СИНТЕЗ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ УПРАВЛЕНИЯ И КОНТРОЛЯ

СИНТЕЗ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ УПРАВЛЕНИЯ И КОНТРОЛЯ

ИЗДАТЕЛЬСТВО «ТЕХНІКА»
Киев — 1975

6Ф7.3
С38

УДК 681.142.1

Синтез вычислительных алгоритмов управления и контроля. Кузьмин И. В., Березюк Н. Т., Фурманов К. К., Шаронов В. Б. «Техніка», 1975, 248 стр.

Рассмотрены методы разработки оптимальных алгоритмов управляющих вычислительных систем (УВС), используемых в сложных системах автоматизированного управления и контроля. Изложены критерии эффективной реализации алгоритмов на УВС, теория оптимизации алгоритмов при перечислении их параметров эффективности в виде кортежей и матриц, а также основы диспетчеризации алгоритмов. Описаны принципы построения высокопроизводительных УВС с одним вычислителем и множеством входящих информационных потоков, с одним информационным потоком и множеством вычислителей и с множеством входящих потоков и множеством вычислителей. Предназначена для инженеров, работающих в области проектирования и эксплуатации сложных систем автоматизированного управления и контроля, а также может быть полезна студентам вузов соответствующих специальностей.

Табл. 12, илл. 30, библ. 168.

Рецензент *В. В. Иванов,* докт. физ.-мат. наук

Редакция литературы по энергетике, электронике, кибернетике и связи

Заведующий редакцией инж. *З. В. Божко*

С $\frac{30502 - 008}{\text{М 202 (04)-75}}$ 51-75

ПРЕДИСЛОВИЕ

В решениях XXIV съезда КПСС отмечено, что в эру научно-технического прогресса, являющегося главным рычагом поднятия производительных сил и создания материально-технической базы коммунизма, необходимо органически соединить достижения научно-технической революции с преимуществами социалистической системы ведения хозяйства. Эту проблему возможно решить путем широкого применения автоматизированных систем управления и контроля. К таким системам в первую очередь следует отнести автоматизированные системы управления предприятиями, отраслями и народным хозяйством в целом.

Все эти системы имеют не только множество сложных устройств и органов, но и высокий уровень организации, сложные функциональные взаимосвязи устройств и органов.

При проектировании автоматизированных систем управления и контроля (АСКУ) на базе электронных цифровых вычислительных машин (ЭЦВМ) существенным является разработка алгоритмов функционирования и создания развитого математического обеспечения. Совершенное математическое обеспечение позволит облегчить решение задач управления и контроля сложных объектов и решить проблему работы оператора в режиме диалога с ЭЦВМ.

Одной из новых проблем современной науки является разработка и внедрение методов исследования функционирования сложных систем. В последние годы интенсивно

развиваются новые методы теории алгоритмов и алгоритмического описания процессов функционирования сложных систем, теории специальных видов случайных процессов и теория массового обслуживания [32].

Задачам оценки эффективности, оптимизации и разработки АСКУ посвящено много работ. В связи с необходимостью создания развитого математического обеспечения разработка основ синтеза вычислительных алгоритмов контроля и управления является неотложной задачей. В данной книге изложены основы теории обоснования критериев оценки эффективности и оптимизации вычислительных алгоритмов, их разработки и реализации на простейших и сложных вычислительных системах.

Авторы выражают благодарность докт. физ.-мат. наук В. В. Иванову, сделавшему ряд замечаний, позволивших улучшить содержание книги.

Отзывы и пожелания просим направлять по адресу: *252601, Киев, 1, ГСП, Пушкинская, 28, издательство «Техніка».*

Глава I

КРИТЕРИИ ЭФФЕКТИВНОЙ РЕАЛИЗАЦИИ АЛГОРИТМОВ УПРАВЛЯЮЩИМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ

1. КРИТЕРИИ ТИПА ПЕРЕЧИСЛЕНИЯ

Эффективность вычислительных алгоритмов может быть оценена перечислением их частных показателей [21]. При этом каждый частный показатель характеризуется количественной оценкой, физическим смыслом и зависит от технических параметров информационной системы. Вместе с тем система должна быть критичной к каждому из этих показателей. Общее число частных показателей не должно быть большим, так как усложняется задача оптимизации выбора наилучших алгоритмов.

Такой подход обусловлен тем, что единые показатели или критерии типа отношений при оценке алгоритмов не всегда приводят к эффективному использованию возможностей системы, хотя при этом упрощается процесс оптимизации [51].

В качестве основных параметров, характеризующих эффективность алгоритмов, выбираем следующие:

погрешность, вносимую самим алгоритмом;

объем памяти, который необходим для алгоритма с учетом констант;

время реализации УВС программы алгоритма.

Таким образом, эффективность алгоритма является функцией в общем случае некоторого числа параметров x_ρ ($\rho = 1, 2, \ldots, \eta$), совокупность которых может быть представлена в виде многомерного вектора или кортежа

$$X = \langle x_1, x_2, \ldots, x_\eta \rangle. \qquad (1.1)$$

Параметры x_ρ могут находиться как в согласии, так и в противоречии. Так, например, увеличение длины программы без разветвлений неизбежно ведет к увеличению времени ее реализации (согласие параметров). Однако эти же параметры находятся в противоречии для программ,

у которых среднее время реализации уменьшается за счет увеличения числа разветвлений (увеличение общей длины программы). Как правило, во взаимном противоречии находятся такие параметры, как погрешность и время реализации. При этом уменьшение погрешности вычислений связано или с выбором более сложного вычислительного метода, требующего большего объема вычислений, или с увеличением разрядности представляемых в машине чисел. Как тот, так и другой способ уменьшения погрешности приводит к увеличению времени реализации. Во взаимном противоречии с временем реализации и объемом программы находятся также параметры, учитывающие объем и характер операций, входящих в систему команд машины. Естественно, что расширенная система команд позволяет создавать программы более простыми и с малым временем реализации.

Приведенные выше примеры взаимосвязей разнозначных параметров показывают, что для одной и той же задачи можно составить набор программ с различными показателями эффективности их реализации на УВС. Выбор той или иной программы зависит от многих факторов и, в частности, от требований, наложенных на частные показатели x_ρ алгоритма управления.

Если эффективность алгоритма управления описывается в виде перечисления некоторых параметров (кортежа), то, очевидно, что и любые выделенные из него элементарные участки также можно описать в виде кортежей (1.1), между однозначными параметрами которых существует вполне определенная связь, позволяющая определить компоненты кортежа алгоритма управления. Таким образом, если известны кортежи элементарных участков, то задача определения кортежа алгоритма управления принимает единственное решение.

Однако, если для каждого выделенного различного участка составить набор алгоритмов (программ) с различной эффективностью, то эта задача будет иметь множество решений. Прежде чем приступить к процессу оптимизации, необходимо изучить характер описания отдельных элементарных участков кортежами, взаимосвязь однозначных параметров между кортежами различных участков, изменения значения компонент с учетом динамики (многократной повторяемости) некоторых выделенных участков в алгоритме управления.

Введем некоторые понятия, которыми будем пользоваться при рассмотрении данной задачи.

Любой выделенный элементарный участок из более сложного представляет собой алгоритм (программу), который в дальнейшем будем называть простым. Необходимым условием простых алгоритмов является то, что их эффективности известны и представлены в виде кортежей. Эффективность сложного алгоритма определяется через компоненты кортежей простых алгоритмов, над которыми проводятся арифметико-логические операции. Полученный кортеж позволяет теперь рассматривать сложный алгоритм как простой, так как его эффективность определена.

В дальнейшем рассматриваются следующие показатели эффективности алгоритмов:

погрешность Δ, вносимая алгоритмом в ходе вычислений;

длина программы d, определяемая числом команд в программе алгоритма;

время реализации t, определяемое через суммарное время реализации каждой команды в программе алгоритма;

константы, необходимые для реализации алгоритма, перечисляются в множестве

$$K = \{k_1, k_2, \ldots, k_\lambda\},$$

а их число равно некоторой величине λ.

Систему команд, необходимую для реализации алгоритма, запишем в виде множества

$$G = \{g_1, g_2, \ldots, g_\vartheta\},$$

количество компонент которого задается величиной ϑ.

Определим межкортежные и внутрикортежные зависимости над этими параметрами. Рассмотрим некоторый сложный алгоритм B, состоящий из ряда простых алгоритмов A_j ($j = 1, 2, \ldots, m$). При этом алгоритм B не содержит циклов и разветвлений, а его составляющие алгоритмы A_j повторяются в B только по одному разу.

Если кортежи $X(A_j)$ простых алгоритмов известны, следуют один за другим согласно операторам вычислительного процесса вне цикла и составляют некоторый более общий алгоритм B, то длина программы d_B алгоритма B

$$d_B = \sum_{j=1}^{m} d_j, \tag{1.2}$$

где d_j — длина программы алгоритма A_j.

Время реализации алгоритма B

$$t_B = \sum_{j=1}^{m} t_j, \tag{1.3}$$

где t_j — время реализации алгоритма A_j.

Если константы алгоритма A_j перечислены в множестве

$$K_j = \{k_{j\mu}\} \qquad (\mu = 1, 2, \ldots, \lambda_j),$$

а λ_j — общее число констант, то для определения множества констант K_B алгоритма B следует применить операцию соединения множеств K_j, так как одинаковые константы различных простых алгоритмов нецелесообразно хранить в разных ячейках памяти. Тогда после операции соединения

$$K_B = \bigcup_{j=1}^{m} K_j \tag{1.4}$$

получим множество

$$K_B = \{k_\mu\} \qquad (\mu = 1, 2, \ldots, \lambda_B),$$

где λ_B — число констант алгоритма B.

Если УВС имеет единую долговременную память для хранения как программ управления, так и констант, то важным показателем эффективности алгоритмов для таких УВС является компонента кортежа, учитывающая общий объем памяти под программу и константы. Для каждого алгоритма A_j этот показатель определяется как

$$D_j = d_j + \lambda_j,$$

а для алгоритма B определяется после операций (1.2) и (1.4) и принимает значение

$$D_B = d_B + \lambda_B.$$

Систему команд G_B алгоритма B находим через операцию соединения систем команд G_j простых составляющих его алгоритмов.

Если

$$G_j = \{g_{j\nu}\} \qquad (\nu = 1, 2, \ldots, \vartheta_j),$$

то

$$G_B = \bigcup_{j=1}^{m} G_j. \tag{1.5}$$

После операции (1.5) получим

$$G_B = \{g_\nu\} \qquad (\nu = 1, 2, \ldots, \vartheta_B),$$

где ϑ_B — общее число кодов операций алгоритма B,

Величина ϑ_j может входить отдельным параметром в кортеж j-го алгоритма. Это объясняется тем, что общее число кодов операций алгоритма управления не должно превышать заданного числа, которое определяется дешифратором команд УВС.

Кроме G_j и ϑ_j в кортеж j-го алгоритма могут входить дополнительные параметры, определяющие, например, стоимость или затраты оборудования на внедрение множества команд G_j. Как ϑ_j, так и дополнительные параметры, характеризующие систему команд, могут быть вычислены только после операции соединения.

Погрешность Δ_B представляет собой сложную функцию от параметров Δ_j. Для ряда частных задач Δ_B может быть выражено через некоторую функцию

$$\Delta_B = \delta(\Delta_1, \Delta_2, \ldots, \Delta_m).$$

Таким образом, для определения эффективности алгоритма управления необходимо предварительно выбрать его частные показатели. При выборе этих показателей можно руководствоваться техническими требованиями к УВС. По установленным внутрикортежным зависимостям (как это было для λ_B, D_B и ϑ_B) и межкортежным связям (1.2) — (1.5) по известным простым кортежам $X(A_j)$ можно получить эффективность сложного алгоритма

$$X(B) = \langle x_{1B}, x_{2B}, \ldots, x_{\eta B} \rangle.$$

Рассмотрим оценку эффективности сложного алгоритма B, в котором простые алгоритмы A_j повторяются многократно, а сам алгоритм B не содержит циклов и разветвлений.

Компоненты сложного кортежа X_B здесь определяются так же, как и ранее. Однако для упрощения вычислений все однотипные A_j можно совместить в один.

Например, пусть алгоритм B описывается последовательностью (операторной схемой) $A_1A_2A_3A_1A_2A_1$. Совмещая однотипные алгоритмы в один A_j, где R_j — ранг совмещения (указывает на число совмещений), получаем $A_{\underset{3}{1}}A_{\underset{2}{2}}A_{\underset{1}{3}}$.

Ранг алгоритма A_3 равен единице, так как A_3 один раз встречается в алгоритме B. Такое совмещение нарушает операторную схему, однако является удобным для определения рангов простых алгоритмов.

Кортеж $X\underset{R_j}{(A_j)}$ любого совмещенного алгоритма определяется через произведение ранга на кортеж исходного простого

$$X\underset{R_j}{(A_j)} = R_j X(A_j) = R_j < x_{1j}, x_{2j}, \ldots, x_{\eta j} > =$$

$$= < x_{1j}^0, x_{2j}^0, \ldots, x_{\eta j}^0 >.$$

В данном случае величина R_j умножается только на те компоненты, которые изменяют свои численные значения, если простой алгоритм повторяется многократно. Например, показатели длины программы и времени реализации увеличиваются в R_j раз, однако константы и система команд останутся прежними, несмотря на многократное повторение A_j:

$$\left.\begin{aligned} d_j^0 &= R_j d_j; \\ t_j^0 &= R_j t_j; \\ K_j^0 &= K_j; \\ G_j^0 &= G_j. \end{aligned}\right\} \qquad (1.6)$$

Если все многократно повторяемые простые алгоритмы представить подобным образом, то вновь полученные их кортежи можно условно считать простыми. Теперь сложный алгоритм B рассматривается как состоящий из простых A_j, повторяющихся в B по одному разу, оценку эффективности которого рассматривали ранее. Такими многократно повторяющимися алгоритмами являются также программы входа в подпрограмму, у которых однозначные компоненты кортежей совпадают.

Рассмотрим случай, когда алгоритм A_j повторяется в цикле N_j раз. Считаем, что простой кортеж $X(A_j)$ известен. Однако с ним оперировать нельзя, так как он не учитывает цикличность A_j. Для того чтобы его можно было считать условно простым, необходимо вычислить те параметры, которые изменяются в процессе циклов. Умножив N_j на кортеж $X\underset{N_j}{(A_j)}$, получим для нового совмещенного алгоритма A_j со степенью цикличности N_j кортеж

$$X\overset{N_j}{(A_j)} = N_j X(A_j) = N_j < x_{1j}, x_{2j}, \ldots, x_{\eta j} > =$$

$$= < x_{1j}^0, x_{2j}^0, \ldots, x_{\eta j}^0 >.$$

Для ранее приведенных показателей очевидно следующее:

$$\left.\begin{aligned} d_j^0 &= d_j; \\ t_j^0 &= N_j t_j; \\ K_j^0 &= K_j; \\ G_j^0 &= G_j. \end{aligned}\right\} \tag{1.7}$$

Соотношения (1.7) справедливы также для кортежей подпрограмм. Если алгоритм A_j охвачен несколькими ветвями циклов (цикл в цикле), при этом ψ-я ветвь цикла ($\psi = 1, 2, \ldots, \Psi$) содержит N_ψ циклов, то система преобразований компонент простого кортежа в условный простой будет иметь вид:

$$\left.\begin{aligned} d_j^0 &= d_j; \\ t_j^0 &= N_1 N_2 \ldots N_\Psi t_j = t_j \prod_{\psi=1}^{\Psi} N_\psi; \\ K_j^0 &= K_j; \\ G_j^0 &= G_j. \end{aligned}\right\} \tag{1.8}$$

В общем случае алгоритм A_j может встречаться в операторной схеме r_j раз вне циклов, а в простых и сложных циклах — Φ_j раз. Пусть φ_j-й алгоритм ($\varphi_j = 1, 2, \ldots, \Phi_j$) находится в Ψ_{φ_j} циклах в цикле ($\psi_{\varphi_j} = 1, 2, \ldots, \Psi_{\varphi_j}$). При этом ψ_{φ_j}-я ветвь содержит $N_{\psi_{\varphi_j}}$ циклов. Тогда

$$\left.\begin{aligned} d_j^0 &= r_j d_j + \Phi_j d_j = (r_j + \Phi_j)\, d_j; \\ t_j^0 &= r_j t_j + t_j \prod_{\psi=1}^{\Psi_1} N_{\psi_1} + t_j \prod_{\psi=1}^{\Psi_2} N_{\psi_2} + \cdots + \\ &+ t_j \prod_{\psi=1}^{\Psi_{\Phi_j}} N_{\psi_{\Phi_j}} = \left(r_j + \sum_{\varphi_j=1}^{\Phi_j} \prod_{\psi=1}^{\Psi_{\varphi_j}} N_{\psi_{\varphi_j}}\right) t_j; \\ K_j^0 &= K_j; \\ G_j^0 &= G_j. \end{aligned}\right\} \tag{1.9}$$

В приведенной системе ранг и степень цикличности

j-го алгоритма соответственно равны:

$$\left.\begin{aligned} R_j &= r_j + \Phi_j; \\ N_j &= r_j + \sum_{\varphi_j=1}^{\Phi_j} \prod_{\psi=1}^{\Psi_{\varphi_j}} N_{\psi\varphi_j}. \end{aligned}\right\} \tag{1.10}$$

При $\Phi_j = 0$ система (1.9) преобразуется в систему (1.6). При $r_j = 0$ и $\Phi_j = 1$ получаем систему (1.8).

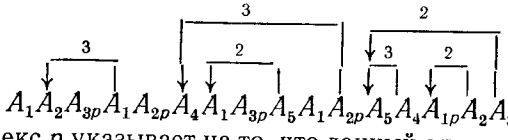
В дальнейшем алгоритм A_j с рангом R_j и степенью цикличности N_j будем называть совмещенным и обозначать

$$\overset{N_j}{\underset{R_j}{A_j}} \tag{1.11}$$

Компоненты такого алгоритма получаются из системы (1.9). В частном случае при $N_j = 1$ и $R_j = 1$ имеем простой алгоритм $\overset{1}{\underset{1}{A_j}}$.

После совмещения всех однотипных алгоритмов и определения по системе (1.9) их кортежей, приступают к оценке эффективности алгоритма B. Для этого используют выражения (1.2) — (1.5). Численные значения ранга R_j и степени цикличности N_j легко получить из операторной схемы алгоритма B. Для этого над операторной схемой необходимо провести несложные преобразования.

Пусть алгоритм B имеет следующую операторную схему:



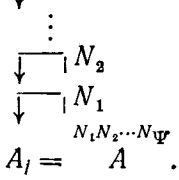
(1.12)

где индекс p указывает на то, что данный алгоритм выполнен в виде подпрограммы; стрелка охватывает ряд операторов, находящихся в цикле; цифра над стрелкой обозначает количество циклов.

Определим степень цикличности для каждого оператора, охваченного несколькими циклами. В соответствии с системой (1.8) для t_j, если некоторый алгоритм A_j охвачен Ψ циклами с числом циклов $N_1, N_2, \ldots, N_\Psi$, то его степень цикличности

$$N_{A_j} = \prod_{\psi=1}^{\Psi} N_\psi,$$

т е. преобразование операторной схемы для A_j имеет вид:



$$A_j = A^{N_1 N_2 \cdots N_\Psi}.$$

Перепишем выражение (1.12) в соответствии с обозначением (1.11), тогда получим

$$A_1^{1}A_2^{3}A_{3p}^{3}A_1^{3}A_{2p}^{1}A_4^{3}A_1^{6}A_{3p}^{6}A_5^{6}A_1^{3}A_{2p}^{3}A_5^{6}A_4^{6}A_{1p}^{4}A_2^{4}A_{3p}^{2}A_3^{1}. \quad (1.13)$$

Сложный алгоритм B состоит из пяти разнотипных простых алгоритмов A_1, A_2, A_3, A_4, A_5. Определим их ранги и степени цикличности. Для этого совместим однотипные алгоритмы в формуле (1.13), при этом воспользуемся следующими правилами определения рангов и степени цикличности.

Совмещение двух или более однотипных алгоритмов дает новый совмещенный, ранг и степень которого определяются суммой рангов и степеней составляющих его алгоритмов.

Например, совмещение алгоритмов

$$A_j^{N_1}, \quad A_j^{N_2} \text{ и } A_j^{N_3}$$
$$R_1 \qquad R_2 \qquad R_3$$

дает

$$A_j^{N_1+N_2+N_3}$$
$$R_1+R_2+R_3.$$

Совмещение двух или более однотипных алгоритмов, выполненных в виде подпрограмм, дает новый совмещенный, рангу которого присваивается единица, а степень цикличности определяется суммой степеней составляющих алгоритмов.

Например, совмещение алгоритмов

$$A_{jp}^{N_1}, \quad A_{jp}^{N_2} \text{ и } A_{jp}^{N_3}$$
$$1 \qquad 1$$

дает

$$\underset{1}{A_{jp}}^{N_1+N_2+N_3}.$$

Записанные правила, естественно, следуют из физического смысла ранга и степени цикличности алгоритма. Ранг алгоритма, выполненного в виде подпрограммы, равен единице, так как такой алгоритм заносится в память машины один раз.

Использовав приведенные правила, совместим однотипные алгоритмы в формуле (1.13). При этом алгоритмы, выполненные в виде подпрограмм, запишем в квадратные скобки, а индекс p для упрощения опустим:

$$\underset{4}{\overset{13}{A_1}}\underset{2}{\overset{7}{A_2}}\underset{1}{\overset{1}{A_3}}\underset{2}{\overset{9}{A_4}}\underset{2}{\overset{12}{A_5}}[\underset{1}{\overset{4}{A_1}}\underset{1}{\overset{4}{A_2}}\underset{1}{\overset{11}{A_3}}]. \tag{1.14}$$

Для окончательного решения данной задачи совместим алгоритмы вне скобок и в скобках:

$$\underset{5}{\overset{17}{A_1}}\underset{3}{\overset{11}{A_2}}\underset{2}{\overset{12}{A_3}}\underset{2}{\overset{9}{A_4}}\underset{2}{\overset{12}{A_5}}. \tag{1.15}$$

Из выражения (1.15) находим ранги и степени цикличности всех составляющих алгоритмов. Эти данные используются в системе (1.9) для определения компонент кортежей совмещенных алгоритмов (1.15). Рассматривая полученные кортежи как простые, оцениваем эффективность сложного алгоритма B.

Рассмотрим описание сложного алгоритма B, у которого каждый элементарный участок (далее этот участок будем называть оператором) может быть реализован множеством простых алгоритмов (набором алгоритмов по каждому оператору).

Пусть для некоторого оператора составлен набор простых алгоритмов A_i, где $i = 1, 2, \ldots, n$. Кортежи этих алгоритмов представим следующей системой:

$$\begin{aligned} X(A_1) &= <x_{11}, x_{12}, \ldots, x_{1\eta}>; \\ X(A_2) &= <x_{21}, x_{22}, \ldots, x_{2\eta}>; \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ X(A_n) &= <x_{n1}, x_{n2}, \ldots, x_{n\eta}>, \end{aligned} \tag{1.16}$$

или

$$X(A_i) = <x_{i1}, x_{i2}, \ldots, x_{i\eta}> \qquad (i = 1, 2, \ldots, n).$$

Пусть алгоритм B составлен из m разнотипных операторов. По каждому оператору составлен набор простых алгоритмов. Общее число алгоритмов в наборе равно n_j ($j = 1, 2, \ldots, m$), общее число алгоритмов по всем наборам равно $\sum_{j=1}^{m} n_j$.

