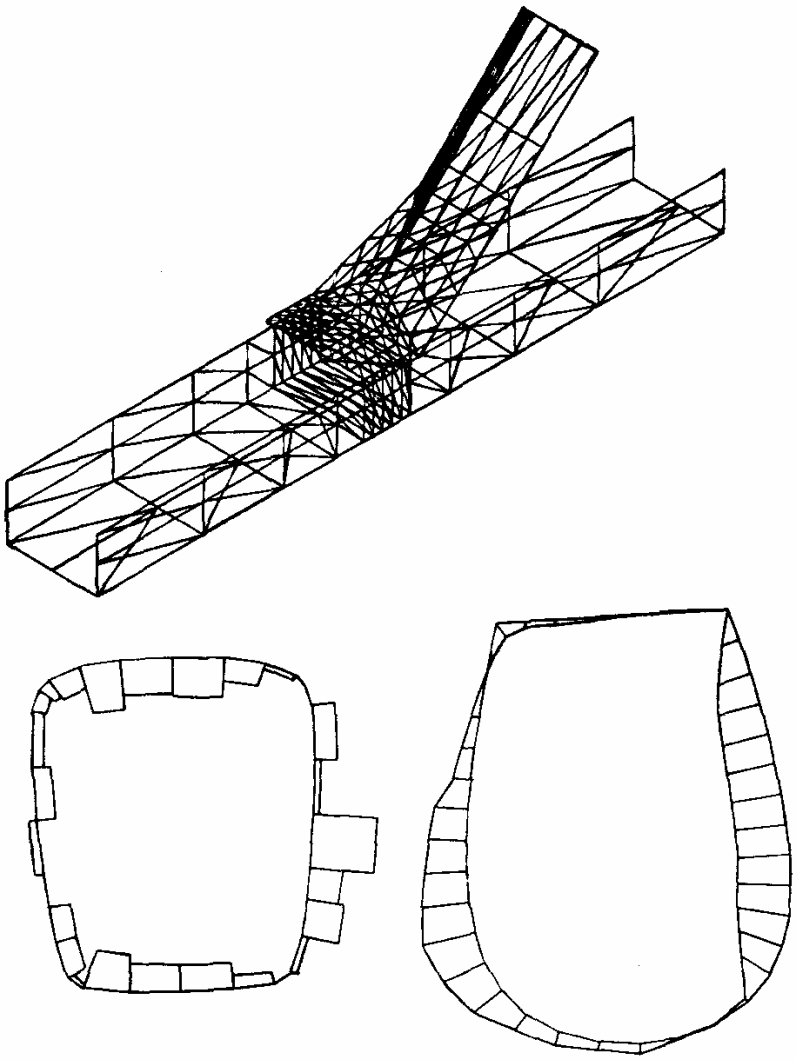


Рис. 11.4. Примеры применения средств машинной графики

Независимо от применяемой модели МКЭ результаты расчета следует выдавать на печать в одной структуре. Это таблицы эпюр нормальных напряжений во внутренней и наружной полках, таблицы эпюр сдвиговых напряжений в обшивке. таблицы нормальных напряжений в поясах имеют одинаковую структуру. Во всех таблицах столбцы представляют собой варианты нагружений. Значения нормальных напряжений в элементах сечений представлены в таблицах, число которых равно числу поперечных сечений расчетной модели. Строки этих таблиц представляют собой значения напряжений, вычисленных в начале и конце участков балочных элементов, расположенных в плоскости расчетного сечения регулярной модели. Порядок указания этих элементов в таблице совпадает с направлением обхода при выборе цепей типового поперечного сечения, для которого осуществлялась подготовка данных. Значения напряжений распечатываются с двумя знаками до и одним после запятой, что вполне согласуется с диапазоном допустимых напряжений. Значения, представленные в таблице символом $*$, относятся к числам, большим ста.

Приложения

Приложение 1. Список операторов ППП СМПО

Оператор	Стр.	Оператор	Стр.	Оператор	Стр.	Оператор	Стр.
ADSP	115	KSTAL	79	SOXL	105	UPAR	112
CON	103	KSTP	80	SOZ	92	UTIMER	107
CSP	115	KSTPZ	179	SPFAK	94	V	92
DIAG	108	MATR	83	SPFOR	95	VBL	113
EOJ	108	NOCH	91	SPSA	93	VCHT	98
FAKPZ	180	OBPZ	179	STAL	79	VOSD	103
FORPZ	180	OPR	83	STAT	106	VOSL	105
FORT	174	OTPE	109	STEK	85	VP	92
GDS	111	PE	97	STP	80	VPL	92
GEQU	111	PROG	77	STPZ	179	VV	99
GSTD	110	PTIMER	107	TEKST	95	VVOD	98
GSTH	110	RETURN	81	TIMER	106	VVS	100
IR	86	SA	93	TKO	106	VVSS	102
KBL	85	SASP	93	UD	89	ZAPO	89
KOCH	91	SLED	109	UDAL	88	ZAPOM	87
KONTR	113	SMATR	82	UDB	104	ZBL	112
KPROG	78	SOXD	103	UNPZ	180	ZR	83