Кортежи таких алгоритмов представим следующей системой:

$$\left.\begin{array}{l}
j=1\left\{\begin{array}{l} X(A_{11}) = <x_{111},\ x_{112},\ \ldots,\ x_{11\eta}> \\ X(A_{12}) = <x_{121},\ x_{122},\ \ldots,\ x_{12\eta}> \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ X(A_{1n_1}) = <x_{1n_11},\ x_{1n_12},\ \ldots,\ x_{1n_1\eta}> \end{array}\right. \\
j=2\left\{\begin{array}{l} X(A_{21}) = <x_{211},\ x_{212},\ \ldots,\ x_{21\eta}> \\ X(A_{22}) = <x_{221},\ x_{222},\ \ldots,\ x_{22\eta}> \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ X(A_{2n_2}) = <x_{2n_21},\ x_{2n_22},\ \ldots,\ x_{2n_2\eta}> \end{array}\right. \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
j=m\left\{\begin{array}{l} X(A_{m1}) = <x_{m11},\ x_{m12},\ \ldots,\ x_{m1\eta}> \\ X(A_{m2}) = <x_{m21},\ x_{m22},\ \ldots,\ x_{m2\eta}> \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ X(A_{mn_m}) = <x_{mn_m1},\ x_{mn_m2},\ \ldots,\ x_{mn_m\eta}>, \end{array}\right.
\end{array}\right\} \quad (1.17)$$

или

$$X(A_{ji}) = <x_{ji\rho}>$$

($\rho = 1, 2, \ldots, \eta$; $i = 1, 2, \ldots, n_j$; $j = 1, 2, \ldots, m$).

Из множества представленных простых алгоритмов можно составить множество $\Gamma = \{\gamma_\xi\}$ решений некоторого сложного алгоритма B ($\xi = 1, 2, \ldots, S$; S — число решений). Любое γ_ξ решение характеризуется тем, что из каждого набора выбирается по одному алгоритму, который используется при синтезе алгоритма B. Число всевозможных решений

$$S = \prod_{j=1}^{m} n_j. \quad (1.18)$$

Для более удобного сравнения однозначных параметров системы (1.17) представим их в виде матриц. В матрицу

вносятся численные значения или множество однозначных параметров. Каждая матрица имеет наименование, определяемое параметрами, заключаемых в ней, например матрица констант, матрица времени реализации, матрица систем команд и т. д.

Матрица составляется следующим образом. В одной колонке размещаются однозначные компоненты только из конкретного набора системы (1.17). Количество колонок определяется величиной m. В каждой строке содержится по одной однозначной компоненте из различного набора. В общем случае для $j = 1, 2, \ldots, m$ и $i = 1, 2, \ldots, n_j$ однозначные параметры простых алгоритмов могут быть представлены в виде системы порядка η многокомпонентных матриц:

$$x_\rho = \begin{vmatrix} x_{11\rho} & x_{21\rho} & \ldots & x_{m1\rho} \\ x_{12\rho} & x_{22\rho} & \ldots & x_{m2\rho} \\ \cdot & \cdot & \cdot & \cdot \\ x_{1n_1\rho} & x_{2n_2\rho} & \ldots & x_{mn_m\rho} \end{vmatrix} \qquad (1.19)$$

$$(\rho = 1, 2, \ldots, \eta).$$

В большинстве случаев матрицы x_ρ не являются прямоугольными, так как число алгоритмов в наборе может быть различно, т. е. $n_1 \neq n_2 \neq \ldots \neq n_m$.

Если матрицы составлены из компонент кортежей (1.17), то такие матрицы называют простыми и записывают как x_ρ. В матрицах записывают количественные или качественные значения параметров, в которых отсутствует индексация кортежа, к которому они относятся. Поэтому надпишем над каждой колонкой алгоритм, к которому она относится. Номера алгоритмов в наборе не указывают, так как однозначные параметры кортежей одного набора в колонках каждой матрицы записываются в строго установленном порядке. Тогда система (1.19) примет вид:

$$x_\rho = \begin{vmatrix} A_1 & A_2 & & A_m \\ x_{11\rho} & x_{21\rho} & \ldots & x_{m1\rho} \\ x_{12\rho} & x_{22\rho} & \ldots & x_{m2\rho} \\ \cdot & \cdot & \cdot & \cdot \\ x_{1n_1\rho} & x_{2n_2\rho} & \ldots & x_{mn_m\rho} \end{vmatrix} \qquad (1.20)$$

$$(\rho = 1, 2, \ldots, \eta).$$

Приведенная система (1.20) описывает эффективность простых алгоритмов и не может быть использована в таком виде для определения эффективности сложного алгоритма B, так как не учитывает многократной повторяемости алгоритмов A_j в B. Для учета динамики простых алгоритмов в сложном необходимо преобразовать систему (1.20) в систему матриц совмещенных алгоритмов, компоненты которой определятся преобразованиями (1.9).

Проведем операцию совмещения простых однотипных алгоритмов, а значения их рангов R_j и степеней цикличности N_j занесем в матрицу-строку L, которая имеет следующий вид:

$$L = |\overset{A_1}{\{R_1, N_1\}}\ \overset{A_2}{\{R_2 N_2\}}\ \ldots\ \overset{A_m}{\{R_m, N_m\}}|. \qquad (1.21)$$

Система матриц совмещенных алгоритмов получается в результате умножения матрицы-строки L на каждую матрицу системы (1.20). При этом каждая компонента матрицы-строки L, принадлежащей вполне определенному алгоритму A_j, умножается на каждую компоненту колонки соответствующего алгоритма. Например, $\{R_1, N_1\}$, принадлежащая алгоритму A_1, умножается на каждую компоненту колонки A_1 в системе (1.20); $\{R_2, N_2\}$ — на компоненты колонки A_2 и т. д. Компоненты $\{R_j, N_j\}$ умножаются в соответствии с преобразованиями, установленными для каждого параметра. Например, для вышеописанных параметров справедливы преобразования (1.9), где ранг R_j умножается на компоненты матрицы d, N_j — на компоненты матрицы t, матрицы K и G остаются без изменений.

Таким образом, после умножения выражения (1.21) на (1.20) получим систему матриц совмещенных алгоритмов:

$$Lx_\rho = \begin{vmatrix} \overset{A_1}{x^0_{11\rho}} & \overset{A_2}{x^0_{21\rho}} & \ldots & \overset{A_m}{x^0_{m1\rho}} \\ x^0_{12\rho} & x^0_{22\rho} & \ldots & x^0_{m2\rho} \\ \ldots & \ldots & \ldots & \ldots \\ x^0_{1n_1\rho} & x^0_{2n_2\rho} & \ldots & x^0_{mn_m\rho} \end{vmatrix} \qquad (1.22)$$

$$(\rho = 1, 2, \ldots, \eta).$$

В дальнейшем индекс 0 возле компонент для упрощения записи будем опускать. Преобразованная система (1.22)

позволяет определить компоненты кортежа сложного алгоритма B. При этом, выбирая из каждой колонки по одной компоненте, что соответствует выбору по одному алгоритму из каждого набора, получаем множество $\Gamma = \{\gamma_\xi\}$ решений алгоритма B.

Выбор наилучших алгоритмов A_j можно ускорить, если перед оптимизацией (см. гл. II) провести некоторые преобразования над матрицами (1.22) для их упрощения.

Обозначим через Θ некоторую операцию над компонентами матрицы Lx_ρ. Тогда при определении $x_{\rho B}$ для некоторого γ_ξ решения над выбранными из каждой колонки по одной компоненте $x_{ji\rho}$ матрицы Lx_ρ проводится операция Θ:

$$x_{\rho B} = \mathop{\Theta}_{j=1}^{m} (x_{ji\rho}). \tag{1.23}$$

Для ранее рассмотренных показателей операция Θ определяется выражениями (1.2) — (1.5). Например, для $x_\rho = d$ и $m = 4$ из формулы (1.2) имеем, что Θ представляет собой операцию сложения четырех компонент

$$x_{\rho B} = \sum_{j=1}^{m} x_{ji\rho}.$$

Приведем некоторые способы упрощения матриц параметров алгоритмов.

1. Если компоненты некоторой колонки A_j равны между собой $x_{j1\rho} = x_{j2\rho} = \ldots = x_{jn_j\rho} = x_{j\rho}$, то $x_{j\rho}$ можно вынести за матрицу, так как в любом варианте γ_ξ решения из j-й колонки выбирается одно и то же значение $x_{j\rho}$. Например,

$$Lx_\rho = \begin{vmatrix} A_1 & A_2 & A_3 & A_4 \\ x_{11\rho} & x_{21\rho} & x_{31\rho} & x_{41\rho} \\ x_{12\rho} & x_{22\rho} & x_{32\rho} & x_{42\rho} \\ x_{13\rho} & x_{23\rho} & & x_{43\rho} \\ & x_{24\rho} & & \end{vmatrix}, \tag{1.24}$$

где число γ_ξ решений определяется из выражения (1.18)

$$S = \prod_{j=1}^{4} n_j = 3 \cdot 4 \cdot 2 \cdot 3 = 72.$$

Пусть компоненты колонок A_1 и A_3 равны между собой:

$$x_{11\rho} = x_{12\rho} = x_{13\rho} = x_{1\rho};$$

$$x_{31\rho} = x_{32\rho} = x_{3\rho}.$$

Вынесем за матрицу (1.24) коэффициенты $x_{1\rho}$ и $x_{3\rho}$

$$Lx_\rho = \overset{A_1}{x_{1\rho}}\Theta\overset{A_3}{x_{3\rho}}\Theta \begin{vmatrix} \overset{A_2}{x_{21\rho}} & \overset{A_4}{x_{41\rho}} \\ x_{22\rho} & x_{42\rho} \\ x_{23\rho} & x_{43\rho} \\ x_{24\rho} & \end{vmatrix}. \qquad (1.25)$$

Выражение (1.25) содержит меньшее число вариантов, так как $n_1 = 1$, $n_3 = 1$ и $S = 1 \cdot 1 \cdot 4 \cdot 3 = 12$.

2. Если колонка A_j имеет только одну компоненту $x_{j1\rho}$, то ее можно вынести за матрицу.

3. Если компоненты $x_{jl\rho}$ матрицы Lx_ρ есть множества и имеются элементы, которые принадлежат каждому множеству $x_{jl\rho}$, то такие элементы можно вынести за матрицу без обозначения принадлежности его к алгоритмам.

Например, в выражении

$$Lx_\rho = K = \begin{vmatrix} \overset{A_1}{\{k_1k_2k_3\}} & \overset{A_2}{\{k_2k_3\}} & \overset{A_3}{\{k_2k_3k_4\}} \\ \{k_2k_3\} & \{k_1k_2k_3\} & \{k_2k_3\} \\ \{k_2\} & \{k_2k_3\} & \{k_1k_2\} \\ \{k_2k_4\} & & \{k_2\} \end{vmatrix}$$

за матрицу можно вынести k_2, так как любой вариант, составленный из трех компонент (из каждой колонки берется одна компонента), содержит после операции соединения компоненту-множество с элементом k_2. Тогда

$$K = k_2 \cup \begin{vmatrix} \overset{A_1}{\{k_1k_3\}} & \overset{A_2}{\{k_3\}} & \overset{A_3}{\{k_3k_4\}} \\ \{k_3\} & \{k_1k_3\} & \{k_3\} \\ \varnothing & \{k_3\} & \{k_1\} \\ \{k_4\} & & \varnothing \end{vmatrix}, \qquad (1.26)$$

где $\varnothing$ — пустое множество.

Данный прием является частным случаем более общего: если компоненты множества некоторой j-й колонки содержат одинаковые элементы, то их можно вынести за матрицу без обозначения принадлежности ее к алгоритму. Например,

в колонке A_2 вынесем k_3, тогда выражение (1.26) имеет вид

$$K = \{k_2k_3\} \cup \begin{vmatrix} \overset{A_1}{\{k_1\}} & \overset{A_2}{\varnothing} & \overset{A_3}{\{k_4\}} \\ \varnothing & \{k_1\} & \varnothing \\ \varnothing & \varnothing & \{k_1\} \\ \{k_4\} & & \varnothing \end{vmatrix},$$

откуда видно, что количество операций при объединении множеств уменьшилось.

Рассмотрим случай, когда общий алгоритм C составляется из множества $M_C = \{B_p\}$ $(p = 1, 2, \ldots, P)$ сложных алгоритмов B_p. При этом каждый алгоритм B_p представляется множеством $M_{B_p} = \{A_j\}$ $(j = 1, 2, \ldots, m_{B_p})$ простых алгоритмов A_j, которые могут входить одновременно в различные B_p. Считаем, что алгоритмы B_p не связаны между собой циклами и разветвлениями.

Пусть алгоритм C состоит из множества

$$M_C = \bigcup_{p=1}^{P} B_p = \{A_j\} \qquad (j = 1, 2, \ldots, m_C). \quad (1.27)$$

Если алгоритмы B_p описаны системой простых матриц и матрицами L_{B_p}, то простые матрицы алгоритма C получаются после операции соединения (1.27)

$$x_{\rho C} = |x_{ji\rho}^{A_j}|$$

$(j = 1, 2, \ldots, m_C; \quad i = 1, 2, \ldots, n_j; \quad \rho = 1, 2, \ldots, \eta)$.

Для преобразования этой системы в матрицы совмещенных алгоритмов необходима матрица-строка L_C. Она получается в результате сложения рангов и степеней циклов одноименных алгоритмов A_j. Такое разбиение общего алгоритма C на ряд сложных B_p является удобным при проектировании алгоритмов, так как каждый алгоритм B_p может рассматриваться различными проектировщиками независимо.

2. ОБОБЩЕННЫЙ КРИТЕРИЙ

Представляя эффективность вычислительных алгоритмов перечислением их параметров в виде кортежей и матриц, приходим к частным показателям алгоритмов с одной стороны и структур УВС — с другой.

На основании частных показателей можно сформулировать критерий [24, 26, 41, 142], характеризующий степень согласования структур алгоритмов со структурами УВС. Прежде чем сформулировать критерий, рассмотрим более подробно частные критерии.

Критерии первого типа отличаются тем, что они характеризуют структуру алгоритма вне зависимости от конкретной логической структуры УВС, на которой он может быть реализован — это общее количество операций различных типов, время реализации алгоритма, количество исходных данных, промежуточных и окончательных результатов и др. Обозначим совокупность параметров алгоритма A в виде множеств

$$X = \{\vec{X}_1, \ldots, \vec{X}_l\}, \qquad (i = 1, 2, \ldots, l).$$

В общем случае каждый параметр может быть многокомпонентным и обозначается в виде

$$\vec{X}_i = (x_{i1}, \ldots, x_{iv_i}),$$

где v_i — число компонент параметра $\vec{X}_i$; l — число параметров алгоритма A.

Если $\vec{X}_i$ характеризует состав операций в алгоритме, то $x_{ii'}$ $(i' = 1, 2, \ldots, v_i)$ определяет количество операций i'-го типа. В зависимости от конкретных требований одни параметры играют первостепенную роль, а другие оказываются несущественными. В таком случае можно выбрать такое подмножество X', которое в достаточной мере характеризует структуру алгоритма A $(X' \subset X)$. Тогда критерий первого типа можно определить в виде функции, зависящей от параметров, входящих в множество X'

$$F(A) = F[A(X')].$$

Обычно выбирают критерий $F(A)$ так, чтобы качество алгоритма, представляемое множеством X' его параметров, оценивалось по экстремальному значению критерия при известных ограничениях на некоторые параметры $\vec{X}_i \in X'$.

Критерии второго типа предназначены для оценки вычислительных устройств вне зависимости от вида обрабатываемой информации. Такие критерии могут быть полезными при сравнении различных вариантов структур вычислительных устройств, близких по назначению. По аналогии

с предыдущим обозначением представим совокупность параметров УВС в виде множества $Y = \{\vec{Y}_1, \vec{Y}_2, \ldots, \vec{Y}_k\}$. Такими параметрами могут быть быстродействие, разрядность, объем памятей различных типов, система команд и др. Некоторые параметры могут оказаться многокомпонентными, т. е. в общем случае можно записать

$$\vec{Y}_j = y_{j1}, \; y_{j2}, \; \ldots, \; y_{j\mu_j} \qquad (j = 1, 2, \ldots, k),$$

где μ_j — число компонент вектора Y_j; k — число параметров структуры машины M.

Если $\vec{Y}$ характеризует память машины, то $y_{jj'}$ ($j' = 1, 2, \ldots, \mu_j$) определяет объем памяти j'-го типа в общей структуре машины. В зависимости от конкретных условий можно выбрать подмножество Y' ($Y' \subset Y$) существенных параметров, так что оно является достаточным для выделения данной структуры среди всех возможных структур, реализующих некоторый алгоритм. Тогда критерий второго типа может быть представлен в виде функции

$$F(M) = F[M(Y')],$$

оценивающей качество структуры по ее экстремальному значению при известных ограничениях на некоторые параметры из множества Y'.

В общем случае обобщенный критерий соответствия структур алгоритмов и УВС может быть представлен в виде $F(A, M)$ некоторой зависимости между параметрами алгоритма X' и структуры Y'. Если A_0 — множество возможных модификаций алгоритма решаемой задачи a, M_0 — множество возможных вариантов структур, способных реализовать множество алгоритмов A_0, то, как и в предыдущих случаях, выбор оптимального соответствия алгоритма и машины определяется экстремальным значением критерия $F(A, M)$. В конкретных условиях множества A_0 и M_0 ограничиваются определенными требованиями по некоторым параметрам из X' и Y'. Поэтому оптимальное соответствие разыскивается в подмножествах A_0' ($A_0' \subset A_0$) и M_0' ($M_0' \subset M_0$). Если эти подмножества каким-либо образом найдены, то решение задачи оптимального соответствия алгоритма и структуры сводится к нахождению такой пары A_{opt} и M_{opt}, которые своими параметрами обращают критерий $F(A, M)$ в максимум или минимум. В простейшем случае

задача решается с использованием простого перебора, что возможно лишь при малом количестве пар вида (A'_{0i}, M'_{0j}), где $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$; n и m — соответственно мощности множеств A'_0 и M'_0.

Использование указанного критерия при исследовании эффективной реализации алгоритмов связано в первую очередь с нахождением соответствующих функциональных связей между параметрами алгоритмов и структур УВС при соответствующих ограничениях, а затем с нахождением алгоритма оптимизации, т. е. алгоритма поиска экстремального значения критерия эффективности $F(A, M)$.

Рассмотрим возможности такого критерия при исследовании основного цикла в адресных структурах УВМ. Такое изучение удобно проводить сравнением качественного состава операций, составляющих данный алгоритм, и вычислительных возможностей машины. Анализируя вычислительный процесс в УВМ различных структур, обладающих вполне определенными основными циклами при выполнении операции, приходим к выводу, что при реализации одного и того же алгоритма такие машины затрачивают различную информационную работу. Это вызвано особенностями логической структуры и видом алгоритмов, подлежащих реализации в машине. Логическая структура характеризуется типом основного цикла, на который оказывают влияние следующие факторы:

число адресов в командах машины;

совмещение выбора команды из памяти и выполнение арифметической или какой-либо другой операции;

наличие раздельных памятей для команд и чисел и возможность одновременного к ним обращения;

наличие в арифметическом устройстве буферных регистров для записи результатов в памяти;

особенность системы команд.

Указанные факторы проявляются в реальных структурах в различных сочетаниях, что приводит к большому многообразию логических структур УВС. Основной цикл характеризуется числом обращений к памяти и количеством микротактов, необходимых для преобразования информации. Каждое обращение, в свою очередь, представляется также в виде некоторого числа требуемых микротактов. По виду основного цикла различают две группы операций: передачи (сортировки) и преобразования (обработки)

информации, что записывается в виде

$$\begin{gathered}\alpha \bigcirc \beta \to \gamma; \\ \alpha \to \gamma,\end{gathered} \tag{1.28}$$

где α, β, γ — содержимое ячеек памяти или регистров арифметического устройства.

Если в операции $\alpha \to \gamma$ символы α и γ обозначают содержимое ячеек памяти, то такая операция выполняется в машине посредством передачи информации через регистр арифметического устройства и имеет вид $\alpha \to \Sigma\gamma$.

Пусть алгоритм, подлежащий реализации УВМ, содержит m типов операций, от соотношения которых зависит количество обращений к памяти и, следовательно, скорость работы машины. Тогда число обращений к памяти в адресной структуре выражается в виде линейной функции

$$F = y_1 x_1' + y_2 x_2' + \cdots + y_m x_m', \tag{1.29}$$

где y_i — число обращений к памяти в конкретной структуре, приходящееся на одну i-ю операцию вида (1.28); x_i' — число операций i-го типа ($i = 1, 2, \ldots, m$).

Нетрудно заметить, что записанная функция связывает показатели алгоритма и структуры машины, причем

$$F(A, M) = \vec{X}\vec{Y},$$

где $\vec{X}\vec{Y}$ — скалярное произведение двух m-мерных векторов.

При этом $l = k = 1$, $\nu = \mu = m$. Оптимальное соответствие алгоритма и структуры будет при $F = F_{\min}$.

Найдем предельное значение критерия $F(A, M)$ при фиксированной структуре УВМ и произвольном алгоритме в смысле качественного и количественного состава операций в нем. Запишем следующие ограничения:

$$\begin{gathered}x_1' + x_2' + \cdots + x_m' = N; \\ 0 \leqslant x_i' \leqslant N; \quad y_i \geqslant 0; \\ N = \text{const}; \quad y_i = \text{const},\end{gathered} \tag{1.30}$$

где y_i — целые положительные числа.

Заметим, что $y_i = 0$, если некоторая операция не требует обращений к обычной памяти; в этой операции используются «быстрые ячейки», т. е. буферные регистры арифмети-

ческого устройства. Таким образом, задача нахождения предельных значений потерь времени на обращение к памяти сводится к задаче линейного программирования, в которой линейной формой является критерий (1.29), а систему ограничений представляют выражения (1.30). Для решения этой задачи можно применить, например, симплекс-метод и найти предельные значения критерия $F(A, M)$. После соответствующих выкладок получим

$$(\min g_i) N \leqslant F(A, M) \leqslant (\max y_i) N. \qquad (1.31)$$

Если задан некоторый класс адресных структур в виде множества $\{M\}$ с мощностью H, то можно определить предельные значения критерия $F(A, M)$. Пусть y_{ih} значение i-й компоненты вектора Y в h-й структуре ($h = 1, 2, \ldots, H$). При этом имеем следующее соотношение $\min y_{ih} \leqslant \leqslant y_{ih} \leqslant \max y_{ih}$, где максимум (минимум) берется по всем i и h. Выражение (1.31) перепишем в виде

$$(\min y_{ih}) N \leqslant F(A, M) \leqslant (\max y_{ih}) N. \qquad (1.32)$$

Для более удобного использования критерия $F(A, M)$ применим нормированные значения компонент вектора $\vec{X}$, для чего в выражении $x_1' + x_2' + \ldots + x_m' = N$ разделим обе части равенства на N и получим

$$\sum_{i=1}^{m} x_i = 1, \quad 0 \leqslant x_i \leqslant 1,$$

где x_i — относительное содержание операций i-го типа в заданном алгоритме.

Величину x_i можно представить как частоту появления операции i-го типа или в предельном случае как вероятность появления i-й операции.

Для УВМ целесообразно использование критерия $F(A, M)$ для целого класса алгоритмов A ($a_j \in A$, $j = 1, 2, \ldots, n$), реализуемых на машине. Обозначим вероятность появления j-й задачи через q_j. При этом очевидно, что

$$\sum_{j=1}^{n} q_j = 1, \quad 0 \leqslant q_j \leqslant 1.$$

Пусть F_j — число обращений к памяти при решении j-й задачи. Тогда при заданном составе операций получим

$$F_j = \sum_{i=1}^{m} y_i x_{ij}, \qquad (1.33)$$

где

$$\sum_{i=1}^{m} x_{ij} = N_j$$

суммарное количество операций в j-й задаче.