Приложение 2. Список стандартных программных единиц ППП СМПО

Приложение 3. Тексты процедур СМПЮ
(для удобства сначала напечатаны колонки 78-80, затем I-72)

```
MEMBER NAME  SMPOKATB
010 //SMPOKATB PROC PASM=NOXREF,PLKED=LIST,
020 // MAC=MACSMPO,BIM=BIMSMPO,BZM=BZMSMPO,
030 //              V=SMPO00
040 //GENR   EXEC  PGM=SMPOGENR,REGION=50K
050 //STEPLIB DD   DSN=&BZM,DISP=SHR,VOL=SER=&V,UNIT=SYSDA
060 //SYSPRINT DD  DUMMY
070 //SYSIN1 DD    DUMMY
080 //SYSUT2 DD    DSN=&BIM(&N),UNIT=SYSDA,VOL=SER=&V,DISP=OLD
090 //ASM    EXEC  PGM=IUM90,PARM='NOXREF,NORLD,NOESD,&PASM',
100 //              REGION=200K
110 //STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR,VOL=SER=&V,UNIT=SYSDA
120 //SYSLIB DD    DSN=&MAC,DISP=SHR,UNIT=SYSDA,VOL=SER=&V
130 //       DD    DSN=SYS1.MACLIB,DISP=SHR
140 //SYSUT1 DD DSN=&S1,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(2,1))
150 //SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=3509,LRECL=121,RECFM=FBM)
160 //SYSPUNCH DD  DSN=&LOAD,UNIT=SYSDA,SPACE=(3200,(20,20)),
170 //     DISP=(MOD,PASS),DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
180 //SYSIN  DD    DSN=&BIM(&N),DISP=SHR,UNIT=SYSDA,VOL=SER=&V
190 //LKED   EXEC  PGM=IEWL,REGION=128K,
200 //              PARM='REUS,SIZE=(120K,32K),LIST,MAP,&PLKED'
210 //SYSLIB DD    DSN=&BZM,DISP=SHR,VOL=SER=&V,UNIT=SYSDA
220 //SYSLIN DD    DSN=&LOAD,DISP=(OLD,DELETE),
230 //              DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
240 //       DD    DDNAME=SYSIN
250 //SYSUT1 DD    DSN=&S1,DISP=(OLD,DELETE)
260 //SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=3146,LRECL=121,RECFM=FBM)
270 //SYSLMOD DD DSN=&BZM(SMPO&N),DISP=OLD,UNIT=SYSDA,VOL=SER=&V
```

MEMBER NAME SMPOKTVR

```
010 //SMPOKTVR PROC PASM=NOXREF,PLKED=LIST,
020 //              MAC=MACSMPO,BIM=BIMSMPO,
030 //              V=SMPO00
040 //ASM    EXEC   PGM=IUM90,PARM='NOXREF,NORLD,NOESD,&PASM',
050 //              REGION=200K
060 //STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR,VOL=SER=&V,UNIT=SYSDA
070 //SYSLIB DD     DSN=&MAC,DISP=SHR,UNIT=SYSDA,VOL=SER=&V
080 //        DD     DSN=SYS1.MACLIB,DISP=SHR
090 //SYSUT1 DD DSN=&S1,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(2,1))
100 //SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=3509,LRECL=121,RECFM=FBM)
110 //SYSPUNCH DD  DSN=&LOAD,UNIT=SYSDA,SPACE=(3200,(20,20)),
120 //     DISP=(MOD,PASS),DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
130 //SYSIN  DD     DSN=&BIM(&N),DISP=SHR,UNIT=SYSDA,VOL=SER=&V
 40 //LKED   EXEC   PGM=IEWL,REGION=128K,
150 //              PARM='REUS,SIZE=(120K,32K),LIST,MAP,&PLKED'
160 //SYSLIN DD     DSN=&LOAD,DISP=(OLD,DELETE),
170 //              DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
180 //        DD     DDNAME=SYSIN
190 //SYSUT1 DD     DSN=&S1,DISP=(OLD,DELETE)
200 //SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=3146,LRECL=121,RECFM=FBM)
210 //SYSLMOD DD DSN=&BZM(SMPO&N),DISP=OLD,UNIT=SYSDA,VOL=SER=&V
```