Зная q_j, находим среднее значение критерия F для класса задач в виде математического ожидания

$$M[F] = \frac{F_1q_1 + F_2q_2 + \cdots + F_nq_n}{\sum_{j=1}^{n} q_j} = \sum_{j=1}^{n} F_jq_j,$$

а с учетом формулы (1.33) получаем

$$M[F] = \sum_{j=1}^{n} q_j \sum_{i=1}^{m} y_i x_{ij}.$$

Найдем величину $M'[F]$, характеризующую среднее число обращений к памяти при вероятностном задании операций в задачах a_j в виде значений x_{ij}. Используя теорему о произведении вероятностей независимых событий, можно утверждать, что вероятность появления операции i-го типа при решении j-й задачи подчиняется выражению

$$P_i^j = x_{ij}q_j.$$

Действительно, если просуммировать величину P_i^j по i и j, то получим

$$\sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij}q_j = 1.$$

Тогда можно записать, что

$$M'[F] = \sum_{i=1}^{m} y_i \sum_{j=1}^{n} x_{ij}q_j. \qquad (1.34)$$

Однако выражение (1.34) справедливо, если выполняется условие $N_1 = N_2 = \ldots = N_m$, т. е. при одинаковых сложностях решаемых задач. При условии $N_1 \neq N_2 \neq \ldots \neq N_m$ вероятность P_i появления операции i-го типа для класса задач зависит от соотношения их сложностей

$$P_i = \sum_{j=1}^{m} x_{ij} \frac{q_jr_j}{\sum_{j=1}^{m} q_jr_j},$$

где

$$r_j = \frac{N_j}{\sum_{j=1}^{m} N_j}.$$

При этом

$$M'[F] = \sum_{i=1}^{n} y_i P_i.$$

Таким образом, обобщенный критерий позволяет оценить степень согласования характеристик алгоритмов со структурами УВС.

Глава II

МЕТОДЫ ОПТИМИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ ПО ИХ ПАРАМЕТРАМ

1. СОДЕРЖАТЕЛЬНОЕ ОПИСАНИЕ ПРОЦЕССА ОПТИМИЗАЦИИ

Исходным для решения задач оптимизации является набор простых алгоритмов по каждой функции или операции. При составлении такого набора алгоримов проектировщику предоставляется возможность свободно выбирать систему команд, вводить специальные команды, которые позволяют изменять длину программы вычислений и время реализации алгоритма в ту или другую сторону, изменять конструкцию УВМ. По окончании составления каждой программы вычисления простой операции численно оцениваются параметры алгоритма. Эти оценки записываются в виде кортежей. Каждый алгоритм одного и того же набора отличается от другого значением параметров. Алгоритм управления, составленный из простых подалгоритмов, определяется параметрами, которые являются функциями параметров простых подалгоритмов.

Обеспечение необходимой эффективности алгоритма управления связано с выполнением противоречивых требований [111].

Из множества простых алгоритмов в каждом наборе необходимо выбрать такие, которые позволяют получить один из параметров сложного алгоритма B. Например,

для того чтобы получить минимальное время реализации алгоритма, необходимо выбрать из каждого набора по одному алгоритму с минимальным временем реализации. Однако по другим парамерам (например, длина программы, объем системы команд) этот алгоритм может не удовлетворять поставленным требованиям.

Таким образом, целью оптимизации является минимизация одного из параметров сложного алгоритма при условии, что все остальные параметры удовлетворяют поставленным ограничениям.

Задачи подобного типа в настоящее время решаются различными методами математического программирования. Однако свести рассматриваемую задачу к уже известным затруднительно, так как это связано с наличием неарифметических операций над простыми переменными (параметрами x_{jlp}). Эти операции определяются из внутри- и межкортежных зависимостей. В частности, для ранее рассмотренных параметров неарифметическими операциями являются соединения множеств и определение числа элементов в множестве. Эти операции могут входить совместно с арифметическими в целевую функцию и ограничения.

Рассматриваемую задачу сформулируем следующим образом. Пусть в УВМ требуется реализовать множество операторов

$$F \equiv f\{f_j\} \qquad (j = 1, 2, \ldots, m),\ F \neq \varnothing, \tag{2.1}$$

где m — общее число операторов; j — номер оператора.

Каждый оператор f_j может быть реализован несколькими алгоритмами (набором) с взаимно противоречивыми показателями. Такой набор представим в виде множества

$$f_j \equiv \{A_{ji}\} \qquad (i = 1, 2, \ldots, n_j),\ f_j \neq \varnothing, \tag{2.2}$$

где n_j — число алгоритмов в наборе f_j; i — номер алгоритма в j-м наборе.

Из тождеств (2.1) и (2.2) следует

$$f_j \subseteq F \text{ и } A_{ji} \in F. \tag{2.3}$$

Для $f_j = F$ согласно выражению (2.3) имеем

$$F = \{A_{11}, A_{12}, \ldots, A_{1n_1}\}. \tag{2.4}$$

Выражение (2.4) соответствует случаю, когда множество состоит из простых алгоритмов одного и того же оператора. Очевидно, что решить эту задачу не представляет труда,

так как процесс оптимизации сводится к выбору одного алгоритма простым перебором.

Рассмотрим случай $f_j \subset F$ при $m \geqslant 2$. Пусть операторы f_j используются для составления некоторого сложного алгоритма B. Если каждый j-й оператор характеризуется набором простых A_{ji} ($i = 1, 2, \ldots, n_j$) алгоритмов, представленных простыми кортежами, то для алгоритма B составляется система простых матриц x_ρ ($\rho = 1, 2, \ldots, \eta$), которая затем преобразуется в систему матриц совмещенных алгоритмов:

$$Lx_\rho = \overset{A_j}{|x_{ji\rho}|} \tag{2.5}$$

($j = 1, 2, \ldots, m$; $i = 1, 2, \ldots, n_j$; $\rho = 1, 2, \ldots, \eta$).

Для получения некоторого решения γ_ξ необходимо из каждой колонки (набора) A_j матриц (2.5) выбрать по одной компоненте. Над выбранными m компонентами каждой матрицы проводится межкортежная операция Θ_ρ, определяемая характером параметров. Тогда

$$x_{\rho B} = \overset{m}{\underset{j=1}{\Theta_\rho}}(x_{ji\rho}) \qquad (\rho = 1, 2, \ldots, \eta). \tag{2.6}$$

Однако другие параметры кортежа алгоритма B (например, λ, ϑ, D) определяются через внутрикортежную операцию Θ_l. Тогда для параметра x_{lB} ($l \neq \rho$)

$$x_{lB} = \Theta_l(x_{\rho B}), \tag{2.7}$$

или

$$x_{lB} = \overset{\Phi}{\underset{\varphi=1}{\Theta_l}}(x_{\varphi B}), \tag{2.8}$$

где φ принимает ряд значений ρ и $\varphi \neq l$.

В выражениях (2.6) — (2.8) операции Θ_ρ и Θ_l могут быть как арифметическими, так и неарифметическими. Например, для определения показателя t_B используем операцию сложения t_{ji}. Однако объем долговременной памяти D_B представляет собой сложную зависимость

$$D_B = d_B + \lambda_B,$$

где

$$d_B = \sum_{j=1}^{m} d_{ji},$$

а λ_B определяется после операции соединения множеств

$$K_B = \bigcup_{j=1}^{m} K_{ji} = \{k_{ji\mu}\}$$

и операции подсчета числа элементов в множестве $\{k_{ij\mu}\}$, где $\mu = 1, 2, \ldots, \lambda_B$.

Таким образом, над компонентами матриц совмещенных алгоритмов (2.5) проводятся операции (2.6) — (2.8), которые позволяют получить компоненты кортежа сложного алгоритма

$$X(B) = \langle x_{1B}, x_{2B}, \ldots, x_{\eta B} \rangle.$$

При этом в процессе оптимизации один из параметров ($x_{\rho^\circ B}$) минимизируется, а остальные должны удовлетворять условиям

$$x_{\rho B} \leqslant x_\rho^0$$

($\rho = 1, 2, \ldots, \eta$; $\rho = \rho^0$),

где x_ρ^0 — ограничения на параметр $x_{\rho B}$.

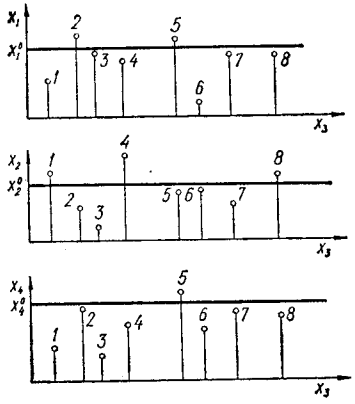


Рис. 1.

Предположим, что удалось получить кортежи всех S вариантов (1.18). Каждое решение γ_ξ представляется вектором эффективности в η-мерном пространстве.

На рис. 1 изображено множество решений в виде дискретных точек, которые являются концами вектора эффективности некоторого алгоритма B, имеющего кортеж с $\eta = 4$ и числом решений $S = 8$. Для удобства выбора оптимального варианта по оси абсцисс отложены полученные значения минимизирующего параметра в каждом варианте, а по оси ординат — значения параметров, на которые наложены ограничения, представленные прямой, параллельной оси абсцисс.

Среди компонент $x_1 — x_4$ кортежа $x(B)$ минимизируется параметр x_3. На остальные наложены ограничения:

$$x_1 \leqslant x_1^0; \quad x_2 \leqslant x_2^0; \quad x_4 \leqslant x_4^0. \tag{2.9}$$

Ограничения (2.9) позволяют сразу исключить из рассмотрения варианты *1*, *2*, *4*, *5*, *8*, как не удовлетворяющие

поставленным требованиям (ограничениям). Из оставшихся вариантов выбирается 3-й, так как он содержит $x_3 = x_{3\min}$, и оптимальными A_{jl} будут те, которые составляют алгоритм B в 3-м варианте.

В процессе оптимизации могут получиться несколько оптимальных вариантов, у которых минимизирующий параметр принимает одинаковое значение, а все остальные компоненты находятся в пределах ограничений. В этом случае необходимо провести минимизацию по другому параметру, а первый — исключить из рассмотрения. При этом минимизация проводится среди оптимальных по первому параметру вариантов. При подобной ситуации предварительно устанавливается система приоритетов на параметры, среди которых минимизируется компонента с высшим приоритетом.

Таким образом, оптимизация некоторого алгоритма по заданным критериям является достаточно сложной задачей. Кроме того, с увеличением числа алгоритмов m и числа n_j в каждом наборе объем вычислительных работ растет весьма быстро. Поэтому для решения таких задач следует применять специальные методы, позволяющие с меньшей трудоемкостью находить оптимальные решения.

2. МЕТОДЫ ПЕРЕБОРА И ПОСЛЕДОВАТЕЛЬНЫХ ПРИБЛИЖЕНИЙ

Метод перебора заключается в последовательном переборе всех S вариантов. Для каждого решения γ_ξ составляется кортеж $x(B)$ из полученных значений $x_{\rho B}$. Отбрасывая те варианты, которые содержат хотя бы одну компоненту, не удовлетворяющую поставленному требованию, из оставшихся выбираем тот, у которого минимизирующая компонента имеет минимальное значение. Очевидно, метод перебора можно использовать при отыскании оптимальных алгоритмов A_{jl} для небольшого числа решений γ_ξ. При больших m и n_j этот метод становится неэффективным.

Метод последовательных приближений представляет собой направленный перебор вариантов. С каждым шагом приближения значение минимизирующей компоненты увеличивают и проверяют систему ограничений. Этот процесс продолжается до того варианта, при котором все остальные компоненты находятся в пределах ограничений.

Пусть система приоритетов по параметрам алгоритма B составлена так, что первым следует минимизировать

$x_{\rho B} = x_{\rho^{\circ}B}$. Направленный перебор начинается с варианта $\gamma_{\xi} = \gamma_1$, у которого параметр $x_{\rho^{\circ}B}$ имеет наименьшее значение. Если этот вариант не удовлетворяет требованиям по другим параметрам, то переходят к варианту $\gamma_{\xi} = \gamma_2$, имеющему значение $x_{\rho^{\circ}B}$ большее, чем в предыдущем варианте.

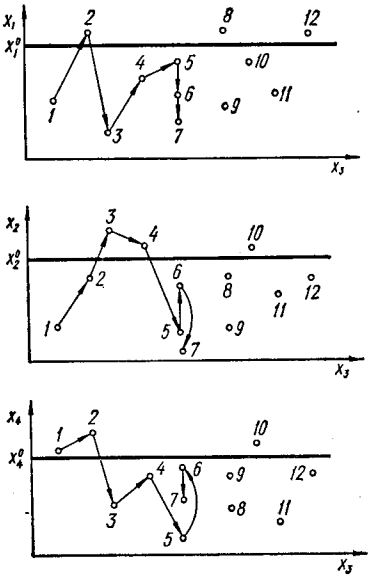


Рис. 2.

В процессе последовательных приближений всегда должно выполняться условие

$$x_{\rho^{\circ}B}(\gamma_{\xi}) < x_{\rho^{\circ}B}(\gamma_{\xi+1}),$$

где $x_{\rho^{\circ}B}(\gamma_{\xi})$, $x_{\rho^{\circ}B}(\gamma_{\xi+1})$ — значения параметров в γ_{ξ} и $\gamma_{\xi+1}$ вариантах.

В каждом решении γ_{ξ} определяют значения всех параметров алгоритма, и если в рассматриваемом варианте не выполняется ограничение хотя бы для одного из параметров, то это решение исключают и переходят к $\gamma_{\xi+1}$ варианту. Этот процесс с монотонным возрастанием минимизирующего параметра продолжается до тех пор, пока не будет найден оптимальный вариант γ_{ξ}^{opt}, содержащий все остальные параметры в пределах ограничений. Полученное значение минимизирующего параметра считается наименьшим ($x_{\rho^{\circ}B} = x_{\rho^{\circ}B\min}$) в рамках заданных наборов простых алгоритмов.

После нахождения первого оптимального варианта γ_{ξ}^{opt} процесс последовательных приближений продолжается для выявления вариантов с одинаковым значением минимизирующего параметра. При этом каждый раз проверяют условие

$$x_{\rho^{\circ}B}(\gamma_{\xi+\alpha}^{opt}) = x_{\rho^{\circ}B\min}, \qquad (2.10)$$

где $x_{\rho^{\circ}B}(\gamma_{\xi+\alpha}^{opt})$ — значение минимизирующего параметра в $\gamma_{\xi+\alpha}^{opt}$-варианте ($\alpha = 0, 1, 2, \ldots$).

Для $\alpha = a + 1$ получим

$$x_{\rho^0 B}(\gamma^{\text{opt}}_{\xi+a+1}) > x_{\rho^0 B\min},$$

тогда $a + 1$ — число вариантов с одинаковым значением минимизирующего параметра. Эти частные решения ($a + 1$ вариант) следует минимизировать теперь по другому параметру, исключив $x_{\rho^0 B}$ из дальнейшего рассмотрения.

Если α принимает единственное значение ($\alpha = 0$), то оптимальный вариант найден, и алгоритмы A_{jt}, составляющие данное решение, являются наилучшими.

На рис. 2 показаны последовательные приближения от γ_1 до оптимального варианта по параметру x_3. При этом оптимальными оказались варианты γ_5, γ_6, γ_7, так как для них выполняется условие (2.10)

$$x_3(\gamma_5) = x_3(\gamma_6) = x_3(\gamma_7).$$

Из трех вариантов γ_5, γ_6, γ_7 выбирают один в зависимости от того, какой из параметров (x_1, x_2, x_4) имеет наивысший приоритет. Если минимизацию проводить по параметру x_1, то выбирают вариант γ_7, если по x_2, то — γ_7, если по x_4, то — γ_5.

В отличие от метода перебора в методе последовательных приближений не рассматривают варианты с $x_{\rho^0 B} > x_{\rho^0 \min}$, которые заведомо являются худшими и число которых может быть значительным.

3. МЕТОДЫ ИСКЛЮЧЕНИЯ ВАРИАНТОВ И ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

При этом методе из рассмотрения исключаются те компоненты в матрицах Lx_ρ, которые заведомо не обеспечат выполнения условия ограничения по данному параметру $x_{\rho B}$, что значительно уменьшает объем вычислений при оптимизации. При этом анализируются только матрицы Lx_ρ, параметры которых входят в систему ограничений. Это дает возможность исключить из рассмотрения максимальное число компонент во всех матрицах, как не удовлетворяющих совместно с другими поставленным требованиям.

Из анализа исключаются кортежи тех простых алгоритмов, для которых выполняются следующие условия:

1. Если в выбранной матрице имеется компонента

$$x_{jl\rho} > x_{\rho}^{0},$$

где x_{ρ}^{0} — ограничение, то все параметры кортежа, содержащего данную компоненту, исключаются из всех матриц, так как эффективность этого алгоритма не удовлетворяет требованиям по одному из параметров.

2. Если максимальная в матрице компонента

$$x_{jl\rho\max} < x_{\rho}^{0},$$

однако совместно с другими наименьшими параметрами $x_{l/\rho\min}$ удовлетворяет условию

$$x_{\rho B} = x_{jl\rho\max} \overset{m}{\underset{j=1}{\Theta}} (x_{jl\rho\min}) > x_{\rho}^{0},$$

то параметры кортежа, содержащего $x_{jl\rho\max}$, исключаются из всех матриц, так как один из параметров не удовлетворяет совместно с любыми другими однозначными компонентами поставленным требованиям. При таком последовательном анализе матриц Lx_{ρ} значительно уменьшается число их компонент, а значит, и число вариантов.

Метод исключения вариантов является предварительным. Однако его можно эффективно использовать совместно с другими методами. Выбор оптимальных алгоритмов A_{jl} можно свести к задаче линейного программирования о назначениях (проблема выбора) [69, 78] только для арифметических операций Θ_{ρ}.

Для минимизирующего параметра x_{ρ^0} целевая функция $x_{\rho^0 B}$ имеет вид

$$x_{\rho^0 B} = \sum_{j=1}^{m} \sum_{l=1}^{n_j} x_{jl\rho} a_{jl},$$

где $a_{jl} = 1$, если алгоритм A_{jl} используется в алгоритме B, и $a_{jl} = 0$, если — не используется.

Для a_{jl} должны выполняться следующие условия:

$$\sum_{l=1}^{n_j} a_{jl} = 1 \qquad (j = 1, 2, \ldots, m);$$

$$a_{jl} \geqslant 0.$$

Для параметров, на которые наложены ограничения, записываем систему линейных неравенств:

$$\sum_{j=1}^{m}\sum_{i=1}^{n_j} x_{jl\rho}a_{jl} \leqslant x_{\rho}^{0} \qquad (\rho = 1, 2, \ldots, \eta).$$

4. НОРМИРОВАНИЕ МАТРИЦ ПАРАМЕТРОВ АЛГОРИТМОВ

Пусть некоторый сложный алгоритм B описан η-системой матриц совмещенных алгоритмов (2.5). Если требования к отдельным параметрам алгоритма B не заданы, однако его необходимо оптимизировать для приведения всех компонент матриц к единой норме, то эту задачу решают следующим образом.

Приведем компоненты всех η-систем матриц к единой норме, в качестве которой для каждой матрицы выбирается максимальное значение компоненты в этой матрице. Пронормировав все остальные компоненты матриц относительно ее максимальной $x_{jl\rho\max}$, получим систему матриц-норм:

$$HLx_{\rho} = \left| \frac{x_{jl\rho}}{x_{jl\rho\max}} \right|$$

$(j = 1, 2, \ldots, m; \quad i = 1, 2, \ldots, n_j; \quad \rho = 1, 2, \ldots, \eta).$

Для определения эффективности простого алгоритма A_{jl} составим сумму разнозначных пронормированных компонент, принадлежащих кортежу одного алгоритма. Тогда нормы-эффективности простых алгоритмов

$$Hx(A_{jl}) = \sum_{\rho=1}^{\eta} b_{jl\rho} \qquad (j = 1, 2, \ldots, m; \quad i = 1, 2, \ldots, n_j),$$

где

$$b_{jl\rho} = \frac{x_{jl\rho}}{x_{jl\rho\max}}.$$

Из каждого j-го набора выбирают по одному алгоритму, для которых

$$Hx(A_{jl}) = Hx(A_{jl})_{\min}.$$

Если на один из параметров сложного алгоритма наложено ограничение, то матрица, описывающая данный параметр, не нормируется. Выбор оптимального по норме варианта

в этом случае решается системой двух матриц:

$$Lx_{\bar{\rho}} = |x_{i\bar{\rho}}|$$

$$(j = 1, 2, \ldots, m; \quad i = 1, 2, \ldots, n_j);$$

$$HLx_{\rho} = |b_{ji\rho}|$$

$$(j = 1, 2, \ldots, m; \quad i = 1, 2, \ldots, n_j),$$

где $Lx_{\bar{\rho}}$ — матрица, на параметры которой наложено ограничение ($\rho = \bar{\rho}$).

Таким же образом находят оптимальные по норме алгоритмы, если наложены ограничения на несколько параметров. Однако это приводит к росту числа матриц и объему вычислений. Данный метод позволяет быстрее других находить оптимальный по норме вариант, который может быть не оптимальным при известных ограничениях. Кроме того, этот метод может быть использован для матриц, компонентами которых являются числа.

5. ВЫБОР УВМ ДЛЯ ЭФФЕКТИВНОЙ РЕАЛИЗАЦИИ АЛГОРИТМОВ УПРАВЛЕНИЯ

Описание эффективности алгоритма, рассмотренное в начале главы, применительно не только к оптимизации алгоритма управления, но и к целенаправленному выбору варианта эффективной реализации алгоритма управления на одной из имеющейся УВМ. Однако в данном случае описанию эффективности подлежит не только алгоритм, но и УВМ с точки зрения реализации заданного алгоритма на конкретной машине.

Задачу эффективной реализации исходного алгоритма на одной из множества управляющих машин сформулируем следующим образом.

Пусть имеется парк из m управляющих вычислительных машин. Для каждой i-й машины ($i = 1, 2, \ldots, m$) известны ее технические характеристики и набор реализуемых операций (система команд) $G_i = \{g_{ji}\}$, где j — номер операций ($j = 1, 2, \ldots, p_i$).

Заданный класс алгоритмов управления представлен в виде математических зависимостей. Задача состоит в том, чтобы из имеющегося парка УВМ выбрать ту, на которой заданный алгоритм управления реализовался бы наилуч-

шим образом, т. е. имел наименьшее время реализации, программа алгоритма разместилась бы в памяти машины и результаты вычислений имели бы погрешность меньше допустимой.

Одним из способов решения этой задачи является составление программы для каждой машины. Оценив эффективность каждой программы путем сравнения, выберем УВМ. Однако этот способ требует значительного времени и больших экономических затрат.

Рассмотрим более экономичный метод выбора варианта для эффективной реализации алгоритма. Программу алгоритма управления составим для гипотетической машины, которую затем можно формально транспонировать на любую из парка УВМ. Система команд (СК) гипотетической машины

$$G_0 = \bigcup_{i=1}^{m} G_i = \{g_{0j}\},$$

где $j = 1, 2, \ldots, p_0$.

Критериями сравнения транспонированных программ для конкретных УВМ могут быть следующие показатели:

погрешность вычисления Δ (абсолютная или относительная);

объем долговременной памяти D^0 (число ячеек);

время реализации T, выраженное в единицах τ_0,— времени реализации наиболее короткой операции.

Для определения этих показателей алгоритма опишем эффективность отдельных ее элементарных участков кортежами

$$A = <\Delta, \partial, K, \tau>,$$

где Δ — погрешность; ∂ — объем программы (число команд); $K = \{k_\xi\}$ — система констант ($\xi = 1, 2, \ldots, \psi$); τ — время реализации.

В качестве элементарных участков программы алгоритма для гипотетической машины примем ее отдельные операции (команды). Эффективность системы команд выразим через показатели СК G_i заданного парка машин и представим ее в виде следующей системы кортежей:

$$g_{01} = \left| \begin{array}{l} <\Delta_{11}\partial_{11}K_{11}\tau_{11}> \\ <\Delta_{12}\partial_{12}K_{12}\tau_{12}> \\ \cdots\cdots\cdots \\ <\Delta_{1m}\partial_{1m}K_{1m}\tau_{1m}> \end{array} \right|$$

$$g_{02} = \left\{ \begin{array}{l} <\Delta_{21}\partial_{21}K_{21}\tau_{21}> \\ <\Delta_{22}\partial_{22}K_{22}\tau_{22}> \\ \ldots\ldots\ldots \\ <\Delta_{2m}\partial_{2m}K_{2m}\tau_{2m}> \end{array} \right\} \qquad (2.11)$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$g_{0p_0} = \left\{ \begin{array}{l} <\Delta_{p_01}\partial_{p_01}K_{p_01}\tau_{p_01}> \\ <\Delta_{p_02}\partial_{p_02}K_{p_02}\tau_{p_02}> \\ \ldots\ldots\ldots \\ <\Delta_{p_0m}\partial_{p_0m}K_{p_0m}\tau_{p_0m}>, \end{array} \right\}$$

или

$$g_{0j} = \left\{ <\Delta_{ji}\partial_{ji}K_{ji}\tau_{ji}> \right.$$
$$\left. (i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, p_0). \right.$$

Следует заметить, что СК G_0 гипотетической машины должна быть совместимой с СК любой ЦВМ из имеющегося парка. Совместимость системы команд предполагает также совместимость операций.

Совместимыми операциями называются те, которые могут быть заменены в i-й машине по своему смыслу либо одной командой, либо программой (подпрограммой).

Например, операция сложения из СК G_0 является совместимой с операцией сложения практически любых УВМ. Совместимыми будут также операции деления и элементарные функции, так как они могут быть представлены в i-й машине одной командой или программой (подпрограммой).

Однако совместимость СК G_0 будем относить не только к СК парка рассматриваемых УВМ, но и к исходному алгоритму. Эффективность оценивается для тех операций СК G_0, которые входят в состав программы алгоритма. Операции СК G_0, не участвующие в вычислении, нет надобности оценивать по эффективности и они могут быть вычеркнуты из списка G_0. Если все оставшиеся операции, используемые в программе, реализуются на любой из m УВМ, то такая СК G_0 называется совместимой.

Одна и та же команда СК G_0 может находиться в различных соотношениях к соответствующей операции СК G_i. Различают совместимые, программносовместимые, подпрограммносовместимые и несовместимые операции.

Совместимые — это операции СК G_0, которые в i-й машине реализуются одной командой.

Программносовместимые — это операции СК G_0, которые в i-й машине, как правило, представляются небольшой программой. Эта программа прошивается в памяти столько раз, сколько она встречается в программе алгоритма управления.

Подпрограммносовместимые — это такие операции СК G_0, которые в i-й машине реализуются через подпрограммы, например операции вычисления элементарных функций.

Несовместимые — это операции СК G_0, которые не могут быть реализованы хотя бы на одной из m УВМ.

Таким образом, если в целом системы команд m УВМ несовместимы, но относительно исходного алгоритма они совмещаются, то такие системы команд называются совместимыми. Эффективность СК G_0 (2.11) гипотетической машины оценивается только для совмещенных операций. Все дальнейшие рассуждения проведем для совместимой СК G_0.

Для совместимых операций показатель $\partial = 1$, для программно- и подпрограммносовместимых операций этот показатель характеризует число команд программы, реализующих заданную операцию в СК G_i.

Показатели Δ, ∂, K, τ для совместимых операций легко вычислить из технических характеристик G_i каждой машины, а для программно- и подпрограммносовместимых операций — из программ. Для более удобного сравнения однозначных показателей систему кортежей (2.11) представляют в виде исходной системы матриц однозначных параметров. При этом в матрице, описывающей время реализации операций, каждый элемент приводится к норме

$$\tau_{ji}[\tau_0] = \frac{\tau_{ji}}{\tau_0},$$

где τ_0 — время реализации самой короткой операции среди всех команд машинного парка.

Система исходных матриц имеет вид:

$$I\Delta = \left\| \begin{vmatrix} \Delta_{11} & \Delta_{21} & \dots & \Delta_{p1} \\ \Delta_{12} & \Delta_{22} & \dots & \Delta_{p2} \\ \dots & \dots & \dots & \dots \\ \Delta_{1m} & \Delta_{2m} & \dots & \Delta_{pm} \end{vmatrix} \right\|;$$

$$
\left.
\begin{aligned}
I\partial &= \begin{vmatrix} \partial_{11} & \partial_{21} & \dots & \partial_{p1} \\ \partial_{12} & \partial_{22} & \dots & \partial_{p2} \\ \dots & \dots & \dots & \dots \\ \partial_{1m} & \partial_{2m} & \dots & \partial_{pm} \end{vmatrix}; \\
IK &= \begin{vmatrix} K_{11} & K_{21} & \dots & K_{p1} \\ K_{12} & K_{22} & \dots & K_{p2} \\ \dots & \dots & \dots & \dots \\ K_{1m} & K_{2m} & \dots & K_{pm} \end{vmatrix}; \\
I\tau &= \begin{vmatrix} \tau_{11} & \tau_{21} & \dots & \tau_{p1} \\ \tau_{12} & \tau_{22} & \dots & \tau_{p2} \\ \dots & \dots & \dots & \dots \\ \tau_{1m} & \tau_{2m} & \dots & \tau_{pm} \end{vmatrix}.
\end{aligned}
\right\} \qquad (2.12)
$$

В матрицах системы (2.12) каждая колонка описывает параметры некоторой операции для различных машин, а каждая строка — показатели различных операций для одной УВМ. Поэтому некоторый показатель алгоритма для i-й машины определяется из показателей i-й строки. Компоненты τ_{ji} в системе (2.12) и далее считаются приведенными к норме.

Элементы матриц (2.12) представляют собой эффективность операций гипотетической машины и не учитывают их многократное повторение в алгоритме управления.

Для учета динамики этих операций в алгоритме управления составим матрицу-строку

$$L = |\{l_{j1}, l_{j2}\}| \qquad (j = 1, 2, \dots, p),$$

где l_{j1}, l_{j2} — число соответственно участков и обращений к j-й операции в программе.

Исходные матрицы (2.12) переводятся в оконечные, учитывающие динамику операций в алгоритме, умножением l_{j1} на $I\partial$, а l_{j2} — на $I\tau$. Элементы матриц $I\Delta$ и IK остаются без изменения и соответствуют элементам конечных матриц Δ и K.

Матрица $I\partial$ переводится в оконечную

$$
d = \begin{vmatrix} d_{11} & d_{21} & \dots & d_{p1} \\ d_{12} & d_{22} & \dots & d_{p2} \\ \dots & \dots & \dots & \dots \\ d_{1m} & d_{2m} & \dots & d_{pm} \end{vmatrix}
$$

в результате следующих преобразований,

1. Для совместимых операций, признаком которых является $\partial_{ji} = 1$, компонента $d_{ji} = l_{j1}\,\partial_{ji}$.

2. Для программно- и подпрограммносовместимых операций величина d_{ji} зависит от величины ∂_{ji} и от других признаков. Определим их.

Условно разобьем долговременную память машины на программное и подпрограммное поле. На программном поле наберем основную программу вычислений и программы p входа в подпрограмму, на подпрограммном поле — только подпрограммы P вычисления некоторых операций и программу q — выхода из подпрограммы.

При этом, если j-я операция i-й машины имеет длину программы ∂_{ji} и набирается на программном поле l_{j1} раз, то загрузка памяти машины этой операцией

$$d_{ji1} = l_{j1}\partial_{ji}. \tag{2.13}$$

В случае выполнения j-й операции в виде подпрограммы, необходимо учесть, кроме показателей ∂ и T основной программы операции, еще показатели ∂_p и τ_p программы входа в подпрограмму и показатели ∂_q и τ_q программы q выхода из подпрограммы.

Показатели программ p и q зависят от типа машины и числа исходных и оконечных результатов. В основном система команд машины содержит операции с одним входом по числу исходных данных и одним выходом по числу результатов. Поэтому можно считать, что показатели ∂_p, τ_p, ∂_q, τ_q для различных операций одной и той же машины имеют соответственно равные значения и не зависят от типа операций.

Таким образом, если j-я операция выполнена в виде подпрограммы, то загрузка этой операцией памяти машины

$$d_{ji2} = \partial_{ji} + \partial_{pi}l_{j1} + \partial_{qi}. \tag{2.14}$$

Для экономии памяти машины j-ю операцию набирают на программном поле, если выполняется неравенство

$$d_{ji1} \leqslant d_{ji2}. \tag{2.15}$$

При невыполнении условия (2.15) j-я операция должна быть выполнена в виде подпрограммы.

Подставив в неравенство (2.15) правые части выражений (2.13) и (2.14), получим

$$\partial_{ji} \leqslant \frac{\partial_{pi}l_{j1} + \partial_{qi}}{l_{j1} - 1}. \tag{2.16}$$

Если j-я операция имеет $\partial_{ji} > 1$, и она не несовместима, то признаком программносовместимости будет выполнение условия (2.14) и ее компонента d_{ji} определится из формулы (2.13). Если условие (2.14) не выполняется, то j-я операция является подпрограммносовместимой и ее показатель d_{ji} определится из выражения (2.16). В частном случае при $l_{j1} = 1$ из формулы (2.14) имеем $\partial_{ji} = \infty$, и j-ю операцию следует набирать на программном поле.

Матрица $I\tau$ переводится в оконечную

$$t = \begin{vmatrix} t_{11} & t_{21} & \dots & t_{p1} \\ t_{12} & t_{22} & \dots & t_{p2} \\ \dots & \dots & \dots & \dots \\ t_{1m} & t_{2m} & \dots & t_{pm} \end{vmatrix}$$

с использованием следующих соотношений для операций:

совместимых

$$t_{jl} = l_{j2}\tau_{jl};$$

программносовместимых

$$t_{jl} = l_{j2}\tau_{jl};$$

подпрограммносовместимых

$$t_{jl} = l_{j2}(\tau_{jl} + \tau_{pl} + \tau_{ql}).$$

Таким образом, для перевода матриц $I\partial$ и It в оконечные необходимо иметь численные значения ∂_{pi}, τ_{pi}, ∂_{qi}, τ_{qi}, которые легко определить для каждой машины. Сведем их в матрицу

$$P = \begin{vmatrix} \partial_{p1} & \partial_{q1} & (\tau_{p1} + \tau_{q1}) \\ \partial_{p2} & \partial_{q2} & (\tau_{p2} + \tau_{q2}) \\ \dots & \dots & \dots \\ (\partial_{pm} & \partial_{qm} & (\tau_{pm} + \tau_{qm}) \end{vmatrix}.$$

Если некоторая операция имеет число входов и выходов больше единицы, то в матрицу P следует ввести ее показатели p и q. Перевод компонент исходных матриц $I\partial$ и It в оконечные показан на блок-схеме рис. 3.

Определим показатели эффективности алгоритма для каждой машины. Объем долговременной памяти D_i^0 машины определится из длины программы алгоритма и числа необходимых констант. Длина программы для i-й машины опре-

деляется через компоненты i-й строки оконечной матрицы d

$$D_i = \sum_{j=1}^{p} d_{ji}.$$

Система констант алгоритма для i-й машины

$$K_i = \bigcup_{j=1}^{p} K_{ji} = \{K_{i\xi}\}.$$

Из $\{K_{i\xi}\}$, где $\xi = 1, 2, \ldots, \psi_i$, определяем число требуемых констант.

Общая загрузка памяти машины алгоритмом управления определяется внутрикортежной связью

$$D_i^0 = D_i + \psi_i.$$

Время реализации алгоритма

$$T_i = \sum_{j=1}^{p} t_{ji}.$$

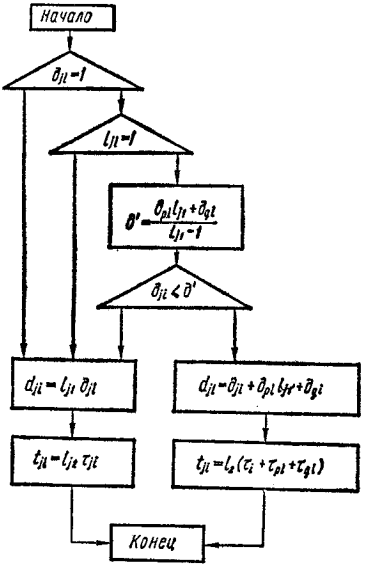


Рис. 3.

Для оценки погрешности алгоритма Δ_i компоненты матрицы Δ используются в той функциональной зависимости, которая составлена для вычисления ошибки оконечных результатов и включает в себя, кроме Δ_{ji}, ошибку выбранного численного метода, ошибки представления исходных данных и др. Принимая погрешности численного метода и исходных данных постоянными для любой машины, получаем ошибку алгоритма

$$\Delta_i = f(\Delta_{ji}),$$

где показатели Δ_{ji} для i-й машины выбирают из i-й строки и подставляют в функциональную зависимость.

Полученные показатели эффективности алгоритма управления для каждой машины сводятся в кортежи

$$A_i = <\Delta_i, T_i, D_i^0>$$

($i = 1, 2, \ldots, m$).

Из полученных m вариантов выбирают УВМ, у которой $T = T_{\min} < T_{\text{доп}}$ при условии, что $D_i^0 \leqslant D_{i\text{доп}}$ и $\Delta_i \leqslant \leqslant \Delta_{\text{доп}}$.

Процесс выбора варианта эффективной реализации алгоритма управления на одной из m УВМ показан на рис. 4.

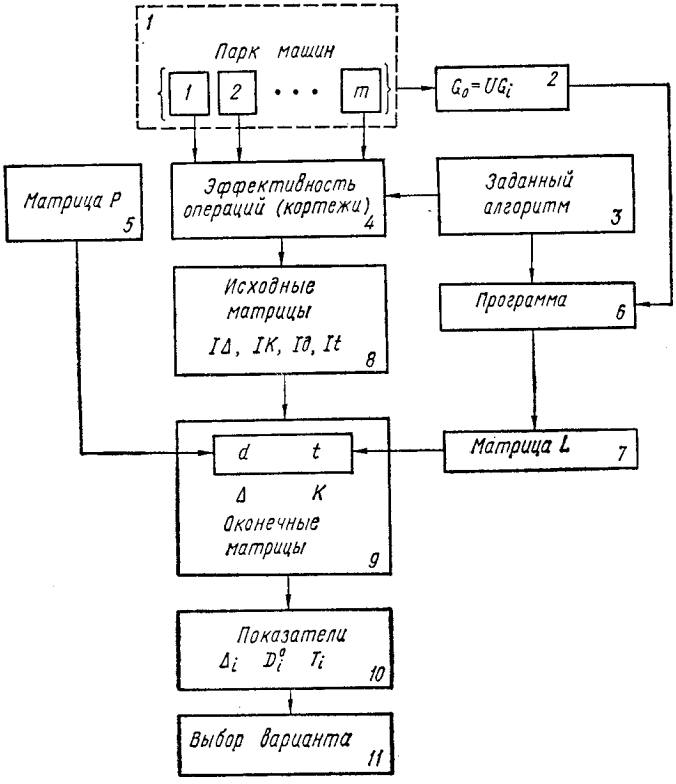


Рис. 4.

Исходными для решения задачи выбора оптимальной УВМ являются парк машин и заданный алгоритм управления (блоки *1* и *3*). Определение совместимости систем команд, оценка эффективности каждой операции СК G_0 и программ p и q для каждой машины, а также составление программы алгоритма осуществляется блоками подготовки *2*, *4—6*. Составление матрицы-строки L и формальные пре-

образования над показателями эффективности проводятся в блоках *7—10*. Блок *11* осуществляет выбор варианта.

При большом количестве вариантов (что значительно увеличивает объем вычислений) операции формальных преобразований выполняются в ЭЦВМ.

Глава III
РЕАЛИЗАЦИЯ АЛГОРИТМОВ В ДИСКРЕТНО-РАЗНОСТНОЙ ФОРМЕ НА УВС

1. АЛГОРИТМЫ, СОСТОЯЩИЕ ИЗ ОПЕРАЦИЙ СЛОЖЕНИЯ И УМНОЖЕНИЯ

Анализ вычислительных алгоритмов, встречающихся в системах управления с УВС, показывает, что большинство из них относится к одному из следующих классов:

1) алгоритмы дискретно-разностной формы; 2) алгоритмы в виде рядов и полиномов; 3) логические алгоритмы.

Алгоритмы первого и второго классов состоят в основном из операций сложения и умножения. Поскольку алгоритмы первого класса встречаются в системах управления с УВС чаще других и требуют при управлении большей частоты решений, остановимся более подробно на минимизации их с точки зрения времени реализации УВС. Представим алгоритмы дискретно-разностной формы в виде

$$y[n] = \sum_{i=0}^{k} a_i x[n-i] - \sum_{i=1}^{k} b_i y[n-i], \qquad (3.1)$$

где a_i, b_i — переменные коэффициенты; $x[n-i]$ — переменные входные величины.

Любую операцию можно записать в виде формальной микропрограммы, а любую микропрограмму — в регулярной форме [42]. Существует алгоритм для преобразования произвольных микропрограмм, записанных в обычном виде, в регулярную форму.

Распространим подобную запись на программы, учитывая, что операторы теперь будут состоять из операций.

Рассмотрим алгоритм, состоящий только из операций сложения и умножения, причем предполагаем, что, кроме первой операции умножения, каждая следующая порождает

одну операцию сложения (как в дискретно-разностных алгоритмах), т. е. только операции умножения являются порождающими [17].

В этом случае запись программы реализации такого алгоритма

$$Q = \{b_i b_j c a_t\}\,\{(b_t b_{t+1})(b_{t+2} s)\ldots\}, \qquad (3.2)$$
$$(i = 1 - k;\quad j = 1 - k;\quad t = 1 - k),$$

где b_i, b_j — операции вызова числа соответственно из i-й и j-й ячеек; c — операция умножения; a_t — операция пересылки результата операции c в ячейку с адресом t; k — количество слагаемых в алгоритме; s — операция сложения.

Фигурные скобки с индексами означают, что оператор $Q_1 = b_i b_j c a_t$ считается выполненным, когда j, i, t изменяют значения от 1 до k. После этого работает оператор

$$Q_2 = \{(b_t b_{t+1} s)(b_{t+2} s)\ldots\}.$$

Если в каком-либо операторе отсутствует операция a_t, то это означает, что результат действия этого оператора остается в арифметическом устройстве (АУ). Например, после выполнения оператора $Q_3 = b_t b_{t+1} s$ его результат остается в АУ, и после этого вызывается содержимое ячейки b_{t+2} и складывается с содержимым АУ (сумматора), что в свою очередь остается в АУ и т. д.

Нетрудно видеть, что запись программы (3.2) рассматриваемого алгоритма не является наилучшей. Целесообразно привести исходный алгоритм к такому виду, чтобы операции сложения и умножения порождали друг друга. Это обстоятельство позволяет исключить в выражении (3.2) операцию a.

Можно доказать, что всякий алгоритм, состоящий из операций сложения и умножения, в котором одна операция умножения порождает одну операцию сложения, приводится к виду

$$Q = \{(b_i b_j c)(b_{i-1} s)(b_{j-1} c)\ldots\} \qquad (3.3)$$
$$(i, j = 1, 2, \ldots, k).$$

Рассмотрим алгоритм в общем виде

$$y = x_1\varphi_1 + x_2\varphi_2 + \cdots + x_k\varphi_k. \qquad (3.4)$$

Когда $x_1\varphi_1$, $x_2\varphi_2$, ..., $x_k\varphi_k$ принимают вещественные значения, можно всегда найти такие числа, что

$$r_1 = \frac{x_2}{x_1};\quad r_2 = \frac{x_3}{x_2};\quad \ldots;\quad r_{k-1} = \frac{x_k}{x_{k-1}}.$$

Подобное представление переменных позволяет записать исходный алгоритм (3.4) в виде

$$y = (\cdots(\varphi_k r_{k-1} + \varphi_{k-1}) r_{k-2} + \cdots + \varphi_2) r_1 + \varphi_1) x_1, \quad (3.5)$$

в котором одна операция умножения порождает одну операцию сложения и наоборот.

Разместив в ячейках b_i величины φ_i и в ячейках b_j величины r_{j-1}, нетрудно убедиться, что запись программы (3.3) соответствует реализации алгоритма (3.5). Но алгоритм (3.5) является эквивалентным алгоритму (3.4).

Точно также находим

$$m_1 = \frac{\varphi_2}{\varphi_1}; \quad m_2 = \frac{\varphi_3}{\varphi_2}; \quad \ldots; \quad m_{k-1} = \frac{\varphi_k}{\varphi_{k-1}}.$$

В этом случае в ячейках b_i размещаются величины x_i, а в ячейках b_j — величины m_{j-1}.

Запись программы (3.3) алгоритма, состоящего из операций сложения и умножения, более экономична, так как исключается операция a_i.

На практике чаще всего встречаются алгоритмы, в которых x_i или φ_i являются постоянными коэффициентами. Тогда удобно искать отношения именно этих коэффициентов, поскольку переменные могут быть наперед неизвестными.

Если записать алгоритм (3.1), обозначая

$$b_1 = a_{k+1}, \; b_2 = a_{k+2}, \; \ldots, \; b_k = a_{2k},$$

то получаем

$$y[n] = a_0 x[n] + a_1 x[n-1] + \cdots + a_k x[n-k] - \\ - a_{k+1} y[n-1] - \cdots - a_{2k} y[n-k]. \quad (3.6)$$

На основании приведенных рассуждений алгоритм (3.6) запишем следующим образом:

$$y[n] = (\cdots(-y[n-k] r_{2k-1} - y[n-k-1]) r_{2k-2} - \\ - \cdots - y[n-1]) r_k - x[n-k]) r_{k-1} + \\ + x[n-k-1]) r_{k-2} + \cdots + x[n-1]) r_0 + x[n]) a_0, \quad (3.7)$$

где

$$r_0 = \frac{a_1}{a_0}; \quad r_1 = \frac{a_2}{a_1}; \quad \ldots; \quad r_{2k-1} = \frac{a_{2k}}{a_{2k-1}}.$$

Количество операций в алгоритмах (3.6) и (3.7) одно и то же, однако машинного времени на реализацию алгоритма (3.7) потребуется меньше, так как не затрачивается время на пересылку промежуточных результатов вычислений.

2. МАСШТАБИРОВАНИЕ АЛГОРИТМОВ В ДИСКРЕТНО-РАЗНОСТНОЙ ФОРМЕ

Повысить точность вычислений при неизменном времени реализации алгоритмов на УВС можно, применяя соответствующие приемы масштабирования. Для этого масштабы на всех этапах работы УВС необходимо выбирать так, чтобы изображения чисел в машине представлялись максимальным количеством разрядов [39]. Рассмотрим некоторые приемы масштабирования, использующие автоматическое изменение масштаба в зависимости от значений промежуточных результатов.

Пусть необходимо вычислить функцию $y = f(x)$, причем аргумент x есть некоторая случайная величина. Очевидно масштабировать надо по максимальному значению промежуточного результата z, получаемого в процессе вычисления для данного x. Один из способов для определения $z_i = f(x)$ состоит в исследовании функционального преобразования $y = f(x)$ и выражения z_i как функции не только от x, но и от y.

Рассмотрим эффективные приемы масштабирования алгоритмов типа (3.1) при реализации их на УВС с фиксированной запятой. Выберем масштабные коэффициенты $m = 2^{\pm l}$ ($l = 0, 1, 2...$) и определим максимальное значение суммы (3.1)

$$z(nT) = f_z[x(n-i)T,\ y(n-i)T].$$

Предположим, что для a_i и b_i введены постоянные масштабы для всего процесса вычислений от $n = 1$ до N. Для входных переменных $x[(n-i)T]$ найдем масштаб

$$m_x(nT) = \frac{1}{z(nT)},$$

который для всех $x[(n-i)T]$ при вычислении только одного значения $y[nT]$ является постоянным.

Для нахождения коэффициента сдвига l_0 входных переменных достаточно $z(nT)$ представить в нормализованном виде $z(nT) = d2^l$, тогда порядок l и будет искомым коэф-

фициентом сдвига. При этом, если $l < 0$, осуществляем сдвиг всех $x[(n-i)T]$ $(i = 0, 1, 2, \ldots, k)$ на l разрядов влево, в противном случае — на l разрядов вправо.