MEMBER NAME SMPOT

```
010 //SMPOT  PROC   PASM=XREF,
020 //              MAC=MACSMPO,BIM=BIMSMPO,
030 //              V=SMPO00
040 //ASM    EXEC   PGM=IUM90,PARM='NORLD,&PASM',REGION=200K
050 //STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR,VOL=SER=&V,UNIT=SYSDA
060 //SYSLIB DD     DSN=&MAC,DISP=SHR,UNIT=SYSDA,VOL=SER=&V
070 //        DD     DSN=SYS1.MACLIB,DISP=SHR
```

```
080 //SYSUT1 DD     DSN=&SYSUT1,UNIT=SYSDA,SPACE=(CYL,(2,1))
090 //SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=3509,LRECL=121,RECFM=FBM)
100 //SYSPUNCH DD  DUMMY
110 //SYSIN  DD     DSN=&BIM(&N),DISP=SHR,UNIT=SYSDA,VOL=SER=&V
MEMBER NAME  SMPOTRV
010 //SMPOTRV PROC PASM=NOXREF,PLKED=LIST,VEL=5,PR=5,R=150,
020 // MAC=MACSMPO,BIM=BIMSMPO,VR=BZMSMPO,BZM=BZMSMPO,
030 // T=900,
040 // V=SMPO00
050 //ASM    EXEC  PGM=IUM90,PARM='NOXREF,NORLD,NOESD,&PASM',
060 //              REGION=200K
070 //STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR,VOL=SER=&V,UNIT=SYSDA
080 //SYSLIB DD     DSN=&MAC,DISP=SHR,UNIT=SYSDA,VOL=SER=&V
090 //        DD     DSN=SYS1.MACLIB,DISP=SHR
100 //SYSUT1 DD DSN=&S1,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(2,1))
110 //SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=3509,LRECL=121,RECFM=FBM)
120 //SYSPUNCH DD  DSN=&LOAD,UNIT=SYSDA,SPACE=(3200,(20,20)),
130 //    DISP=(MOD,PASS),DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
140 //SYSIN  DD     DSN=&BIM(&N),DISP=SHR,UNIT=SYSDA,VOL=SER=&V
150 //LKED EXEC PGM=IEWL,PARM='SIZE=(120K,32K),LIST,MAP,&PLKED',
160 //              COND=((0,LT,ASM)),REGION=128K
170 //SYSLIB DD     DSN=&BZM,DISP=SHR,UNIT=SYSDA,VOL=SER=&V
180 //SYSLIN DD     DSN=&LOAD,DISP=(OLD,DELETE),
190 //              DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
200 //        DD     DDNAME=SYSIN
210 //SYSUT1 DD     DSN=&S1,DISP=(OLD,DELETE)
220 //SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=3146,LRECL=121,RECFM=FBM)
230 //SYSLMOD DD    DSN=&SYSLMOD(GO),SPACE=(1024,(50,20,1)),
240 //              DISP=(MOD,PASS),UNIT=SYSDA
250 //GO EXEC PGM=*.LKED.SYSLMOD,COND=((0,LT,ASM),(0,LT,LKED)),
```

```
260 //                REGION=&R.K,PARM='&R',TIME=&T
270 //STEPLIB DD  DSN=&VR,DISP=(SHR,PASS),UNIT=SYSDA,VOL=SER=&V
280 //        DD    DSN=&BZM,DISP=SHR,UNIT=SYSDA,VOL=SER=&V
290 //SYSPRINT DD  SYSOUT=A
300 //DISKSMPO DD  UNIT=SYSDA,
310 //              DSN=&SYSUT1,
320 //              SPACE=(CYL,(&VEL,&PR))
330 //DUMP   DD    SYSOUT=A
340 //SYSIN  DD    DUMMY,DCB=(BLKSIZE=80,BUFNO=2)
MEMBER NAME  SMPOV
010 //SMPOV PROC R=150,T=900,VR=BZMSMPO,
020 // VEL=100,PR=10,
030 // BZM=BZMSMPO,V=SMPO00
040 //GO EXEC PGM=SMPO&N,TIME=&T,REGION=&R.K
050 //STEPLIB DD DSN=&VR,DISP=(SHR,PASS),UNIT=SYSDA,VOL=SER=&V
060 //     DD DSN=&BZM,DISP=SHR,UNIT=SYSDA,VOL=SER=&V
070 //DISKSMPO DD UNIT=SYSDA,SPACE=(CYL,(&VEL,&PR)),
080 // DSN=&SYSUT1
090 //SYSPRINT DD SYSOUT=A
```