Исследуем алгоритм (3.1), когда $x(t)$ — ступенчатая функция, и запишем его в виде

$$y_n = \sum_{i=0}^{k} a_i x_{n-i} - \sum_{i=1}^{r} b_i y_{n-i},$$

где y_n описывает переходный процесс и его установившееся значение найдем из выражения

$$y_{\text{уст}} = x \sum_{i=0}^{k} a_i - y_{\text{уст}} \sum_{i=1}^{r} b_i, \tag{3.8}$$

или

$$y_{\text{уст}} = \frac{x \sum_{i=0}^{k} a_i}{1 + \sum_{i=1}^{r} b_i}.$$

Для определения максимального значения в процессе вычислений рассмотрим процесс суммирования попарных произведений $a_i x_{n-i}$, $b_i y_{n-i}$. Пусть суммирование происходит в следующем порядке:

$$\begin{aligned}
&S_0 = a_0 x_n;\\
&S_0 = a_0 x_n + a_1 x_{n-1} = S_0 + a_1 x_{n-1};\\
&S_2 = S_1 + a_2 x_{n-2};\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&S_k = S_{k-1} + a_k x_{n-k};\\
&S_{k+1} = S_k - b_1 y_{n-1};\\
&S_{k+2} = S_{k+1} - b_2 y_{n-2};\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&S_{k+r} = S_{k+r-1} - b_r y_{n-r} = y_n.
\end{aligned}$$

Промежуточная сумма S_j $(j = 0, 1, 2, \ldots, k + r)$ изменяется с изменением j и в некоторой точке принимает максимальное значение z_n, причем этот максимум будет либо при $0 \leqslant j \leqslant k$, либо при $k + 1 \leqslant j \leqslant k + r$.

Промежуточная сумма принимает свое наибольшее значение при суммировании первой части алгоритма, если

$$y_{\text{уст}} < x \sum_{i=0}^{k} a_i,$$

или

$$\frac{\sum_{i=0}^{k} a_i}{1 + \sum_{i=1}^{r} b_i} < \sum_{i=0}^{k} a_i; \quad \gamma = \left| 1 + \sum_{i=1}^{r} b_i \right| > 1.$$

Если же $\gamma < 1$, то промежуточная сумма достигает максимального значения при $k + 1 \leqslant j \leqslant k + r$.

При выполнении алгоритма (3.8) S_i образуются суммированием коэффициентов. Обозначим максимальную промежуточную сумму при сложении коэффициента a через S_a, а при сложении коэффициентов b — через S_b. Тогда при $\gamma > 1$

$$z_{n1} = xS_a, \tag{3.9}$$

при $\gamma \leqslant 1$

$$z_{n2} = x \sum_{i=0}^{k} a_i - y_{\text{уст}} S_b. \tag{3.10}$$

Если x — медленно изменяющийся процесс, то

$$z_{n1} = x_{\max} S_a; \tag{3.11}$$

$$z_{n2} = x_n \sum_{i=0}^{k} a_i - y_{\max} S_b, \tag{3.12}$$

где $x_{\max}$ — максимальное по модулю из значений x_n, x_{n-1}, ..., x_{n-k}; $y_{\max}$ — максимальное по модулю из значений y_{n-1}, ..., y_{n-r}.

Полученные таким образом величины z_n отражают только процесс сложения попарных произведений. Учитывая, что произведение чисел меньших единицы меньше любого из сомножителей, поэтому для того, чтобы не получить z_n меньше входных переменных, необходимо при $S_a < 1$ (или $S_b < 1$) вместо S_a (или S_b) в формулы (3.9) — (3.12) подставлять единицу.

Если x (t) изменяется достаточно медленно, то

$$z_{n1} = x_{n-1} S_a; \tag{3.13}$$

$$z_{n2} = x_{n-1} \sum_{i=0}^{k} a_i - y_{n-1} S_b. \tag{3.14}$$

На рис. 5, *а*, *б* показаны идеальная (*1*) и расчетная (*2*) кривые, рассчитанные по формуле (3.14) для $x = 1\,(t)$, а на рис. 5, *б* для $x = \sin \pi t$ с шагом квантования $\Delta t = 0{,}1$.

Как видно из графиков, только для больших значений y_n имеем $z_{\text{расч}}\,(nT) > z\,(nT)$, а при малых значениях $y_n - z_{\text{расч}}\,(nT) < z\,(nT)$. Для того чтобы этого избежать, необходимо в области малых значений y_n ограничить $z_{\text{расч}}(nT)$ снизу на некотором заранее выбранном уровне, который больше или равен $z\,(nT)_{\min}$. Также и формулы (3.11) и (3.12) дают оценки, которые всегда больше истинного $z\,(nT)$. Но процедура выборки максимального из ряда чисел весьма длительна и занимает много места в программе. Для медленно изменяющихся сигналов можно брать максимальные значения из двух крайних точек, т. е.

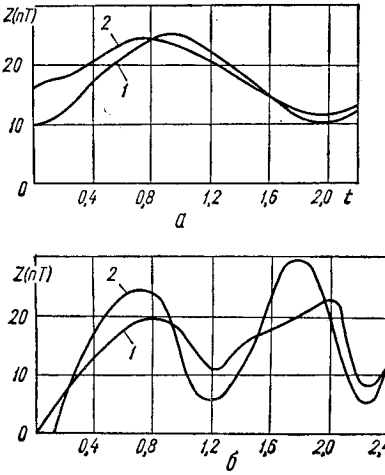


Рис. 5.

$$x_{\max} = \max\,(x_n, x_{n-k}); \quad y_{\max} = \max\,(y_{n-1}, y_{n-r}).$$

Как показали опыты, рассчитанное значение $z\,(nT)$ больше истинного максимального.

Для увеличения точности необходимо минимизировать значение $z\,(nT)$, но сделать его меньше максимальной величины из ряда

$$\{a_0, a_1, \ldots, a_k, b_1, b_2, \ldots, b_r, x_n, \ldots, x_{n-k},$$
$$y_{n-1}, \ldots, y_{n-r}\}$$

не представляется возможным.

Минимизировать $z\,(nT)$ можно, изменив определенным образом порядок суммирования попарных произведений при вычислении выражения (3.1). Если x постоянная или медленно изменяющаяся функция, то порядок суммирования задается, исходя из минимума промежуточной суммы при

суммировании коэффициентов. Для получения $S_{a\,\min}$ первым берут максимальный по модулю коэффициент $a_{\max}$, затем — максимальный из коэффициентов противоположных по знаку $a_{\max}$, далее коэффициент выбирают так, чтобы промежуточная сумма их не превышала значение $a_{\max}$. Если это удается осуществить, то $S_{a\,\max} = a_{\max}$.

При минимизации может получиться $S_{a\,\min} < a_{\max}$ и $S_b = b_{\max}$, но при вычислениях все же необходимо подставлять $S_a = a_{\max}$, $S_b = b_{\max}$.

Задача получения наименьшей промежуточной суммы ряда чисел может быть сведена к задачам линейного и динамического программирования, как задача оптимального распределения этих чисел в смысле минимума максимального выброса.

Зависимость $z\,(nT) = f\,(x,\ y)$ получают и другим способом. Рассмотрим несколько первых значений $y\,[n]$, заменив в них величины $y\,[n-j]$ значениями $y\,[n-j-1]$; $y\,[n-j-2]$; ... $y\,[1]$; $y\,[0]$.

Обозначая $\varphi\,[n] = \varphi_n$, имеем

$$y_0 = a_0x_0;$$

$$y_1 = a_0x_1 + (a_1 - a_0b_1)\,x_0;$$

$$y_2 = a_0x_2 + (a_1 - a_0b_1)\,x_1 + [a_2 - (a_1 - a_0b_1)\,b_1 - a_0b_2]\,x_0;$$

$$y_3 = a_0x_3 + (a_1 - a_0b_1)\,x_2 + [a_2 - (a_1 - a_0b_1)\,b_1 - a_0b_2]\,x_1 + \{a_3 - [a_2 - (a_1 - a_0b_1)\,b_1 - a_0b_2]\,b_1 - (a_1 - a_0b_1)\,b_2 - a_0b_3\}\,x_0;$$

$$y_4 = a_0x_1 + (a_1 - a_0b_1)\,x_3 + [a_2 - (a_1 - a_0b_1)\,b_1 - a_0b_2]\,x_2 + \{\ \}\,x_1 + \{a_4 - (\ )\,b_1 - [\ ]\,b_2 - (\ )\,b_3 - a_0b_4\}\,x_0,$$

где

$$c_0 = a_0;$$

$$c_1 = (\ ) = a_1 - c_0b_1;$$

$$c_2 = [\ ] = a_2 - c_1b_1 - c_0b_2;$$

$$c_3 = \{\ \} = a_3 - c_2b_1 - c_1b_2 - c_0b_3;$$

$$c_4 = (\ ) = a_4 - c_3b_1 - c_2b_2 - c_1b_3 - c_0b_4.$$

Находим следующие значения функции с учетом введенных обозначений:

$$y_5 = c_0x_5 + c_1x_4 + c_2x_3 + c_3x_4 + c_4x_1 - (c_4b_1 + c_3b_2 + c_2b_3 + c_1b_4)\,x_0;$$

$$y_6 = c_0x_6 + c_1x_5 + c_2x_4 + c_3x_3 + c_4x_2 - c_5x_1 - \\ - (c_4b_2 - c_3b_3 + c_2b_4 - c_5b_1)\,x_0;$$

$$y_7 = c_0x_7 + c_1x_6 + c_2x_5 + c_3x_4 + c_4x_3 - c_5x_2 - c_6x_1 - \\ - (c_4b_3 + c_3b_4 - c_6b_1 - c_5b_2)\,x_0;$$

$$y_8 = c_0x_8 + c_1x_7 + c_2x_6 + c_3x_5 + c_4x_4 - c_5x_3 - c_6x_2 - \\ - c_7x_1 - (c_4b_4 - c_5b_3 - c_6b_2 - c_7b_1)\,x_0,$$

где

$$c_5 = c_4b_1 + c_3b_2 + c_2b_3 + c_1b_4;$$
$$c_6 = c_4b_2 + c_3b_3 + c_2b_4 - c_5b_1;$$
$$c_7 = c_4b_3 + c_3b_4 - c_6b_1 - c_5b_2;$$
$$c_8 = c_4b_4 - c_5b_3 - c_6b_2 - c_7b_1.$$

Продолжая, получаем:

$$y_9 = c_0x_9 + c_1x_8 + c_2x_7 + c_3x_6 + c_4x_5 - c_5x_4 - c_6x_3 - \\ - c_7x_2 - c_8x_1 + c_9x_0;$$

$$y_{10} = c_0x_{10} + c_1x_9 + c_2x_8 + c_3x_7 + c_4x_6 - c_7x_3 - c_8x_2 + \\ + c_9x_1 + c_{10}x_0;$$

$$y_{11} = c_0x_{11} + c_1x_{10} + c_2x_9 + c_3x_8 + c_4x_7 - c_5x_6 - c_6x_5 - \\ - c_7x_4 - c_8x_3 + c_9x_2 + c_{10}x_1 + c_{11}x_0$$

и так далее.

Записываем выражение в общем виде для функции $y\,[n] = F\,(c_n\ x\,[n])$

$$y\,[n] = \sum_{i=0}^{\nu} c_i x\,[n-i] - \sum_{j=\nu+1}^{2\nu} c_j x\,[n-j] + \\ + \sum_{m=2\nu+1}^{3\nu} c_m x\,[n-m] - \sum_{p=3\nu+1}^{4\nu} c_p x\,[n-p] + \cdots, \quad (3.15)$$

где

$$c_i = a_i - \sum_{q=0}^{\nu-1} c_q b_{i-q};$$

$$c_m = \sum_{j=\nu+1}^{2\nu} c_j b_{2\nu+1-j} - \sum_{q=1}^{m-2\nu} c_{m-q} b_{m-2\nu-q};$$

$$c_p = \sum_{m=\nu+1}^{3\nu} c_m b_{3\nu+1-m} - \sum_{q=1}^{p-3\nu} c_{p-q} b_{p-2\nu-q}.$$

Исследуем поведение $y[n]$ при действии на интегрирующий и дифференцирующий контуры постоянного напряжения с амплитудой, равной единице (для аналогии исследуем переходной процесс в контуре) при

$$y[n] = \sum_{i=0}^{\nu} c_i - \sum_{j=\nu+1}^{2\nu} c_j + \sum_{m=2\nu+1}^{3\nu} c_m - \sum_{p=3\nu+1}^{4\nu} c_p + \cdots \quad (3.16)$$

Как известно, в интегрирующем и дифференцирующем контурах при $n \to \infty$ $y[n] = \text{const}$ и $y[n] \approx y_{\max}$ при малых значениях n.

Если $a_i \leqslant 1$, $b_i \leqslant 1$, то из выражения (3.15) следует

$$c_i > c_j > c_m > c_p > \cdots,$$

откуда видно, что $y[n] \to y_{\max}$ среди первых $\nu + 1$ значений. Действительно,

$$y[\nu+2] = \sum_{i=0}^{\nu} c_i - \sum_{i=1}^{\nu} c_i b_{k+1-i} < y[\nu+1];$$

$$y[\nu+2] < y[\nu+3] = \sum_{i=0}^{\nu} c_i - \sum_{i=1}^{\nu} c_i b_{\nu+1-i} + c_{\nu+2} b_2 < y[\nu+1]$$

и так далее.

Обозначим

$$Q_{\text{пр}0} = c_0;$$
$$Q_{\text{пр}1} = c_0 + c_1;$$
$$Q_{\text{пр}2} = c_0 + c_1 + c_2;$$
$$Q_{\text{пр}3} = c_0 + c_1 + c_2 + c_3;$$
$$Q_{\text{пр}4} = c_0 + c_1 + c_2 + c_3 + c_4,$$

тогда $y[n]_{\max} = [Q_{\text{пр}i}]_{\max}$.

Если вычисление $y[n]$ проводить по схеме

$$y[n] = a_0 x[n] - b_1 y[n-1] + a_1 x[n-1] - b_2 y[n-2] + \cdots - b_n y[n-k] + a_n x[n-k],$$

то максимальное промежуточное число, получаемое в сумматоре УВС, не может быть больше величины

$$z_n = |b_{j\max}[Q_{\text{пр}}]_{\max}| + |a_{i\max} x_{\max}|. \quad (3.17)$$

Это значение z_n и необходимо взять в качестве масштабного коэффициента из условия переполнения разрядной сет-

ки машины

$$M = \frac{1}{z_n}.$$

На рис. 6 показаны графики изменения $x\,[n]$ и $y\,[n]$, вычисляемые по приведенному алгоритму, а $z_{маш}\,[n]$ — машинное и $z_{расч}\,[n]$ — расчетное получены из выражения (3.17). Кривые рис. 6, *а* рассчитаны при $a_i = b_i = 0{,}5$; кривые рис. 6, *б* — при $a_0 = 0{,}5$; $a_1 = -0{,}5$; $a_2 = 0{,}5$;

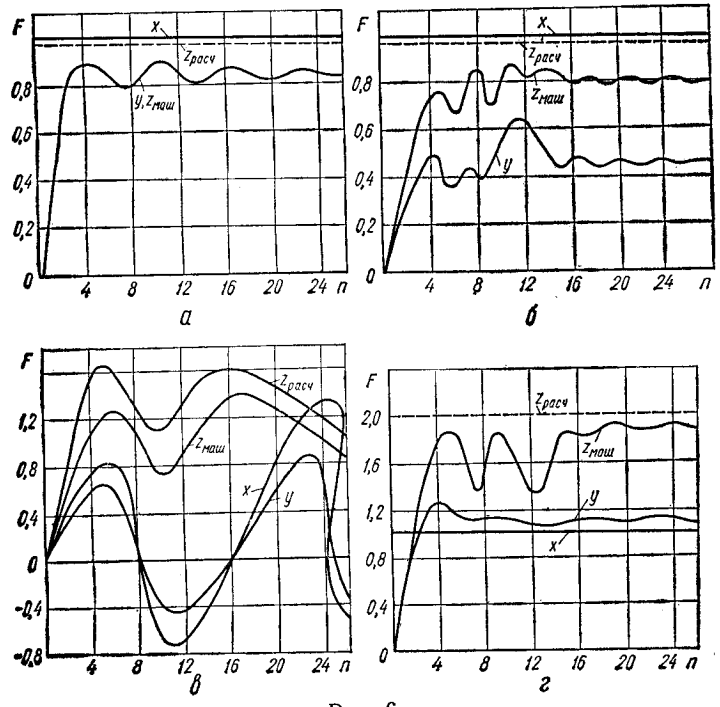


Рис. 6.

$a_3 = -0{,}5$; $a_4 = 0{,}5$; $b_1 = -0{,}5$; $b_2 = -0{,}5$; $b_3 = -0{,}5$; $b_4 = 0{,}5$; кривые рис. 6, *в* — при $a_0 = 1$; $a_1 = 0{,}6$; $a_2 = 0{,}2$; $a_3 = 0{,}06$; $a_4 = 0{,}02$; $b_1 = 0{,}8$; $b_2 = 0{,}4$; $b_3 = 0{,}2$; $b_4 = 0$; кривые рис. 6, *г* — $a_0 = 1$; $a_1 = -0{,}6$; $a_2 = 0{,}2$; $a_3 = -0{,}06$; $a_4 = 0{,}02$; $b_1 = -0{,}8$; $b_2 = 0{,}4$; $b_3 = -0{,}2$; $b_4 = 0{,}1$. Кривые $z_{маш}$ являются оптимальными с точки зрения получения повышенной точности вычислений и отсутствия переполнения разрядной сетки УВС.

Определим величины ошибок, которые получаются при принятом способе масштабирования, учитывая, что машины работают не с самими числами, а с их изображениями

$$x_i = 2^{-l} x_i.$$

Оценим ошибку при умножении, так как эта операция основная при вычислении алгоритма (3.1), и выведем зависимость этой ошибки от масштаба, в котором изображаются переменные x и y.

Максимальная ошибка умножения числа a_i на число x_{n-i}

$$\delta_i = \frac{2^{-n-1}}{a_i x_{n-i}} = \frac{\Delta}{a_i 2^{-l} x_i} = 2^l \frac{\Delta}{a_i x_i}.$$

Если считать, что погрешность сложения равна нулю и не учитывает ошибку представления чисел в машине, так как она постоянна и от принятого масштаба не зависит, то максимальная относительная ошибка одного просчета

$$\delta_y = \sum_{i=0}^{k+r} \delta_i = 2^l \Delta \left(\frac{1}{a_0 x_n} + \frac{1}{a_1 x_{n-1}} + \cdots + \frac{1}{b_r y_{n-r}} \right) =$$

$$= 2^l \Delta \psi (x, y),$$

максимальная относительная ошибка при определении $y\,[n]$

$$\delta_n = \sum_{l=1}^{n} \delta_{yl}.$$

Если выбрать постоянный масштаб для всего процесса вычисления $y\,[n]$ ($n = 1, 2, \ldots, N$), то, приняв функцию $\psi\,(x, y) = \text{const}$ (что не вносит существенных погрешностей в определение максимальной ошибки), запишем

$$\delta_n = n 2^l \Delta \psi (x, y).$$

Если для вычисления каждого значения $y\,[n]$ применить свой оптимальный масштаб 2^{-l_n}, где $l_n \leqslant l$, то получим меньшее значение ошибки. Например, при вычислении зависимости (см. рис. 5, *б*) во избежание переполнения разрядной сетки УВС необходимо выбрать постоянный масштаб $l = 2$, тогда

$$\delta_n = 4n\Delta\psi.$$

Если же при вычислении каждого $y[n]$ выбрать масштаб, ориентируясь на $z[n]$, то для десяти точек

$$\delta_{10} = 1 \cdot 2^0\Delta\psi + 4 \cdot 2^1\Delta\psi + 5 \cdot 2^2\Delta\psi = 29\Delta\psi$$

и, считая функцию $z[n]$ периодической с периодом 10 точек, получаем

$$\delta_n = 29\Delta\psi \frac{n}{10} = 2{,}9\Delta\psi.$$

Таким образом, для уменьшения ошибок вычисления дискретно-разностных алгоритмов перед получением каждого значения $y[n]$ необходимо вычислять масштабные коэффициенты.

Глава IV

СИНТЕЗ АЛГОРИТМОВ ПОВЫШЕННОЙ ТОЧНОСТИ ВЫЧИСЛЕНИЙ ДЛЯ ПРОСТЫХ ОПЕРАЦИЙ ПРИ ОГРАНИЧЕННОЙ ДЛИНЕ РАЗРЯДНОЙ СЕТКИ УВМ

1. ОСНОВНЫЕ ЗАДАЧИ СИНТЕЗА

Работа управляющей вычислительной машины связана с реализацией некоторого набора управляющих алгоритмов. При этом каждый управляющий алгоритм с целью обеспечения необходимой эффективности управления требует различной разрядности управляющей машины. Это связано, прежде всего, с получением различной требуемой точности вычислений выходной информации. На разрядность УВМ влияют: выбор численного метода решения; объем вычислений; величины погрешностей, вносимые элементарными алгоритмами (операциями), составляющими алгоритм управления. Кроме того, для одного и того же алгоритма на некоторых участках управления возникает необходимость вычисления с повышенной точностью. Для эффективного управления разрядность УВМ следует рассчитывать из условия обеспечения максимальной точности для всех алгоритмов на всем этапе управления. Однако этот путь не всегда является приемлемым, так как рост

разрядной сетки неизбежно ведет к увеличению экономических и габарито-весовых показателей УВМ.

Если рост разрядности приводит к нежелательным экономическим затратам и размерам УВМ, то длина разрядной сетки определяется из условия обеспечения максимальной одинарной точности, а повышенная точность (точность, превышающая разрядность УВМ) реализуется программным путем.

При введении алгоритмов, реализующих повышенную точность вычислений, в общее математическое обеспечение УВМ следует учитывать, что эти алгоритмы требуют для своей реализации значительно большего времени и объема памяти УВМ. Кроме того, для введения этих алгоритмов необходимо произвести некоторые изменения и дополнения в структуре и системе команд УВМ. Естественно, что чем шире возможность изменения структур алгоритмов и их показателей, тем выше степень оптимизации и эффективнее управление.

2. АЛГОРИТМЫ ПРОСТЫХ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ

Разработка алгоритмов повышенной точности вычислений для простых операций является одной из важной, так как эти алгоритмы представляют собой основу построения более сложных алгоритмов. От их показателей существенным образом зависят показатели алгоритма управления и всей системы управления в целом. Число многократной точности Ф в машинах с фиксированной запятой определено как число, состоящее из знака числа и правильной дроби из k последовательных слов памяти [162]. Если основание системы счисления $q = 2$, а каждое слово вычислительного устройства включает n двоичных разрядов, то

$$\Phi = S_\Phi, \sum_{i=1}^{k} \varphi_i 2^{-in},$$

где S_Φ — знак числа Ф (для $\Phi > 0$ $S_\Phi = 0$; $\Phi < 0$ $S_\Phi = 1$); φ_i — целая цифровая n-разрядная i-я часть числа Ф (слово).

Если число Ф представлено в одном из кодов (прямом, обратном, дополнительном), то в общем виде его можно записать

$$[\Phi] = S_\Phi, \sum_{i=1}^{k} \varphi_i' 2^{-in},$$

где φ_i' — целая цифровая n-разрядная i-я часть числа [Ф] (слово).

Каждое слово φ_i' хранится в отдельной ячейке памяти машины.

Исследуем некоторые возможные алгоритмы, реализующие простые операции с повышенной точностью. Для каждого алгоритма будем оценивать погрешность метода реализации данной операции при условии, что ошибка представления исходных данных равна нулю. Данный показатель характеризует погрешность, вносимую самим алгоритмом.

Сложение длинных чисел характеризуется тем, что складываются последовательно слова, начиная с младших. Если при сложении слов возникает единица переноса, то она прибавляется к младшему разряду суммы соседних старших слов. Таким образом, при сложении длинных чисел возникает необходимость в анализе сумм отдельных слов с целью выявления единицы переноса в соседнюю старшую часть суммы, или в запоминании значения переноса в специальном разряде с последующим его добавлением к сумме старших частей.