Приложение 4. Пример программы на языке СМПО

```
        PRINT NOGEN
LANC    STAL  (KN,B,M,N,N1,FLANC)
J       EQU   9
I       EQU   8
        TIMER
        TEKST 'CA  LANC  - МЕТОД ЛАНЦОША.'
        IR    1,2M
        ZR    1,(2QS,2QN,2RAB,2R,2QB)
        LH    1,N
        ZR    1,(2BETA,2ALFA,Q,2WS,2ZS)
        MATR  (ZS,QS,QN,R,RAB,WS,WN),(1,0)
        OPR   Q,(0,1),(ALFA,BETA),(1,0)
        ZAPOM (QS,QN,WS,WN,Q,ALFA,BETA,M,KN,R,RAB,ZS)
        SOZ   NM,ZS
        SOZ   NM,QS
        SOZ   NM,WS
        SOZ   NM,WN
        VP    PER,(B),R.          R=B.
        SR    J,J                 J=0.
CIKLJ   STH   J,IJ1+2             IJ1=(1,J-1)
        MVI   PR,X'00'            ПРИЗНАК:НЕ БЫЛО ОРТОГОНАЛ
        LA    J,1(J)              J=J+1
        CH    J,N                 ВЫПОЛНЕНЫ N ЭТАПОВ?
        BH    KONEC               ДА --> KONEC.
VBETA   VP    UMP,(M,R),RAB       RAB=M(*ПОЭЛЕМЕНТНОЕ)R.
        VP    UMT,(R,RAB),MAT11
        VP    IKP,(MAT11),MAT11   MAT11=BETAJ.
        GSTH  J,(IJ+2,IJ2)
```

```
        VPL   SBR,(MAT11,IJ),BETA  BETA(J)=BETAJ.
        MVC   BETAN,MAT111         BETAN=BETAJ.
        VP    DK,(R,BETAN),QN      Q(J)=R(J-1)/BETA'J).
        TM    PR,X'01'             БЫЛА ОРТОГОНАЛИЗАЦИЯ?
        BO    RSU                  ДА ->РЕШЕНИЕ СИСТЕМЫ УР-Й.
        CH    J,=H'2'              НОМЕР ЭТАПА?
        BM    RSU                  J=1? -> РЕШЕНИЕ С.У.
        BE    JRAV2                J=2? -> JRAV2
VCWN    SASP  1,WN,WS,BETA,ALFA,IJ2  ВЫЧИСЛЕНИЕ WN.
        V     MAXA,(WN),WMAX       WMAX=MAX(WN).
        LD    0,WMAX               ЕСЛИ WMAX<KAPPA,
        CD    0,KAPPA              ТО ПЕРЕЙДЕМ К RSU, ИНАЧЕ
        BM    RSU                  ОРТОГОНАЛИЗИРУЕМ R К Q .
ORTO    SASP  1,R,M,Q,IJ2
        SOZ   KM,WN,(KDEM15)       WN=10**15.
        OI    PR,X'01'             УСТАНОВИМ ПРИЗНАК:
        B     VBETA                БЫЛА ОРТОГОНАЛИЗАЦИЯ.
JRAV2   VP    UMP,(M,QS),R
        VP    UMT,(QN,R),MAT11     W(1)=QT(2)*M*Q(1).
        VPL   SBR,(MAT11,IJ1),WN   WN(1)=W(1)
        VPL   ZEL,(KD1,MAT11),WS   WS(1)=1.
RSU     VP    UMP,(M,QN),R         R=M*Q(J).
        VP    PER,(R),RAB          RAB=MQ(J).
SRSS    SASP  1,KN,R               РЕШИМ С.У.(K-σM)R(J)=MQ(J)
        VP    UMK,(QS,BETAN),QS
        VP    V,(R,QS),R           R(J)=R(J)-Q(J-1)BETA(J).
        VP    UMT,(R,RAB),MAT11    ALFA=R(J)T*M*Q(J).
        VPL   SBR,(MAT11,IJ),ALFA  ALFA(J)=ALFA.
        MVC   ALF,MAT111           ALFAJ=ALFA.
OPRQB   OPR   QB,(1,0)
```

```
        STEK   1
        VP     PER,(QN),QB          Q(J) - В МАТРИЦУ ВСЕХ Q.
        ZBL    QB,(IJ2),Q
        VP     UMK,(QN,ALF),RAB     RAB=QN*ALFA.
        VP     V,(R,RAB),R          R=R-QN*ALFA.
        VP     PER,(QN),QS          QS=QN.
        CH     J,=H'1'              ПЕРВЫЙ ЭТАП?
        BE     CIKLJ                ДА -> НЕ ВЫЧИСЛЯЕМ Z.
        ZR     J,(2D,2E)
        MATR   (D,E),(1,0)          ОПРЕДЕЛЯЕМ D И E.
        ZAPOM  (D,E)
        VP     RAZ,(ALFA,MAT11),D   D=(J ЭЛЕМЕНТОВ) ИЗ ALFA.
        VP     RAZ,(BETA,MAT11),E   E=(J ЭЛЕМЕНТОВ) ИЗ BETA.
FQLZ    SPFOR  (Z)                  РЕШАЕМ СОБСТВ. ПРОБЛЕМУ:
QLZ     SASP   1,D,E,FQLZ           QL-АЛГОРИТМ.
        V      SOV,(Z,ZS,KAPPA),RSR СКОЛЬКО ФОРМ СОШЛОСЬ?
        CLC    RSR(2),N1            ЕСЛИ НЕ МЕНЬШЕ ЗАДАННОГО
        BNM    SOKF                 КОЛИЧЕСТВА, ТО ->SOKF.
        VPL    SBR,(Z,MAT11),ZS
        UDAL   1                    УДАЛИМ D И E.
        KBL    1                    УНИЧТОЖИМ Z.
        B      CIKLJ
SOKF    TEKST  'СОШЛОСЬ ЗАДАННОЕ К-ВО ФОРМ!'
KONEC   UDAL   1,(Q,ALFA,BETA)
        SPFAK  (Q,ALFA,BETA),FLANC
        TIMER
        KSTAL
D       GDS    (KN,B,M,BETAN,WMAX,RSR,ALF)
MAT11   SMATR  (1,1),(0)
KAPPA   DC     D'1.E-8'
```

```
KD1       DC    D'1'
KDEM15    DC    D'1.E-15'
H         GDS   (N,N1)
IJ1       DC    H'1,0'
IJ2       DC    H'0,1'
IJ        DC    H'1,0'
PR        DS    X
          KSTP
          END
```

Список литературы

1. **Автоматизированные** системы расчета на прочность конструкций летательных аппаратов (по материалам иностранной печати): Обзоры. Переводы. Рефераты/Составители Г. А. Косушкин, Г. В. Жебракова. ЦАГИ, 1979, № 564. 79 с.

2. **Агнистиков И. М.** Создание пакетов прикладных программ для реализации матричных алгоритмов и расчета фюзеляжа вертолета методом конечных элементов на ЕС ЭВМ: Дис... канд. техн. наук. М., 1983. 161 с.

3. **Агнистиков И. М., Зархин Б. Я.** Распараллеливание вычислительного процесса в суперэлементном методе сил//Вопросы прочности, устойчивости и колебаний конструкций летательных аппаратов: Межвузовский сб. Казань: КАИ, 1985. С. 77—80.

4. **Бастраков С. М., Денисов Ю. А., Попов Ю. Г.** К расчету на прочность авиационных конструкций шаговым методом с применением метода конечных элементов//ИВУЗ, Авиационная техника. 1983, № 4. С. 9—14.

5. **Бате К., Вилсон Е.** Численные методы анализа и метод конечных элементов/Пер. с англ. А. С. Алексеева и др.; Под ред. А. Ф. Смирнова. М.: Стройиздат, 1982. 448 с.

6. **В. П. Борисов, З. И. Бурман, Р. А. Шакирзянов.** Расчет на собственные колебания свободных конструкций с использованием суперэлементного метода сил/КИСИ. Казань, 1982. 25 с. Деп. в ВИНИТИ 20.08.1982, № 2019—82.

7. **Браун П.** Обзор макропроцессоров/Пер. с англ. под ред. Э. З. Любимского. М.: Статистика, 1975. 80 с.

8. **Бурман З. И., Десятник Г. А.** Применение суперэлементного смешанного метода для расчета комбинированных конструкций//IX Internationaler Kongress über Anwendungen der Mathematik in den Ingenierwissenhaften, Berichte 4. Weimar, 1981. S. 17—19.