Пусть имеем два числа

$$[\Phi] = S_\Phi, \sum_{i=1}^{k} \varphi_i' 2^{-in};$$

$$[\Psi] = S_\Psi, \sum_{i=1}^{k} \psi_i' 2^{-in}.$$

Запишем сложение i-х слов в виде итерационной последовательности

$$\varphi_i' + \psi_i' + p_i = p_{i-1} + (\Phi + \Psi)_i^* \qquad (i = k, k-1, \ldots, 2, 1), \tag{4.1}$$

где p_i — значение переноса от сложения $(i+1)$-х слов; p_{i-1} — значение переноса от сложения i-х слов; $(\Phi + \Psi)_i^*$ — значение суммы i-х слов.

При сложении слов $i = 1$ возникший перенос p_0 от сложения цифровых частей поступает в знаковые разряды. При этом

$$S_\Phi + S_\Psi + p_0 = z + S_{(\Phi+\Psi)}, \tag{4.2}$$

где $S_{(\Psi+\Phi)}$ — знак суммы $(\Phi + \Psi)$; z — значение переноса от сложения слов $i = 1$ совместно со знаками.

Если длинные числа складываются в обратных кодах, то z (значение циклического переноса) прибавляется к младшему разряду полученной суммы длинного числа. Это представляется в виде следующей итерационной последовательности:

$$(\Phi + \Psi)_i^* + p_i = p_{i-1} + (\Phi + \Psi)_i \qquad (i = k, k-1, \ldots, 2, 1), \tag{4.3}$$

где $(\Phi + \Psi)_i$ — значение суммы i-х слов после учета циклического переноса.

Для $i = k$ имеем $p_k = z$. Если длинные числа складываются в дополнительном коде, то значение z не учитывается и последовательность (4.3) отсутствует.

Каждое слово длинного числа представляется в машине цифровой частью и своим знаком в прямом, обратном или дополнительном кодах. Обозначим через

$[S_{\varphi i}, \varphi_i]$ ($[S_{\psi_i}, \psi_i]$) — i-е слово числа Φ (Ψ), представленное в одном из кодов, где S_{φ_i} (S_{ψ_i}) соответствует знаку S_{φ_1} (S_{ψ_1}) первого слова.

В общем случае, если слагаемые Φ и Ψ представлены в прямом, обратном или дополнительном кодах, а сложение происходит в обратном или дополнительном коде, то сложение длинных чисел представляется следующими итерационными последовательностями.

Итерации сложения i-х слов:

для $i = k$

$$[S_{\varphi_1}, \varphi_k] + r_\Phi + [S_{\psi_1}, \psi_k] + r_\Psi = p_{k-1} + 0, \ (\Phi + \Psi)_k^*; \tag{4.4}$$

для $i = k-1, \ k-2, \ \ldots, \ 3, 2$

$$[S_{\varphi_1}, \varphi_i] + [S_{\psi_1}, \psi_i] + p_i = p_{i-1} + 0, \ (\Phi + \Psi)_i^*; \tag{4.5}$$

для $i = 1$

$$[S_{\varphi_1}, \varphi_1] + [S_{\psi_1}, \psi_1] + p_1 = z + S_{(\Phi+\Psi)_1}, \ (\Phi + \Psi)_1^*. \tag{4.6}$$

Итерации учета циклического переноса и присвоения знаков младшим словам имеют вид:

для $i = k, k-1, \ldots, 3, 2$

$$S_{(\Phi+\Psi)_1} \to S_{(\Phi+\Psi)_i} \text{ (знаку } S_{(\Phi+\Psi)_i} \text{ присвоить знак } S_{(\Phi+\Psi)_1}); \tag{4.7}$$

$$[S_{(\Phi+\Psi)_1}, (\Phi + \Psi)_i^*] + p_i = p_{i-1} + S_{(\Phi+\Psi)_1}, \ (\Phi + \Psi)_i \ (p_k = z); \tag{4.8}$$

для $i = 1$

$$[S_{(\Phi+\Psi)_1}, (\Phi + \Psi)_1^*] + p_1 = S_{(\Phi+\Psi)_1}, \ (\Phi + \Psi)_1, \qquad (4.9)$$

где обозначены слова сумм i-х слов без учета циклического переноса. О величинах r_Φ и r_Ψ будет сказано ниже.

Покажем, как изменяются последовательности (4.4) — (4.9) в зависимости от кодов представления чисел в машине. Пусть числа Φ и Ψ представлены в памяти машины в прямом коде $[\Phi]_{\text{пр}}$, $[\Psi]_{\text{пр}}$, а сложение их происходит в обратном коде. Тогда, принимая r_Φ и r_Ψ равными нулю, выполняем последовательности (4.4) — (4.9), где отдельные i-е слова, вызванные из памяти машины, переводятся в обратный код в арифметическом устройстве УВС. Погрешность такого алгоритма $\Delta = 0$.

Если исключить циклический перенос, однако оставить выражение (4.7), то погрешность такого алгоритма $\Delta \leqslant \leqslant 2^{-kn}$. В обоих случаях итерации (4.7) должны остаться, так как слова сумм записываются в память УВС в прямом коде и должны содержать знак $S_{(\Phi+\Psi)}$.

Сложение чисел $[\Psi]_{\text{пр}}$ и $[\Phi]_{\text{пр}}$ в дополнительном коде исключает итерации циклического переноса (4.8), (4.9). Кроме того, перевод $[\Phi]_{\text{пр}}$ и $[\Psi]_{\text{пр}}$ в дополнительный код требует перевода их отдельных слов в обратный с добавлением к их младшему разряду k-го слова некоторой величины r. При этом $r = 0$, если слагаемое имеет положительный знак, и $r = 2^{-kn}$ — если отрицательный. Погрешность алгоритма $\Delta = 0$. Если не учитывать r_Φ и r_Ψ, то погрешность алгоритма $\Delta \leqslant 2 \cdot 2^{-kn}$.

Пусть теперь Φ и Ψ хранятся в памяти машины в обратном коде ($[\Phi]_{\text{обр}}$, $[\Psi]_{\text{обр}}$). При сложении этих чисел в данном коде исключается перевод их отдельных слов в обратный код, а величины r_Φ и r_Ψ равны нулю. Выполняя последовательности (4.4)—(4.9), получаем алгоритм с $\Delta = 0$. Если младшие слова φ_i и ψ_i хранить с положительным знаком и исключить циклический перенос, то при разработке алгоритма можно пользоваться только итерациями (4.4) — (4.6). Тогда погрешность алгоритма составит $\Delta \leqslant 2^{-kn}$.

Наиболее простые алгоритмы получаются для чисел Φ и Ψ, которые представлены в памяти УВС в дополнительном коде, при условии, что сложение их происходит в этом же коде. Здесь $r_\Phi = 0$ и $r_\Psi = 0$. Последовательности (4.7) — — (4.9) исключаются, так как отсутствует циклический

перенос, а младшие слова можно хранить с положительным знаком. Для сравнения приведем эти последовательности:
для $i = k,\ k-1,\ \ldots,\ 2$

$$0,\ \varphi_i' + 0,\ \psi_i' + p_i = p_{i-1} + 0,\ (\Phi + \Psi)_i' \qquad (p_k = 0); \quad (4.10)$$

для $i = 1$

$$S_{\varphi_1},\ \varphi_1' + S_{\psi_1},\ \psi_1' + p_1 = S_{(\Phi+\Psi)_1},\ (\Phi + \Psi)_1'.$$

Погрешность алгоритма (4.10) $\Delta = 0$.

Вычитание длинных чисел производится через сложение уменьшаемого и вычитаемого с обратным знаком. Поэтому для вычитания используем последовательности (4.4) — (4.9), где слова вычитаемого заменяем на $[-S_{\psi_i},\ \psi_i]$ $(i = 1, 2, \ldots, k)$. Кроме того, в выражении (4.4) r_Ψ заменяется на его отрицание $(\bar{r}_\Psi)$, так как меняется знак вычитаемого. Итерации (4.7) — (4.9) остаются без изменения.

Ниже приводим итерации вычитания только для дополнительного кода с использованием системы (4.10):
для $i = k,\ k-1,\ \ldots,\ 2$

$$0,\ \varphi_i' + 0,\ [\psi_i']_{\text{обр}} + p_i = p_{i-1} + 0,\ (\Phi - \Psi)_i' \qquad (p_k = 2^{-kn}); \quad (4.11)$$

для $i = 1$

$$S_{\varphi_1},\ \varphi_1' + [-S_{\psi_1},\ \psi_1']_0 + p_1 = S_{(\Phi-\Psi)_1},\ (\Phi - \Psi)_1'.$$

Переходим к рассмотрению алгоритмов умножения длинных чисел. Пусть цифровые части сомножителей представлены в прямом коде

$$\Phi = \sum_{i=1}^{k} \varphi_i 2^{-in};$$

$$\Psi = \sum_{j=1}^{k} \psi_j 2^{-jn}.$$

Умножение двух отдельных слов длинных чисел дает произведение удвоенной разрядности

$$(\varphi_i 2^{-in})(\psi_j 2^{-jn}) = P_{i+j-1} 2^{-(i+j-1)n} + \Pi_{i+j} 2^{-(i+j)n}, \quad (4.12)$$

где P_{i+j-1}, Π_{i+j} — соответственно старшая и младшая части произведения (слова).

Произведение длинного числа Ψ на отдельное слово φ_i дает длинное частичное произведение, которое получается

путем последовательного умножения ψ_j ($j = k,\ k-1,\ \ldots,$ $\ldots,\ 1$) на φ_i и сложения полученных частичных произведений. Это представляется совместно в итерационной последовательности:

$$(\varphi_i 2^{-in})(\psi_j 2^{-jn}) + P_{i+j}2^{-(i+j)n} = P_{i+j-1}2^{-(i+j-1)} + \\ + \Pi_{i+j}2^{-(i+j)n} \qquad (j = k,\ k-1,\ \ldots,\ 2,\ 1;\ P_{i+k} = 0). \quad (4.13)$$

Реализация выражения (4.13) дает длинное частичное произведение

$$\pi_i = (\varphi_i 2^{-in})\,\Psi = P_i 2^{-in} + \sum_{j=1}^{k} \Pi_{i+j}2^{-(i+j)n}, \qquad (4.14)$$

где P_i — старшее слово; Π_{i+j} — младшее слово π_i.

Обозначим P_i через Π_i. Тогда формулу (4.14) перепишем следующим образом:

$$\pi_i = \sum_{j=0}^{k} \Pi_{i+j}2^{-(i+j)n},$$

или, обозначая $i + j$ через ξ,

$$\pi_i = \sum_{\xi=i}^{i+k} \Pi_\xi 2^{-\xi n}. \qquad (4.15)$$

Для $i = k$ из выражения (4.15) имеем первое длинное частичное произведение

$$\pi_k = \sum_{\xi=k}^{2k} \Pi_\xi 2^{-\xi n}. \qquad (4.16)$$

Просуммируем длинные частичные произведения

$$\pi_k,\ \pi_{k-1},\ \ldots,\ \pi_{i+2},\ \pi_{i+1}.$$

Получим сумму

$$S_{i+1}^{k} = \sum_{\nu=i+1}^{k} \pi_\nu = \sum_{\xi=i+1}^{2k} \Pi_\xi^{i+1} 2^{-\xi n}, \qquad (4.17)$$

где Π_ξ^{i+1} — отдельное слово суммы длинных частичных произведений (верхний индекс указывает на принадлежность этого слова к сумме S_{i+1}^{k}).

Очевидно, что сумма S_i^k определяется слагаемым S_{i+1}^{k} и i-м длинным частичным произведением:

$$S_i^k = S_{i+1}^{k} + \pi_i, \qquad (4.18)$$

где π_i определим из выражения (4.15):

Согласно формулам (4.17) и (4.18)

$$S_i^k = \sum_{\xi=i+1}^{2k} \Pi_\xi^{i+1} 2^{-\xi n} + \pi_i = \sum_{\xi=i}^{2k} \Pi_\xi^i 2^{-\xi n}.$$

Составим итерационную последовательность для получения суммы S_i^k. Для этого в левую часть равенства (4.13) введем дополнительно операцию сложения с соответствующим словом $\Pi_\xi^{i+1} = \Pi_{i+j}^{i+1}$.

Произведение длинных чисел найдем из итерационной последовательности

$$P_{i+j} 2^{-(i+j)n} + \Pi_{i+j}^{i+1} 2^{-(i+j)n} + (\varphi_i 2^{-in})(\psi_j 2^{-jn}) = \\ = P_{i+j-1} 2^{-(i+j-1)n} + \Pi_{i+j}^{i} 2^{-(i+j)n}. \qquad (4.19)$$

Для каждого фиксированного $i = k,\ k-1,\ \ldots,\ 2,\ 1$ параметр j принимает значения $k,\ k-1,\ \ldots,\ 2,\ 1$.

Начальным условием данной итерационной последовательности является $i = j = k$. Кроме того, в процессе проведения итераций (включая также и первую) для любого i и $j = k$ значение $P_{i+j} = 0$, что следует из равенства (4.13). Начальное значение Π_{i+j}^{i+1} определим из суммы частичных произведений (4.18), где для $i = k$

$$S_k^k = \pi_k.$$

Это соответствует первому длинному частичному произведению (4.16).

По формуле (4.17) для $i = k$ находим

$$S_{k+1}^k = \sum_{\xi=k+1}^{2k} \Pi_\xi^{k+1} 2^{-\xi n} = 0.$$

Отсюда слова Π_ξ^{k+1} равны нулю. Тогда в итерационной последовательности (4.19) для $i = k$ и $j = k,\ k-1,\ \ldots,\ 2,\ 1$ значения $\Pi_{i+j}^{k+1} = 0$.

Реализация итераций (4.19) дает произведение

$$S_1^k = \sum_{\xi=1}^{2k} \Pi_\xi^1 2^{-\xi n}.$$

Округляя полученный результат по старшему разряду слова $\Pi_{k+1}^{(1)}$, получаем

$$(\Phi\Psi) = \sum_{\xi=1}^{k} (\Phi\Psi)_\xi 2^{-\xi n}$$

с абсолютной ошибкой $\Delta < 0{,}5 \cdot 2^{-kn}$.

Заметим, что количество итераций в выражении (4.19) равно k^2. Однако это число можно значительно уменьшить, если отбросить частичные произведения

$$(\varphi_i \psi_l) < 2^{-(k+1)n}.$$

В этом случае число обращений к итерациям (4.19)

$$\frac{k}{2}(k+3) - 1.$$

Абсолютная погрешность такого произведения увеличится за счет отброшенных частичных произведений на величину $2^{-(k+1)n}(2k-3)$. Ошибка такого алгоритма после округления

$$\Delta < 0{,}5 \cdot 2^{-kn} + 2^{-(k+1)n}(2k-3).$$

Ввиду малости второго слагаемого считаем, что

$$\Delta < 0{,}5 \cdot 2^{-kn},$$

т. е. появляется возможность построения множества алгоритмов умножения с различными показателями Δ.

Например, отбрасывая n-разрядные члены частичных произведений меньше чем 2^{-kn}, получаем алгоритм с абсолютной погрешностью

$$\Delta < 2^{-kn}(2k-1)$$

и числом обращений к последовательности (4.19) значительно меньшим k^2. Если члены меньше чем 2^{-kn} представлять с округлением, то

$$\Delta < 2^{-kn}(1{,}5k-1).$$

Отбрасывая и округляя различные частичные произведения, получаем множество алгоритмов с различной эффективностью. При этом изменяется не только показатель Δ, но и число обращений к последовательности (4.19), а значит, и время реализации программы умножения длинных чисел.

Выше рассматривался вопрос изменения структуры алгоритма, которое достигалось выбором метода для получения различной погрешности алгоритма.

Однако для одной и той же структуры алгоритма можно построить программы с различной сложностью и с взаимно противоречивыми показателями. При этом сложность алгоритма зависит от системы команд, используемой в данном алгоритме, и определяет показатели объема памяти,

быстродействия, экономические и аппаратурные затраты на введение используемых команд.

Рассмотрим изменение структуры алгоритма в зависимости от выбора различной системы команд. Каждая итерационная последовательность характеризуется набором вполне определенных операций. Для их реализации на машине выбирают различные коды, которые могут изменяться по своему содержанию или дополняться новыми.

Для прямого кода представления чисел, вводя специальные команды сложения младших и старших слов, программно реализуем выражения (4.4) и (4.6):

$$\varphi_i \to \Sigma$$
$$\langle\Sigma\rangle \oplus \psi_i$$
$$\langle\Sigma\rangle \to (\Phi + \Psi)_i^*,$$

где Σ — сумматор; $\langle\Sigma\rangle$ — содержимое сумматора; $\to$ — оператор пересылки из ЗУ в сумматор или из сумматора в ЗУ; $\oplus$ — код специального сложения.

Выражения (4.7) — (4.9) реализуются программой:

$$\left.\begin{array}{l}(\Phi + \Psi)_i \to \Sigma \\ \text{ПЗ}\,(\Phi + \Psi)_1 \\ \langle\Sigma\rangle \oplus 0 \\ \langle\Sigma\rangle \to (\Phi + \Psi)_i,\end{array}\right\} \qquad (4.20)$$

где ПЗ — код операции присвоения знака числа $(\Phi + \Psi)_1$ содержимому сумматора.

Однако для улучшения показателей d и t алгоритма программу (4.20) можно реализовать с помощью одной специальной команды

$$\text{СПЗ}\,(\Phi + \Psi)_i,$$

где СПЗ — код операции сложения числа со значением переноса и присвоения результата знака числа, находящегося до этого на сумматоре.

Выражение (4.7) также может быть реализовано программой

$$\left.\begin{array}{l}(\Phi + \Psi)_i \to \Sigma \\ \text{ПЗ}\,(\Phi + \Psi)_1 \\ \langle\Sigma\rangle \to (\Phi + \Psi)_i\end{array}\right\} \qquad (4.21)$$

* Здесь и далее знаки слов S_Φ и S_Ψ, имеющиеся в выражениях (4.4) и (4.6), для упрощения опускаем.

или одной командой

$$\text{ПЗСП}\,(\Phi + \Psi)_i,$$

где ПЗСП — код операции присвоения слову памяти знака числа, находившегося до этого на сумматоре.

Для дополнительных кодов представления чисел в памяти машины составим алгоритм:

для $i = k$

$$\varphi_k \rightarrow \Sigma$$
$$\langle\Sigma\rangle \oplus \psi_k$$
$$\text{Сдв}\, n \rightarrow$$
$$|\langle \text{P}\rangle| \rightarrow (\Phi + \Psi)_k;$$

для $i = k - 1,\ k - 2,\ \ldots,\ 3,\ 2$

$$\langle\Sigma\rangle \oplus \varphi_i$$
$$\langle\Sigma\rangle \oplus \psi_i$$
$$\text{Сдв}\, n \rightarrow$$
$$|\langle \text{P}_r\rangle| \rightarrow (\Phi + \Psi)_i;$$

для $i = 1$

$$\langle\Sigma\rangle \oplus \varphi_1$$
$$\langle\Sigma\rangle + \psi_1$$
$$\langle\Sigma\rangle \rightarrow (\Phi + \Psi)_1,$$

где $\oplus$ — код операции сложения без переполнения; Сдв n — код операции переноса цифровых разрядов в регистр $\text{P}_\text{г}$, а значения p_{i-1} — в младший разряд Σ; $|\langle \text{P}_\text{г}\rangle|$ — код операции записи содержимого P_r по модулю в память машины.

Применяя специальный код записи $\langle\Sigma\rangle$ по модулю в память машины с последующим обнулением цифровых разрядов Σ и перенос значения p_{i-1} в младший разряд Σ (код $-|\langle\Sigma\rangle| \Rightarrow$), получаем алгоритм: для $i = k$

$$\left.\begin{array}{l} \varphi_k \rightarrow \Sigma \\ \langle\Sigma\rangle \oplus \psi_k \\ |\langle\Sigma\rangle| \Rightarrow (\Phi + \Psi)_k; \end{array}\right\} \qquad (4.22)$$

для $i = k,\ k - 2,\ \ldots,\ 3,\ 2$

$$\langle\Sigma\rangle \oplus \varphi_i$$
$$\langle\Sigma\rangle \oplus \psi_i$$
$$|\langle\Sigma\rangle| \Rightarrow (\Phi + \Psi)_i;$$

для $i = 1$

$$\langle\Sigma\rangle \oplus \varphi_k$$
$$\langle\Sigma\rangle + \psi_k$$
$$\langle\Sigma\rangle \to (\Phi + \Psi)_1.$$

Если ввести специальные коды сложения младших и старших частей с одновременным сложением значения p_{i-1}, которое хранится в специальном разряде [24, 129], то имеем:

для $i = k,\ k-1,\ \ldots,\ 3,\ 2$

$$\left.\begin{array}{l}\varphi_i \to \Sigma \\ \langle\Sigma\rangle \oplus_2 \psi_i \\ \langle\Sigma\rangle \to (\Phi + \Psi)_i;\end{array}\right\} \qquad (4.23)$$

для $i = 1$

$$\left.\begin{array}{l}\varphi_1 \to \Sigma \\ \langle\Sigma\rangle \oplus_1 \psi_1 \\ \langle\Sigma\rangle \to (\Phi + \Psi)_1,\end{array}\right\}$$

где $\oplus_2$ — код операции сложения младших частей с p_{i-1}; $\oplus_1$ — код операции сложения старших частей с p_2.

При составлении набора алгоритмов вычитания с повышенной точностью вводятся специальные команды вычитания [129] или используются коды, которые были введены для других алгоритмов.

Для алгоритмов умножения длинных чисел наиболее эффективным кодом, улучшающим параметры d и t, есть код, реализующий в системе (4.19) операции умножения и сложения (умножение с накоплением).

Анализ структур итерационных последовательностей показывает, что более эффективными при прочих равных условиях являются алгоритмы для обработки чисел в дополнительных кодах. Изменение структуры в зависимости от требуемой степени точности значительно влияет на показатели объема программы и быстродействия алгоритма.

Основной показатель (t) алгоритма определяется в зависимости от степени точности (k) следующим образом.

Алгоритм сложения имеет

$$t^k_{\text{сл}} = klt_0, \qquad (4.24)$$

где t_0 — время выполнения простой операции с одинарной точностью; l — число простых операций в одной итерации сложения.

Для алгоритма вычитания

$$t^k_{\text{выч}} = kbt_0, \tag{4.25}$$

где b — число простых операций в одной итерации вычитания.

Для прямого и обратного кодов представления чисел в памяти машины величины l и b содержат операции, учитывающие циклический перенос и присвоение знаков младшим частям.

Для алгоритмов умножения система итераций (4.19) повторяется k^2 раз, тогда

$$t^k_{\text{умн}} = k^2 mt_0 + (k+1)\,at_0, \tag{4.26}$$

где mt_0 — время выполнения одной итерации системы (4.19); a — число простых операций в одной итерации округления.

Если из алгоритма умножения исключить округление, то

$$t^k_{\text{умн}} = k^2 mt_0. \tag{4.27}$$

При повторении системы (4.19) $\frac{k}{2}(k+3) - 1$ раз получаем

$$t^k_{\text{умн}} = \left[\frac{k}{2}(k+3) - 1\right] mt_0. \tag{4.28}$$

Для приведенных алгоритмов вновь введенные специальные команды, реализующие одну и ту же операцию, рассматриваются как команды с незначительными изменениями их микропрограммной структуры. Каждая операция, входящая в алгоритм управления, потребует вполне определенных аппаратурных затрат. Поэтому при незначительных изменениях в микропрограммной структуре различных кодов одной и той же операции, приводящих к небольшим изменениям схем, аппаратурные затраты на видоизменение команды при описании эффективности алгоритма не учитываются.

Таким образом, изменяя структуру алгоритмов и выбирая различные коды операций, разрабатывают наборы алгоритмов, конкретное использование которых зависит от класса решаемых задач и определяется процессом оптимизации.

Следует различать рассмотренный метод изменения структуры алгоритма от выбранного численного метода решения задачи, так как для каждого численного метода удается построить множество структур с различными погрешностями. В то же время каждая структура в зависимости от кодов позволяет получить программы с различной эффективностью.

Процесс проектирования алгоритмов с различными численными методами, структурой и программой представляется в виде трех уровней:

Численный метод
↓
Структура
↓
Программа

На верхнем уровне рассматриваются различные численные методы решения задачи. Для каждого численного метода на среднем уровне составляются различные структуры. Нижний уровень представляет собой разработку программ с различной сложностью для каждой структуры.

3. АЛГОРИТМЫ ТИПА НАКОПЛЕНИЯ

Рассмотрим проектирование эффективных алгоритмов многократного сложения нескольких чисел или отдельных однозначных слов слагаемых для дополнительных кодов представления чисел в памяти машины. Однако общие принципы проектирования распространяются и на алгоритмы накопления суммы для прямого и обратного кодов представления чисел.

Сложение двух чисел с повышенной точностью требует программной реализации итерационной последовательности (4.10). При этом число команд, реализующих одну итерацию, может быть различно и зависит от выбранной структуры команд. Условимся, что действие одной команды программы связано с одним обращением к памяти машины. Тогда минимальная программа сложения двух i-х слов не может быть меньше трех команд. Примерами таких программ являются программы (4.22) и (4.23), в которых согласно выражению (4.24) при $l = 3$ имеем

$$t_{\text{сл}}^{k} = 3kt_0.$$

Для сложения трех чисел с k-значной точностью необходимо два раза обратиться к программе сложения двух чисел, тогда

$$t^k_{сл} = 2 \cdot 3kt_0.$$

Очевидно, что при сложении m чисел

$$t^k_m = (m-1)\,3kt_0. \qquad (4.29)$$

Преобразуем алгоритм (4.10), последовательно примененный

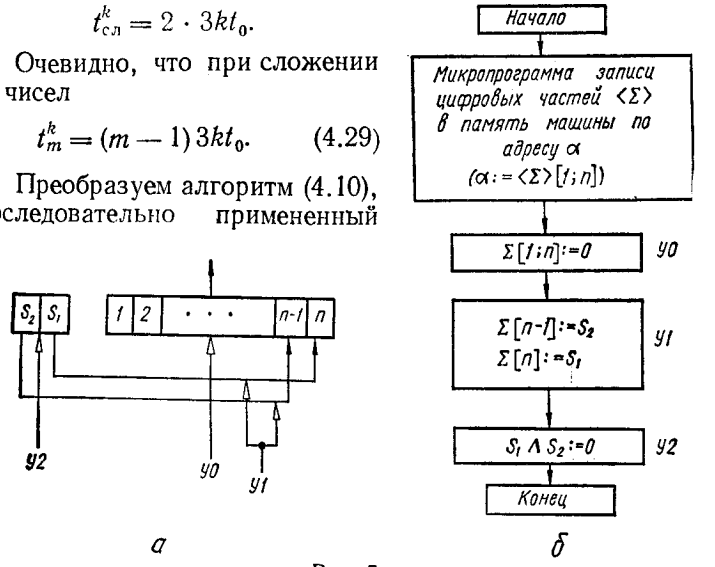


Рис. 7.

$m-1$ раз для сложения m чисел,

$$\sum_{j=1}^{m} \varphi_{ji} + P_{i+1} = P_i + \left(\sum_{j=1}^{m} \Phi_j\right)_i$$

$$(i = k, k-1, \ldots, 2, 1;\ P_{k+1} = 0), \qquad (4.30)$$

где P_{i+1} и P_i — значение переноса от сложения $(i+1)$ и i-х слов соответственно, при этом $P_i \leqslant m-1$.

Рассмотрим возможную реализацию такого вида алгоритма для

$$\sum_{j=1}^{m} \Phi_j < 1.$$

Пусть $m = 4$. Для реализации выражения (4.30) введем команды следующего содержания.

1. Специальное сложение (условное обозначение $\oplus$). По этой команде происходит сложение двух чисел по всем правилам сложения чисел в модифицированном дополнительном коде. Сигнал о переполнении разрядной сетки блокируется.

2. Специальная запись (условное обозначение $\|\Rightarrow$). Операционное устройство и микропрограмма специальной записи соответственно показаны на рис. 7, *а*, *б*, где *У*0, *У*1, *У*2 — управляющие сигналы).

Последовательно сложим однозначные слова четырех слагаемых, используя команду специального сложения:

$$\begin{array}{r} 00.\ 111\ \ldots\ 1111 \\ 00.\ 111\ \ldots\ 1111 \\ 00.\ 111\ \ldots\ 1111 \\ \underline{00.\ 111\ \ldots\ 1111} \\ 11.\ 111\ \ldots\ 1100 \end{array}$$

Сложение четырех максимальных по величине слов дает три единицы переноса, которые зафиксируются в знаковых разрядах.

Применяя к полученной сумме команду специальной записи ($\|\Rightarrow$), переписываем единицы переноса из знаковых разрядов в младшие разряды сумматора. Очевидно, что последующее сложение данного переноса со словами φ_{ji} четырех слагаемых может также дать не больше, чем три единицы переноса, которые также зафиксируются в знаковых разрядах. Продолжая таким образом последовательное сложение однозначных слов от $i = k$ до $i = 1$, записываем алгоритм для $\sum_{j=1}^{m} \Phi_j$:

для $i = k$

$$\begin{gathered} \varphi_{1k} \rightarrow \Sigma \\ \langle\Sigma\rangle \oplus \varphi_{2k} \\ \langle\Sigma\rangle \oplus \varphi_{3k} \\ \langle\Sigma\rangle \oplus \varphi_{4k} \\ \langle\Sigma\rangle \|\Rightarrow (\Phi_1 + \Phi_2 + \Phi_3 + \Phi_4)_k; \end{gathered}$$

для $i = k - 1,\ k - 2,\ \ldots,\ 3,\ 2$

$$\left.\begin{gathered} \langle\Sigma\rangle \oplus \varphi_{1i} \\ \langle\Sigma\rangle \oplus \varphi_{2i} \\ \langle\Sigma\rangle \oplus \varphi_{3i} \\ \langle\Sigma\rangle \oplus \varphi_{4i} \\ \langle\Sigma\rangle \|\Rightarrow (\Phi_1 + \Phi_2 + \Phi_3 + \Phi_4)_i; \end{gathered}\right\} \qquad (4.31)$$

для $i = 1$

$$\langle\Sigma\rangle \oplus \varphi_{11}$$
$$\langle\Sigma\rangle \oplus \varphi_{21}$$
$$\langle\Sigma\rangle \oplus \varphi_{31}$$
$$\langle\Sigma\rangle + \varphi_{41}$$
$$\langle\Sigma\rangle \to (\Phi_1 + \Phi_2 + \Phi_3 + \Phi_4)_1.$$

Сравним показатели t для алгоритмов (4.22), (4.23) и (4.31). Из алгоритма (4.31) следует, что для сложения четырех однозначных слов требуется пять простых операций. Тогда время сложения четы-

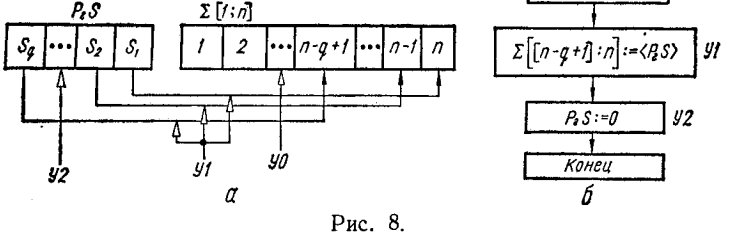


Рис. 8.

рех чисел с k-значной точностью с использованием метода накопления суммы

$$t_{4\text{н}}^{k} = 5kt_0. \qquad (4.32)$$

Для $k = 6$ согласно выражению (4.29) получим

$$t_4^6 = (4 - 1)\,3 \cdot 6t_0 = 54t_0.$$

Для алгоритма (4.31) из формулы (4.32) находим

$$t_{4\text{н}}^{6} = 5 \cdot 6 \cdot t_0 = 30t_0.$$

Абсолютный выигрыш во времени $\Delta t = 24t_0$, что примерно составляет 44,4%. В общем случае, если требуется накопить сумму $m > 4$ чисел, необходимо ввести q знаковых разрядов для фиксирования $m - 1$ возможных переносов. При этом q выбирается из условия

$$2^{q-1} < m \leqslant 2^q. \qquad (4.33)$$

Здесь вводится специальная команда засылки по q-знаковым разрядам ($q\Rightarrow$), которая осуществляет запись цифровых разрядов в память машины, а содержимое q-знаковых разрядов переписывает в младшие разряды сумматора. Операционная часть и микропрограмма команды $q\Rightarrow$ показаны соответственно на рис. 8, *а* и *б*.

Алгоритм сложения m чисел с k-значной точностью:

$$
\begin{aligned}
&\text{для } i = k \\
&\qquad \varphi_{1k} \to \Sigma \\
&\qquad \langle\Sigma\rangle \oplus \varphi_{jk} \quad (j = 2, 3, \ldots, m) \\
&\qquad \langle\Sigma\rangle\, q \Rightarrow \left(\sum_{j=1}^{m} \Phi_j\right)_k; \\
&\text{для } i = k-1, k-2, \ldots, 3, 2 \\
&\qquad \langle\Sigma\rangle \oplus \varphi_{ji} \quad (j = 1, 2, 3, \ldots, m) \\
&\qquad \langle\Sigma\rangle\, q \Rightarrow \left(\sum_{j=1}^{m} \Phi_j\right)_i; \\
&\text{для } i = 1 \\
&\qquad \langle\Sigma\rangle \oplus \varphi_{j1} \quad (j = 1, 2, 3, \ldots, m-1) \\
&\qquad \langle\Sigma\rangle + \varphi_{jm} \\
&\qquad \langle\Sigma\rangle \to \left(\sum_{j=1}^{m} \Phi_j\right)_1.
\end{aligned}
\qquad (4.34)
$$

Время сложения m чисел с k-значной точностью

$$t^k_{m\text{в}} = (m+1)\, kt_0. \qquad (4.35)$$

Абсолютный выигрыш во времени

$$\Delta t_m = t^k_m - t^k_{m\text{в}} = (m-1)\, 3kt_0 - (m+1)\, kt_0 = 2k\,(m-2)\, t_0.$$

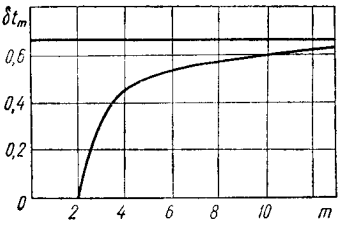


Рис. 9.

Относительный выигрыш

$$\delta t_m = \frac{\Delta t_m}{t^k_m} = \frac{2k\,(m-2)\, t_0}{3k\,(m-1)\, t_0} = \frac{2}{3}\left(\frac{m-2}{m-1}\right).$$

График зависимости относительного выигрыша от числа слагаемых показан на рис. 9. С ростом числа слагаемых

$$\lim_{m\to\infty} \delta t_m = \lim_{m\to\infty} \frac{2}{3}\left(1 - \frac{1}{m-1}\right) = \frac{2}{3} \approx 0{,}66\,(6).$$

Покажем относительный выигрыш в объеме программы. Для последовательного сложения чисел число команд

$$d_m^k = 3k(m-1).$$

Используя метод накопления сумм, имеем

$$d_{mн} = k(m+1).$$

Относительный выигрыш в объеме памяти

$$\delta d_m = \frac{3k(m-1) - k(m+1)}{3k(m-1)},$$

отсюда

$$\lim_{m\to\infty} \delta d_m = \lim \frac{2k - \frac{4k}{m}}{3k - \frac{3k}{m}} = \frac{2}{3} \approx 0{,}66.$$

Таким образом, при $m \to \infty$ относительный выигрыш в объеме памяти составляет 66%.

Если частичные суммы членов ряда превышают единицу, то алгоритм (4.34) дает верное решение при условии, что общий результат не переполняет разрядную сетку,

$$\sum_{j=1}^{m} \Phi_j < 1. \qquad (4.36)$$

Это обеспечивается введением специальной команды сложения, по которой блокируется сигнал о переполнении разрядной сетки ξ. Например, сложим четыре числа с удвоенной точностью:

$\Phi_1 = +0{,}1101\ 1111 \quad \varphi_{11} = 00{,}1101 \quad \varphi_{12} = 00{,}1111$
$\Phi_2 = +0{,}1110\ 0101 \quad \varphi_{21} = 00{,}1110 \quad \varphi_{22} = 00{,}0101$
$\Phi_3 = +0{,}0001\ 0110 \quad \varphi_{31} = 00{,}0001 \quad \varphi_{32} = 00{,}0110$
$\Phi_4 = -0{,}1110\ 0101 \quad \varphi_{41} = 11{,}0001 \quad \varphi_{42} = 00{,}1011$
$\Sigma = +0{,}11110101\ (\xi = 0).$

Алгоритм:

$$\varphi_{12} \to \Sigma$$
$$\langle\Sigma\rangle \oplus \varphi_{22}$$
$$\langle\Sigma\rangle \oplus \varphi_{32}$$
$$\langle\Sigma\rangle \oplus \varphi_{42}$$

$$\langle\Sigma\rangle \Vert\Rightarrow \left(\sum_{j=1}^{4} \Phi_j\right)_2$$
$$\langle\Sigma\rangle \oplus \varphi_{11}$$
$$\langle\Sigma\rangle \oplus \varphi_{21}$$
$$\langle\Sigma\rangle \oplus \varphi_{31}$$
$$\langle\Sigma\rangle + \varphi_{41}$$
$$\langle\Sigma\rangle \rightarrow \left(\sum_{j=1}^{4} \Phi_j\right)_1,$$

где $\langle\Sigma\rangle \Vert\Rightarrow$ — код специальной записи по $q = 2$ [см. алгоритм (4.34)].

Сложим числа согласно приведенному алгоритму:

$$\begin{array}{l}
\oplus\ \varphi_{12} = 00{,}1111 \\
\quad\ \varphi_{22} = 00{,}0101 \\
\hline
\quad\ \ 01{,}0100 \\
\oplus\ \varphi_{32} = 00{,}0110 \\
\hline
\quad\ \ 01{,}1010 \\
\oplus\ \varphi_{42} = 00{,}1011 \\
\hline
\quad\ \ 10{,}0101 \\
\Vert\Rightarrow \quad \left(\sum_{j=1}^{4} \Phi_j\right)_2 = 00{,}0101 \\
\oplus\ \langle\Sigma\rangle = 00{,}0010 \\
\quad\ \varphi_{11} = 00{,}1101 \\
\hline
\quad\ \ 00{,}1111 \\
\oplus\ \varphi_{21} = 00{,}1110 \\
\hline
\quad\ \ 01{,}1101 \\
\oplus\ \varphi_{31} = 00{,}0001 \\
\hline
\quad\ \ 01{,}1110 \\
+\ \varphi_{41} = 11{,}0001 \\
\hline
\quad\ \ 00{,}1111 \\
\rightarrow \quad \left(\sum_{j=1}^{4} \Phi_j\right)_1 = 00{,}1111.
\end{array}$$

В этом примере при сложении старших частей во второй и третьей частичной сумме возникло переполнение, однако сигнал $\xi = 1$ блокируется, так как сложение осуществля-

лось по специальной команде. Четвертая частичная сумма — есть старшая часть результата. Она не имеет переполнения и по команде обычного сложения машиной отрабатывается сигнал $\xi = 0$.

Если в формуле (4.30) условие (4.36) не выполняется, то для случая

$$2^q > \sum_{j=1}^{m} \Phi_j \geqslant 2^q - 1 \qquad (P_1 = m - 1) \qquad (4.37)$$

сигнал ξ о переполнении разрядной сетки равен нулю, так как знаковые разряды заполнены единицами.

Отсюда следует, что для правильного контроля переполнения разрядной сетки при выполнении условия (4.37) необходимо накапливать сумму $m - 1$ чисел. При этом $P_1 \leqslant m - 2$, а q выбирается из условия (4.33).

Данный метод накопления суммы применяется и для чисел Φ_j ($j = 1, 2, \ldots, m$), с которыми необходимо провести сложения и вычитания. Введение специальных команд вычитания позволяет свести время реализации к (4.35).

Для прямых и обратных кодов представления чисел алгоритм накопления суммы сложнее, так как требует учета суммарного циклического переноса и операции присвоения знака младшим частям.

4. АЛГОРИТМЫ ВЫЧИСЛЕНИЯ ЛИНЕЙНЫХ ФУНКЦИЙ

Во многих случаях алгоритмы управления и контроля требуют для своей реализации вычисления элементарных функций, представленных полиномами степени p, или различных многочленов, в состав которых входит линейная функция типа $\Lambda = ax + b$. Как правило, при вычислении на машине полиномы приводятся к схеме Горнера, где линейная функция Λ есть отдельный участок, к которому обращаются p раз.

Повышенная точность вычисления линейной функции может быть реализована алгоритмами умножения и сложения в отдельности. При этом время реализации линейной функции с учетом формул (4.24) и (4.27)

$$t_{\Lambda}^{k} = t_{\text{умн}}^{k} + t_{\text{сл}}^{k} = k^2 m t_0 + k l t_0 = k(km + l)\, t_0. \qquad (4.38)$$

Для уменьшения времени t_{Λ} воспользуемся методом накопления суммы однозначных слов частичных произведений и соответствующих слов коэффициента b.

Перемножение длинных чисел a и x дает k^2 частичных произведений удвоенной длины

$$(a_i 2^{-in})(x_j 2^{-jn}) = (ax)_{(i+j-1)} 2^{-(i+j-1)n} + (ax)_{(i+j)} 2^{-(i+j)n} \quad (4.39)$$
$$(i = 1, 2, \ldots, k;\ j = 1, 2, \ldots, k),$$

при сложении которых получаем произведение

$$П = \sum_{\varphi=1}^{2k} П_\varphi 2^{-\varphi n}.$$

Для $\varphi = 1, 2, \ldots, k$ формула (4.39) дает $\psi_\varphi = 2\varphi - 1$ слов частичных произведений. Кроме того, для $\varphi = k + 1$ число слов слагаемых $\psi_{k+1} = 2k - 1$.

Отбросив слова меньше чем $2^{-(k-1)n}$, сложим однозначные (одного значения φ) слова частичных произведений и слова коэффициента b по методу накопления суммы

$$P_\varphi 2^{-\varphi n} + \sum_{\psi=1}^{2\varphi-1} (ax)_{\varphi\psi} 2^{-\varphi n} + b_\varphi 2^{-\varphi n} =$$
$$= P_{\varphi-1} 2^{-(\varphi-1)n} + (ax + b)_\varphi 2^{-\varphi n} \quad (4.40)$$
$$(\varphi = k + 1, k, k - 1, \ldots, 2, 1).$$

В формуле (4.40) величина $P_{\varphi-1}$ определяет значение переноса от сложения слов φ частичных произведений, величины b и значения переноса P_φ от предыдущей итерации.

Для первой итерации ($\varphi = k + 1$) значение $P_\varphi = 0$, так как отсутствует нулевая итерация и ее перенос, а величина $b_\varphi = 0{,}10\ldots00$, которая введена для округления произведения по $(k + 1)$-му слову.

Максимальное число однозначных слов с учетом слова коэффициента b, подлежащих сложению, равно $\Psi = 2k$. Поэтому число знаковых разрядов q выбираем из условия

$$2^{q-1} < 2k \leqslant 2^q. \quad (4.41)$$

Реализуем выражение (4.39) и (4.40) с удвоенной точностью.

Из формулы (4.41) имеем $q = 2$. Схема вычисления выражения $ax + b$ показана на рис. 10. При накоплении суммы чисел A_2, A_4, A_7, b_2 возможны три единицы переноса. Поэтому для сохранения знака суммы (A_2 и A_4 — старшие части частичных произведений представлены со знаками) введем дополнительно третий знаковый разряд.

При составлении программы воспользуемся командой специальной записи ($\|\Rightarrow$), которую реализуем следующим образом.

1. Содержимое цифровых разрядов Σ переносится в память машины с положительным знаком.

2. Цифровые разряды Σ гасятся.

3. Содержимое знаковых разрядов Σ переносится в младшие разряды Σ. При этом если $\omega = 0$ (знак суммы положителен), то остальные разряды сумматора, включая также и знаковые, заполняются нолями; при $\omega = 1$ все разряды сумматора заполняются единицами.

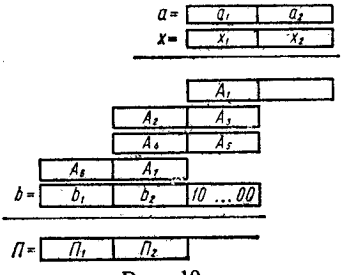


Рис. 10.

Введем две команды.

1. Специальное умножение без округления с сохранением младшей части произведения на регистре множителя-частного ($P_{мч}$). Условное обозначение $\otimes$.

2. Запись содержимого $P_{мч}$ с положительным знаком в память машины. Условное обозначение $|\langle P_{мч}\rangle|\rightarrow$.

Тогда алгоритм с использованием введенных команд (см. гл. IV, параграф 3):

$$
\begin{array}{ll}
a_1 \rightarrow \Sigma & \langle\Sigma\rangle \otimes x_2 \\
\langle\Sigma\rangle \otimes x_1 & \langle\Sigma\rangle \oplus A_3 \\
\langle\Sigma\rangle \rightarrow A_6 & \langle\Sigma\rangle \oplus A_5 \\
|\langle P_{мч}\rangle| \rightarrow A_7 & \langle\Sigma\rangle \oplus 0{,}10 \ldots 00 \\
a_2 \rightarrow \Sigma & \langle\Sigma\rangle \|\Rightarrow 0 \\
\langle\Sigma\rangle \otimes x_1 & \langle\Sigma\rangle \oplus A_2 \\
\langle\Sigma\rangle \rightarrow A_4 & \langle\Sigma\rangle \oplus A_4 \\
|\langle P_{мч}\rangle| \rightarrow A_5 & \langle\Sigma\rangle \oplus A_7 \\
a_1 \rightarrow \Sigma & \langle\Sigma\rangle \oplus b_2 \\
\langle\Sigma\rangle \otimes x_2 & \langle\Sigma\rangle \|\Rightarrow \Pi_2 \\
\langle\Sigma\rangle \rightarrow A_2 & \langle\Sigma\rangle \oplus A_6 \\
|\langle P_{мч}\rangle| \rightarrow A_3 & \langle\Sigma\rangle + b_1 \\
a_2 \rightarrow \Sigma & \langle\Sigma\rangle \rightarrow \Pi_1.
\end{array}
\qquad (4.42)
$$

Построим алгоритм, реализующий линейную функцию, с использованием итерационной последовательности (4.19).

Сохраняя итерации (4.19) прежними, при $i = k, k - 1, \ldots, 3, 2$ и $j = k, k - 1, \ldots, 2, 1$ вводим для итераций $i = 1$ операцию сложения слов величины b со словами произведения ax.

Тогда из формулы (4.19) имеем

$$P_{(1+j)}2^{-(1+j)n} + \Pi^{(2)}_{(1+j)}2^{-(1+j)n} + (\varphi_1 2^{-n})(\psi_j 2^{-jn}) + b_{(1+j)} = \\ = P_j 2^{-jn} + \Pi^{(1)}_{(1+j)}2^{-(1+j)n}, \qquad (4.43)$$

где b_{1+j} — слово величины b (для $j = k$ значение $b_{k+1} = 0{,}10\ldots00$ — константа округления по $(k+1)$-му слову функции Λ).

Максимальное количество чисел, которые приходится складывать в одной итерации, равно четырем. Этот случай соответствует $i = 1$. Число единиц переносов в знаковый разряд при сложении не больше трех. Поэтому для сохранения знака частичных произведений и их сумм достаточно иметь три знаковых разряда.

Реализуем выражения (4.19) и (4.43) с удвоенной точностью вычислений. Для этого используем следующие команды.