9. **Бурман З. И., Десятник Г. А.** Расчет на прочность авиационных конструкций смешанным суперэлементным методом//ИВУЗ. Авиационная техника, 1984, № 2. С. 89—91.

10. **Бурман З. И., Лукашенко В. И., Тимофеев М. Т.** Расчет тонкостенных подкрепленных оболочек методом конечных элементов с применением ЭЦВМ. Казань: Изд. КГУ, 1973. 569 с.

11. **Бурман З. И.** Решение задачи кручения стержня произвольной формы нерегулярного характера в матричном виде с помощью ЭЦВМ//Труды КИСИ, вып. X. Казань: Изд. КИСИ. 1967. С. 15—28.

12. **Бурман З. И., Тимофеев М. Т.** Математическое обеспечение для матричных расчетов тонкостенных пространственных конструкций с применением МКЭ// Вопросы оптимального использования ЭЦВМ в расчете сложных конструкций. Казань: Изд. КГУ, 1973. С. 87—94.

13. **Вахитов М. Б., Сафариев М. С., Снигирев В. Ф.** Расчет крыльевых устройств судов на прочность. Казань: Татарское книжное издательство, 1975. 212 с.

14. **Галлагер Р.** Метод конечных элементов. Основы/Пер. с англ. В. М. Картвелишвили под ред. Н. В. Баничука. М.: Мир, 1984. 428 с.

15. **Гаян Р.** Приведение матрицы жесткости и масс//Ракетная техника и космонавтика. 1965. № 2. Т. 3, С. 287.

16. **Джермейн К.** Программирование на IBM/360/Пер. с англ. под ред. В. С. Штаркмана. М.: Мир, 1971. 432 с.

17. **Джорж А., Лю Дж.** Численное решение больших разреженных систем уравнений/Пер. с англ. Х. Д. Икрамова. М.: Мир, 1984. 333 с.

18. **Ерунов Б. Г., Фатхуллин Р. Р.** Висячие мосты общего типа//Вопросы надежности мостовых конструкций. Л.: Изд. ЛИСИ, 1984. С. 95—101.

19. **Землянский А. А., Персиц М. Г.** Основы операционной системы ЕС ЭВМ. М.: Сов. радио, 1980. 144 с.

20. **Йодан Э.** Структурное проектирование и конструирование программ/Пер. с англ. В. В. Фролова и Л. А. Топчеева; Под ред. Л. Н. Королева. М.: Мир, 1979. 415 с.

21. **Кнут Д.** Искусство программирования для ЭВМ/Пер. с англ.; Под ред К. И. Бабенко. М.: Мир, 1977. 724 с.

22. **К вопросу** расчета тонкостенных конструкций шаговым методом с применением метода конечных элементов/С. М. Бастраков, Ю. А. Денисов, А. Ф. Кучинский, Ю. Г. Попов. КАИ. Казань, 1982. 12 с. Деп. в ВИНИТИ 13.05.1982, № 5709—82.

23. **Майерс Г.** Надежность программного обеспечения/Пер. с англ. Ю. Ю. Галимова под ред. В. Ш. Кауфмана. М.: Мир, 1980. 360 с.

24. **Мартин Дж.** Организация баз данных в вычислительных системах/Пер. с англ.; Под ред. А. А. Стогния, А. Л. Щерса. М.: Мир, 1980. 662 с.

25. **Метод** конечных элементов в механике твердых тел/Под общей ред. А. С. Сахарова и И. Альтенбаха. Киев: Віща школа, 1982. 480 с.

26. **Метод** конечных элементов в проектировании транспортных сооружений/ А. С. Городецкий, В. И. Зоворoцкий, А. И. Лантух-Лященко, А. О. Рассказов. М.: Транспорт, 1981. 143 с.

27. **Образцов И. Ф., Савельев Л. М., Хазанов Х. С.** Метод конечных элементов в задачах строительной механики летательных аппаратов. М.: Высшая школа, 1985. 392 с.

28. **Операционная** система ОС ЕС: Справочное пособие/В. П. Данилочкин, В. В. Митрофанов, Б. В. Одинцов, Г. В. Пеледов; Под ред. Л. Д. Райкова. М.: Статистика, 1980. 480 с.

29. **Парлетт Б.** Симметричная проблема собственных значений. Численные методы/Пер. с англ. Х. Д. Икрамова и Ю. А. Кузнецова. М.: Мир, 1983. 382 с.

30. **Райс Дж.** Матричные вычисления и математическое обеспечение/Пер. с англ. О. Б. Арушаняна; Под редакцией В. В. Воеводина. М.: Мир, 1984. 264 с.

31. **Расчет** пространственной стержневой виброизолированной конструкции при кинематическом возбуждении/Н. А. Пикулев, Г. А. Десятник, Г. Н. Шмелев. И. А. Шаихов/Расчет и испытание металлических конструкций. Казань: Изд КИСИ, 1982. С. 52—54.

32. **Розин Л. А.** Вариационные постановки задач для упругих систем. Л.: Изд. ЛГУ, 1978. 224 с.

33. **Современные** методы расчета сложных статически неопределимых систем: Сборник статей/Под ред. А. П. Филина. Л.: Судпромгиз, 1961. 876 с.

34. **Старокомодская З. М., Тепеницын М. П.** Исследование прочности треугольных крыльев на основе дискретной расчетной схемы с применением метода сил//Труды ЦАГИ, Вып. 1119. 1969. 47 с.

35. **Суперэлементный** расчет подкрепленных оболочек/З. И. Бурман, О. М. Аксенов, В. И. Лукашенко, М. Т. Тимофеев. М.: Машиностроение, 1982. 256 с.

36. **Уилкинсон, Райнш.** Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра/Пер. с англ.; Под ред. Ю. И. Топчеева. М.: Машиностроение, 1976. 389 с

37. **Филин А. П.** Матрицы в статике стержневых систем. Л.: Стройиздат, 1966. 438 с.

38. **Хусаинов Б. С.** Макросредства в языке ассемблера ЕС ЭВМ. М.: Статистика, 1978. 94 с.

39. **Allman.** A compatible triangular element including verfex rotations for plane elasticity analysis//Computers and Structures. 1984. V. 19. P. 1—8.

40. **Arora J. S., Nguyen D. T.** Eigensolution for large structural systems with substructures//Int. J. Num. Meth. in Eng. 1980. No 3. P. 333—341.

41. **Argiris I. H., Kelsey S.** Modern fuselage analysis and the elastic aircraft. Basic theory. London: Butterworths, 1963. P. 176.

42. **Batoz J.—L., Bathe K.—J., Ho L. W.** A study of threenode triangular plate bending elements//Int. Journal for Numer. Meth. in Eng. 1980. V. 15. No 12. P. 1771—1812.

43. **Dankert J., Limpert H.** Ein System zur Verwaltung grober Datenmengen// Rechentechnik und Datenverarbeitung, Beihelf 1, 1977.

44. **Ein Matrizeninterpretationssystem** der Strukturmechanik für Praxis, Forschung und Lehre/Krätzig W. B., Metz H., Schmid G., Weber B. Techn. Wiss. Mitt. No 77—5.

45. **Gabbert U., Berger H.** Universelles FEM Programmsystem COSAR — Leustungsumfang, Anvendungserfahrungen, Weiterentwicklungen//X Internationalen Kongress über Anwendungen der Mathematik in den Ingenieurwissenschaften, Berichte 5.— Weimar, 1984. P. 70—74.

46. **Hybrid** and mixed finite element methods/Eds. SN. Atluri, R. H. Gallagher, O. C. Zienkiewicz. New York: Wiley, 1983. 582 p.

47. **IBM** Application Program System/360. Matrix Language (MATLAN) (360—cm—05x): Program Description Manual. IBM, 1968. 360 p.

48. **Jachimowicz J.** Zastosowanil macierzowej metody sil w obliczeniach konstrukcji lotniczycn//Prace instytutu lotnictwa, v. V9: Wydawnictva przemyslu lotniczego. 1982. P. 61—91.

49. **Kaneko J., Lawo M., Thierauf G.** On computational procedures for the force method//International journal for numerical methods in engineering. V. 18. 1982. P. 1469—1495.

50. **Meirovitch L., Hale A. L.** On the substructure sinthesis method//AIAA Journ. 1981. V. 19. No 7. P. 940—947.

51. **Nagy D. A.** Software engineering for finite element analysis//Journal of the structural division/Proc. ASCE. V. 104, NST8. August, 1978. P. 1287—1298.

52. **Nour-Omid B., Clough R.** Dynamic analysis of structures using Lanczos co-ordinates//Earthquake engineering and structural dynamics. 1984. V. 12. P. 565—577.

53. **Preconditioning methods:** analysis and applications/Ed. D. J. Evans. No 4, 1983. XII, 556 p.

54. **Przemieniecki I. S.** Theory of Matrix Structural Analysis. — N. Y.: McGraw-Hill Book Company, 1968. 468 p.

55. **Robinson I.** Integrated Theory of Finite Element Methods. New York: Wiley, 1973. II+428 p.

56. **Robinson I.** Structural Matrix Analysis for the Engineer. New York: Wiley, 1966. 344 p.

57. **Robinson I.** Understanding finite element stress analysis. Wimborn (England): Robinson and Associates, 1981. 405 p.

58. **Smith I. M.** An applications of software library for finite element analysis// Eng. Software III. Proc., 3rd Int. Conf., London, 11—13 Apr. 1983. Berlin e. a., 1983. P. 899—905.

59. **Stolarski H., Belytschko T., Carpenter N., Kennedy I. M.** A simple triangular curved shell element//Eng. Comput. 1984. V. 1.— September. P. 210—218.

60. **Wezyk W., Budziszenski T.** MES — pewne problemy modelowania konstrukcji//Biul inf. przem. motoryz. — 1980. No 3—4. P. 24—28

Оглавление

ПРОИЗВОДСТВЕННОЕ ИЗДАНИЕ

Бурман Залман Иосифович, Артюхин Георгий Алексеевич, Зархин Борис Яковлевич

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МАТРИЧНЫХ АЛГОРИТМОВ И МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ В ИНЖЕНЕРНЫХ РАСЧЕТАХ

Редактор *Е. И. Черных*
Художественный редактор *В. В. Лебедев*
Переплет художника *Е. Н. Волкова*
Технический редактор *Л. А. Макарова*
Корректор *Л. А. Ягупьева*

ИБ № 5486

Сдано в набор 20.10.87. Подписано в печать 22.03.88. Т-05163.
Формат 60×88 1/16. Бумага офсетная № 2. Гарнитура литературная.
Печать офсетная. Усл. печ. л. 15,68. Усл. кр.-отт. 15,68. Уч.-изд. л. 16,75.
Тираж 14 600 экз. Заказ 1681. Цена 1 р. 30 к.

Ордена Трудового Красного Знамени издательство «Машиностроение».
107076, Москва, Стромынский пер., 4

Московская типография № 8 Союзполиграфпрома
при Государственном комитете СССР по делам издательств,
полиграфии и книжной торговли,
101898, Москва, Центр, Хохловский пер., 7

УВАЖАЕМЫЕ ТОВАРИЩИ!

В 1989 году издательство «Машиностроение» выпустит в свет следующие книги:

Вольмир А. С., Куранов Б. А., Турбаивский А. Т. **Статика и динамика сложных структур:** Прикладные многоуровневые методы исследований. 18 л.

Изложены многоуровневые методы расчета современных тонкостенных конструкций. Основное внимание уделено методу конечных элементов и вопросам рационального сочетания его с другими численными методами. Рассмотрены особенности построения расчетных моделей и формирования систем разрешающих уравнений, метод решения задач линейного и нелинейного деформирования, устойчивости, свободных и вынужденных колебаний. Показаны специфика реализации многоуровневых методов и их эффективность. Описан комплекс программ расчета оболочечных конструкций.

Для инженеров, занятых расчетом тонкостенных конструкций.

Методы динамических расчетов и испытаний тонкостенных конструкций/ *А. В. Кармишин, Э. Д. Скурлатов, В. Г. Старцев и др.* 17 л.

Изложены подходы к описанию напряженно-деформированного состояния неоднородных пластин и оболочек, методы расчета тонкостенных конструкций на ударные нагрузки и способы экспериментального и экспериментально-теоретического исследования прочности и устойчивости этих конструкций, а также конструкций, содержащих во внутренней полости жидкость, пористый заполнитель и жесткие присоединенные массы. Особое внимание уделено динамическому поведению конструкций, подвергающихся многократному комбинированному статико-динамическому нагружению.

Для инженеров, занятых расчетом, разработкой и испытанием тонкостенных конструкций.

Методы оптимизации авиационных конструкций/*В. М. Фролов, В. И. Бирюк, Ю. Ф. Яремчук и др.* 20 л.

Изложены современные методы оптимизации силовых конструкций летательных аппаратов (ЛА); даны расчетные модели, применяемые в оптимизации авиационных конструкций. Сформулированы задачи весовой оптимизации силовых конструкций на основе взаимной увязки требований прочности и аэроупругости, и дано их решение. Расчетные методы проиллюстрированы конкретными примерами проектирования конструктивно-силовых схем ЛА. Даны рекомендации по выбору наиболее рациональных конструктивно-силовых схем.

Для инженеров, занимающихся расчетом и проектированием авиационных конструкций.