1. Посылка числа на $P_{\text{мч}}$ ($\langle \alpha \rangle \to P_{\text{мч}}$).
2. Запоминание ($P_{\text{мч}}$) ($|\langle P_{\text{мч}} \rangle| \to \alpha$).
3. Специальное сложение, примененное в формуле (4.31).
4. Умножение с накоплением. Здесь $\langle P_{\text{мч}} \rangle$ умножается на $\langle \alpha \rangle$ и складывается с накоплением с $\langle \Sigma \rangle$ ($\langle P_{\text{мч}} \rangle \otimes \langle \alpha \rangle + \langle \Sigma \rangle$). Алгоритм:

$a_2 \to \Sigma$	$\langle P_{\text{мч}} \rangle \otimes x_1 + \langle \Sigma \rangle$
$\langle \Sigma \rangle x_2$	$\langle \Sigma \rangle \oplus A_2$
$a_1 \to P_{\text{мч}}$	$\langle \Sigma \rangle \oplus b_2$
$\langle P_{\text{мч}} \rangle \otimes x_2 + \langle \Sigma \rangle$	$a_1 \to P_{\text{мч}}$
$\langle \Sigma \rangle \to A_2$	$\langle P_{\text{мч}} \rangle \otimes x_1 + \langle \Sigma \rangle$ (4.44)
$\lvert\langle P_{\text{мч}} \rangle\rvert \to A_3$	$\lvert\langle P_{\text{мч}} \rangle\rvert \to \Pi_2$
$A_3 \to \Sigma$	$\langle \Sigma \rangle + b_1$
$\langle \Sigma \rangle \oplus 0{,}10 \ldots 00$	$\langle \Sigma \rangle \to \Pi_1$.
$a_2 \to P_{\text{мч}}$	

В алгоритмах (4.42) и (4.44) на сложение с каждым словом коэффициента b затрачено по одной простой команде.

Для k-значной точности затрачено k простых команд сложения. Время реализации алгоритма линейной функции с использованием метода накопления

$$t^k_{\text{Ан}} = k^2 m t_0 + k t_0.$$

Абсолютный выигрыш во времени

$$\Delta t = t^k_{\text{А}} - t^k_{\text{Ан}} = k(l-1)\, t_0.$$

Алгоритмом линейной функции можно реализовать не только функции, приведенные к схеме Горнера, но и функции вида:

$$F = a_0 + \sum_{i=1}^{m} a_i b_i. \tag{4.45}$$

Приведем функцию (4.45) к линейной схеме

$$F = ((\cdots((a_0 + a_1 b_1) + a_2 b_2) + \cdots + a_m b_m). \tag{4.46}$$

В формуле (4.46) каждая скобка реализуется линейной функцией. Количество обращений к алгоритму линейной функции определяется порядком m.

Глава V

СИНТЕЗ АЛГОРИТМОВ ВЫЧИСЛЕНИЯ НЕКОТОРЫХ ФУНКЦИЙ ПРИ ОГРАНИЧЕННОЙ ДЛИНЕ РАЗРЯДНОЙ СЕТКИ УВМ

1. ФУНКЦИИ ПОВЫШЕННОЙ ТОЧНОСТИ ВЫЧИСЛЕНИЙ В КЛАССАХ СТЕПЕННЫХ РАЗЛОЖЕНИЙ И ИТЕРАЦИОННЫХ ПРОЦЕССОВ

Выбор численных методов вычисления элементарных функций [56, 57] зависит от специфики вычислительной машины и требований, предъявляемых к некоторым показателям алгоритма вычисления. Повышенная точность вычислений значительно ухудшает основные показатели t и d эффективности алгоритмов. Это обусловливается, прежде всего, большими соответствующими показателями алгоритмов простых операций повышенной точности, которые являются составляющими алгоритмов вычисления элементарных функций.

При вычислении элементарной функции на машине широко используются различные численные методы: разложение в ряд Тейлора-Маклорена, приближение разного рода многочленами, метод итераций и др. [53, 95]. Любой из этих вычислительных методов может быть разбит на отдельные повторяющиеся участки с однозначными операциями.

Пусть для такого участка составлена программа вычислений с k-значной точностью. Тогда при обращении к этой программе J_k раз общее время реализации

$$T = J_k t_{\text{уч}}^k, \tag{5.}$$

где $t_{\text{уч}}^k$ — время реализации отдельного участка с k-значной точностью.

Каждый участок состоит из простых операций (сложения, вычитания, умножения), которые также выполняются с повышенной точностью. Как было показано в гл. IV, время $t_{\text{уч}}^k$ тем больше, чем выше степень точности k. Поэтому величина T растет с увеличением k. Более того, с ростом степени точности растет и число обращений J_k к повторяющемуся участку. Например, с ростом точности в ряде Тейлора следует выбирать большее число членов разложения.

Очевидно, что значительный рост времени T приводит к нежелательным показателям эффективности t алгоритмов управления. В настоящей главе рассматриваются некоторые возможности уменьшения J_k и $t_{\text{уч}}^k$ в различных классах вычисления элементарных функций.

В классе степенных разложений Тейлора функция

$$f(x) \approx \sum_{j=0}^{p} \frac{f^{(j)}(\xi)}{j!} (x - \xi)^j.$$

Для знакочередующегося ряда остаточный член [6]

$$|R_p| \leqslant |a_{p+1}|, \tag{5.2}$$

где a_{p+1} — $(p+1)$-й член ряда.

Если все члены ряда одного знака, то

$$\int_{p+1}^{\infty} f(x)\,dx < R_p < \int_{p}^{\infty} f(x)\,dx.$$

Если члены ряда одного знака удовлетворяют условию

$$a_{j+1} \leqslant \frac{1}{2} a_j,$$

то конечный член ряда

$$a_p \geqslant \sum_{j=p+1}^{\infty} a_j.$$

Отсюда остаточный член

$$|R_p| \leqslant |a_p|. \tag{5.3}$$

Для степенных разложений элементарных функций, содержащих в знаменателе $j!$, в интервале изменения аргумента [0—1] при $p > 4$ условие (5.3) можно записать

$$|R_p| < |a_p|. \tag{5.4}$$

Конечный p член ряда для заданной абсолютной погрешности δ при выполнении неравенств (5.2) и (5.4) выбирается из условия

$$a_p > \delta > a_{p+1},$$

где a_p и a_{p+1} — максимальные значения членов на всем интервале изменения аргумента.

Реализуя $p + 1$ член разложения по схеме Горнера, согласно формулам (5.1) и (4.27) для $J_k = p$ получаем

$$T = pt_{\text{уч}}^k = p\,(t_{\text{сл}}^k + t_{\text{умн}}^k) =$$
$$= p\,(klt_0 + k^2 mt_0) = pk\,(l + km)\,t_0. \tag{5.5}$$

Пусть для n-разрядной сетки $\delta = 2^{-n}$, а для повышенной точности абсолютная ошибка метода $\delta = 2^{-kn}$. Обозначим через J_i — порядковые номера членов ряда, которые обеспечивают точность вычисления в (in)-м разряде $i = 1, 2, \ldots, \ldots, k$. Тогда

$$a_{J_i} > 2^{-in} > a_{J_i+1}. \tag{5.6}$$

Для достижения точности в (kn)-м разряде из формулы (5.6) следует:

$$\begin{aligned} &a_{J_1} > 2^{-n} > a_{J_1+1};\\ &a_{J_2} > 2^{-2n} > a_{J_2+1};\\ &\cdots\cdots\cdots\\ &a_{J_k} > 2^{-kn} > a_{J_k+1}, \end{aligned} \tag{5.7}$$

где $J_k = p$; $J_k + 1 = p + 1$.
Преобразуем систему (5.7):

$$\left.\begin{array}{l} a_0, \quad a_1, \quad a_2, \ldots, a_{J_1} > 2^{-n}; \\ 2^{-n} > a_{J_1+1}, \quad a_{J_1+2}, \ldots, a_{J_2} > 2^{-2n}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 2^{-(i-1)n} > a_{J_{i-1}+1}, \; a_{J_{i-1}+2}, \ldots, a_{J_i} > 2^{-in}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 2^{-(k-1)n} > a_{J_{k-1}+1}, \; a_{J_{k-1}+2}, \ldots, a_{J_k} > 2^{-kn}; \\ 2^{-kn} > a_{J_k+1}. \end{array}\right\} \quad (5.8)$$

Из выражений (5.8) видно, что члены $a_{J_{k-1}+1}$, $a_{J_{k-1}+2}$, ..., ..., a_{J_k} меньше значения $2^{-(k-1)n}$ и при выполнении условий (5.2) и (5.4)

$$\sum_{\varphi=J_{k-1}+1}^{J_k} a_\varphi < 2^{-(k-1)n}. \quad (5.9)$$

Таким образом, суммирование данных членов ряда можно проводить с одинарной точностью.

Для членов $i = k - 1$ с учетом выражения (5.9) записываем

$$\sum_{\varphi=J_{k-2}+1}^{J_k} a_\varphi < 2^{-(k-2)n}$$

и суммирование этих членов ряда следует проводить не ниже, чем с удвоенной точностью.

В общем случае для членов $a_{J_{i-1}+1}$, $a_{J_{i-1}+2}$, ..., a_{J_i} с учетом последующих членов

$$\sum_{\varphi=J_{i-1}+1}^{J_k} a_\varphi < 2^{-(i-1)n}. \quad (5.10)$$

Условие (5.10) показывает, что суммирование этих членов ряда можно проводить с точностью не ниже чем $(k - i + 1)$. Для $i = 1$ члены ряда a_0, a_1, ..., a_{J_1} суммируются с k-значной точностью.

Таким образом, вычисляя с k-значной точностью первые члены ряда, а остальные с точностью меньшей k, зна-

чительно улучшаем показатель t эффективности алгоритма. Вычисление степенного ряда следует начинать с младших членов с одинарной точностью. Затем переходят к удвоенной и так далее. Заканчиваются вычисления с k-значной точностью.

Для наиболее часто встречающейся схемы Горнера, если коэффициенты ее удовлетворяют условиям (5.8), то первые участки вычисляют с точностью меньше k. Каждый такой участок состоит из операций умножения и сложения

$$t_{\text{уч}}^{k-(i-1)} = t_{\text{сл}}^{k-(i-1)} + t_{\text{умн}}^{k-(i-1)} =$$
$$= [k-(i-1)]\{l + [k-(i-1)]m\} t_0.$$

Общее время реализации всего ряда

$$T = J_1 t_{\text{уч}}^{k} + (J_2 - J_1) t_{\text{уч}}^{k-1} + \cdots + (J_k - J_{k-1}) t_{\text{уч}}^{1} =$$
$$= J_1 k (l + km) t_0 + (J_2 - J_1)(k-1)[l + (k-1)m] t_0 +$$
$$+ \cdots + (J_k - J_{k-1})(l + m) t_0 = \{J_1 [l + (2k-1)m] +$$
$$+ \cdots + J_{k-2}(l + 5m) + J_{k-1}(l + 3m) + J_k (l + m)\} t_0,$$

откуда

$$T = \sum_{i=1}^{k} J_i \{l + [2(k-i) + 1]m\} t_0. \qquad (5.11)$$

При реализации ряда Маклорена по схеме Горнера для хранения постоянных коэффициентов

$$c_j = \frac{f^{(j)}(0)}{j!}$$

членов ряда $a_j = c_j x^j$ требуется объем долговременной памяти

$$D = (J + 1) k = (p + 1) k. \qquad (5.12)$$

Однако этот объем значительно уменьшается, если учесть, что для хранения констант $c_{J_{k-1}+1}, c_{J_{k-2}+2}, \ldots, c_{J_k}$ достаточно по одной ячейке [из системы (5.8) данные члены меньше $2^{-(k-1)n}$], для констант $c_{J_{k-2}+1}, c_{J_{k-2}+2}, \ldots, c_{J_{k-1}}$ достаточно по две ячейки, а для констант $c_0, c_1, c_2, \ldots, c_{J_1}$ необходимо по k ячеек.

Тогда объем долговременной памяти

$$D = (J_1 + 1) k + (J_2 - J_1)(k-1) + (J_3 - J_2)(k-2) +$$
$$+ \cdots + (J_i - J_{i-1})[k-(i-1)] + \cdots + (J_{k-1} +$$
$$+ J_{k-2}) 2 + (J_k - J_{k-1}) = k + J_1 + J_2 + \cdots + J_k,$$

или

$$D = k + \sum_{i=1}^{k} J_i. \qquad (5.13)$$

Например, для разрядной сетки $n = 8$ следует вычислить e^x с точностью $\delta \leqslant 2^{-32}$. Для достижения такой точности достаточно выбрать четырнадцать членов разложения ($p = 13$)

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{13}}{13!}$$

и вычисления проводить с учетверенной точностью ($k = 4$).

Если рассматривается интервал $0 \leqslant x < 1$, то суммирование членов a_{11}, a_{12}, a_{13} производится с одинарной точностью, так как

$$2^{-24} > a_{11}, a_{12}, a_{13} > 2^{-32} \qquad (J_4 = p = 13);$$

суммирование членов a_9, a_{10} — с удвоенной точностью, так как

$$2^{-16} > a_9, a_{10} > 2^{-24} \qquad (J_3 = 10);$$

суммирование членов a_6, a_7, a_8 — с утроенной точностью, так как

$$2^{-8} > a_6, a_7, a_8 > 2^{-16} \qquad (J_2 = 8);$$

суммирование членов a_0, a_1, a_2, a_3, a_4, a_5 — с учетверенной точностью, так как они больше 2^{-8} ($J_1 = 5$).

Объем долговременной памяти без экономии ячеек

$$D = (p + 1)k = 14 \cdot 4 = 56,$$

с экономией ячеек

$$D = 4 + \sum_{i=1}^{4} J_i = 40.$$

Время реализации оценим по формулам (5.5) и (5.11). Пусть $l = 3$ (см. гл. IV, 2), а $mt_0 = t_{сл} + t_{умн} + t_{зап}$. Примем, что $t_{умн} = 5t_0$, тогда $mt_0 = 1t_0 + 5t_0 + 1t_0 = 7t_0$. Согласно формуле (5.5)

$$T = pk(l + km)\, t_0 = 13 \cdot 4\,(3 + 4 \cdot 7)\, t_0 = 1612t_0.$$

Однако, если не все члены ряда вычислять с четырехзначной точностью, то согласно выражению (5.11)

$$T = \sum_{i=1}^{4} J_i \{3 + [2(4 - i) + 1]7\} = 5\,(3 + 49) +$$

$$+ 8(3 + 35) + 10(3 + 21) + 13(3 + 7) = 934t_0.$$

В классе различных многочленных приближений функция $f(x)$ представляется полиномом степени p

$$P_p(x) = c_0 + c_1x + \cdots + c_{p-1}x^{p-1} + c_px^p.$$

Если для коэффициентов c_j $(j = 0, 1, \ldots, p)$ составить систему

$$2^{-(i-1)n} > \sum_{\varphi=J_{i-1}+1}^{J_k} c_\varphi > 2^{-in} \qquad (i = k-1, k-2, \ldots, 1), \tag{5.14}$$

то все вычисления данного полинома при $0 \leqslant x < 1$ нецелесообразно проводить с k-значной точностью, иначе это увеличивает время реализации. Система (5.14) позволяет установить, какие члены и с какой степенью точности их следует вычислять. Очевидно, что члены, удовлетворяющие условию (5.14), вычисляют с $[k - i + 1]$-й степенью точности. Объем долговременной памяти может быть определен по формуле (5.13), а время реализации — по формуле (5.11).

В классе итерационных приближений

$$y_{j+1} = F(y_j),$$

погрешность каждой итерации

$$\delta_{j+1} = f(\delta_j).$$

Если для достижения заданной точности $\delta \leqslant 2^{-kn}$ потребуется p итераций, то составляется многоитерационная система повышенной точности:

$$\left.\begin{array}{l}
\delta_0, \delta_1, \delta_2, \ldots, \delta_{J_1} > 2^{-n}; \\
2^{-n} > \delta_{J_1+1}, \delta_{J_1+2}, \ldots, \delta_{J_2} > 2^{-2n}; \\
\ldots\ldots\ldots\ldots\ldots\ldots \\
2^{-in} > \delta_{J_i+1}, \delta_{J_i+2}, \ldots, \delta_{J_{i+1}} > 2^{-(i+1)n} \\
\ldots\ldots\ldots\ldots\ldots\ldots \\
2^{-(k-1)n} > \delta_{J_{k-1}+1}, \delta_{J_{k-1}+2}, \ldots, \delta_{J_k} > 2^{-kn}; \\
2^{-kn} > \delta_{J_k+1},
\end{array}\right\} \tag{5.15}$$

где J_i — число итераций.

Для δ_{J_i} и δ_{J_i+1} выполняется условие

$$\delta_{J_i} > 2^{-in} > \delta_{J_i+1}.$$

Из системы (5.15) следует, что итерации 1, 2, ..., J_1 нецелесообразно выполнять с k-значной точностью, так как в процессе этих итераций уточняются разряды, значения которых больше 2^{-n}. Поэтому эти итерации достаточно выполнить с одинарной точностью. По этой же причине итерации $j = J_1 + 1, J_1 + 2, \ldots, J_2$ следует выполнять с удвоенной точностью. В конечном приближении итерации $j = J_{k-1} + 1, J_{k-1} + 2, \ldots, J_k, J_{k+1}$ необходимо выполнять с k-значной точностью. Уменьшение степени точности вычисления первых итераций дает значительный выигрыш во времени.

Для обратной величины и корня квадратного итерационные процессы дают быструю сходимость $(\delta_{i+1} \leqslant (\delta_i)^2)$. Поэтому система (5.15) приводится к одноитерационной системе повышенной точности, значительно ускоряющей процесс приближения:

$$\left.\begin{aligned} \delta_0,\ \delta_1,\ \delta_2,\ \ldots,\ \delta_{\bar{J}_1} &> 2^{-n} \\ \delta_{\bar{J}_1+1} &> 2^{-2n} \\ \ldots\ldots\ldots & \\ \delta_{\bar{J}_1+q} &< 2^{-kn}. \end{aligned}\right\} \qquad (5.16)$$

Особенностью системы (5.16) является то, что она содержит по одной итерации с высшей степенью точности.

Пример. Вычислим обратную величину $Y = \frac{1}{x}$ с погрешностью $\delta \leqslant 2^{-32}$ на машине с разрядностью $n = 8$. Для итерационной формулы

$$y_{j+1} = y_j(2 - Xy_j)$$

относительные погрешности итераций находятся в зависимости

$$\bar{\delta}_{j+1} = (\bar{\delta}_j)^2. \qquad (5.17)$$

На интервале [0,5; 1] максимальные относительные и абсолютные ошибки по своему значению совпадают

$$\bar{\delta} = \frac{\delta}{y_{\min}};\quad y_{\min} = 1;\quad \bar{\delta} = \delta.$$

В соответствии с выражением (5.15) составим систему абсолютных погрешностей для наилучшей начальной константы приближения

$$y_0 = \frac{4}{3}\ (\delta_0 = 0{,}33).$$

Система абсолютных погрешностей:

$$\left.\begin{array}{rl} \delta_0 = 0{,}33\ (3); \quad \delta_1 = 0{,}11\ (1); \quad \delta_2 = 0{,}01234; & > 2^{-8}; \\ 2^{-8} > \delta_3 = 0{,}000154 & > 2^{-16}; \\ 2^{-24} > \delta_4 = 0{,}000000024 & > 2^{-32}; \\ \delta_5 = 0{,}58 \cdot 10^{-15} & < 2^{-32}. \end{array}\right\} \quad (5.18)$$

Приведем систему (5.18) к виду (5.16)

$$\left.\begin{array}{r} \delta_0,\ \delta_1,\ \delta_2,\ \delta_3 > 2^{-8}; \\ \delta_4 = 2^{-16}; \\ \delta_5 = 2^{-32}. \end{array}\right\}$$

Отсюда следует, что для достижения заданной точности достаточно провести три итерации с одинарной точностью, одну с удвоенной и одну с учетверенной.

Если выбрать в качестве начального приближения линейное

$$y_0 = \frac{48}{17} - \frac{32}{17}x,$$

то получаем систему:

$$\left.\begin{array}{ll} \delta_0 = 0{,}059 & > 2^{-8}; \\ \delta_1 = 0{,}00348 & > 2^{-16}; \\ \delta_2 = 0{,}12 \cdot 10^{-4} & > 2^{-24}; \\ \delta_3 = 0{,}15 \cdot 10^{-9} & < 2^{-32}. \end{array}\right\}$$

После преобразования имеем систему

$$\left.\begin{array}{r} \delta_0,\ \delta_1 = 2^{-8}; \\ \delta_2 = 2^{-16}; \\ \delta_3 = 2^{-32}, \end{array}\right\}$$

которая содержит одну итерацию с одинарной точностью, одну с удвоенной и одну с учетверенной.

Оценим время реализации метода итераций, если все p итераций вычислять с k-значной точностью

$$T = pt^k_{\text{итц}}, \quad (5.19)$$

где $t^k_{\text{итц}}$ — время выполнения одной итерации с k-значной точностью.

Для системы (5.15) время реализации

$$T = m_1 t^1_{\text{итц}} + m_2 t^2_{\text{итц}} + \cdots + m_k t^k_{\text{итц}} = \sum_{i=1}^{k} m_i t^i, \quad (5.20)$$

где m_i — число итераций в i-й строчке системы (5.15).

Для быстро сходящихся итераций ($\frac{1}{X}$, $\sqrt{X}$) составляется система (5.16), для которой время реализации

$$T = m_1 t^1_{\text{итц}} + t^2 + \cdots + t^k_{\text{итц}} = m_1 t^1_{\text{итц}} + \sum_{i=1}^{k} t^i_{\text{итц}}. \quad (5.21)$$

Определим показатель T для обратной величины. Одна итерация с k-значной точностью согласно формуле итерации

$$y_{j+1} = y_j (2 - X y_j)$$

имеет

$$t^k_{\text{итц}} = t^k_{\text{сл}} + 2t^k_{\text{умн}} = klt + 2k^2 m t_0.$$

Для $l = 3$ и $m = 7$

$$t_{\text{итц}} = (3k + 14k^2)\, t_0.$$

Из системы (5.18) следует, что $m_1 = 2$; $m_2 = 1$; $m_3 = 0$; $m_4 = 2$,
тогда

$$t^1_{\text{итц}} = (3 + 14)\, t_0 = 17t_0;$$

$$t^2_{\text{итц}} = (6 + 56)\, t_0 = 62t_0;$$

$$t^4_{\text{итц}} = (12 + 224)\, t_0 = 236t_0.$$

Выполняя все итерации с четырехзначной точностью $\left(y_0 = \frac{4}{3}\right)$, получаем из формулы (5.19)

$$T = 5 \cdot 236t_0 = 1180t_0.$$

Для системы (5.15) с учетом выражения (5.20)

$$T = 2 \cdot 17t_0 + 62t_0 + 2 \cdot 236t_0 = 568t_0.$$

Из формулы (5.21) для $m_1 = 3$; $m_2 = 1$; $m_3 = 0$; $m_4 = 1$

$$T = 3 \cdot 17t_0 + 62t_0 + 236t_0 = 349t_0.$$

2. ПОЛИНОМО-ВЫБОРОЧНЫЕ АЛГОРИТМЫ

Вычисления некоторых функций на всем отрезке изменения аргумента занимают, как правило, большое время и не всегда могут удовлетворять требованиям к быстродействию алгоритмов для УВМ.

В одном из методов уменьшения времени реализации алгоритмов [52, 98] предполагается разбить интервал

[а, в], на котором отыскивается значение функции, на систему непересекающихся подынтервалов. На каждом подынтервале строится свой аппроксимирующий полином с меньшей степенью, чем полином на интервале [а, в], при сохранении той же точности. Для отыскания необходимого подынтервала потребуется $\log_2 k$ операций сравнения (k — число подынтервалов). Однако в логическую схему отыскания подынтервала входит $(k-1)$ операций сравнения.

Объем памяти, отводимый для хранения вычислительных констант

$$D = \sum_{j=1}^{k} (p_j + 1) + (k - 1),$$

где p_j — порядок полинома j-го подынтервала.

Непосредственно вычислению полинома предшествует логическая схема отыскания необходимого подынтервала (соответствующего полинома) [52, 98]. При этом объем логической схемы (число сравнений) прямо пропорционально зависит от числа подынтервалов (k), а время реализации $t \sim \log_2 k$. При этом важно построить логическую схему небольшого объема, которая бы оставалась постоянной для любого числа подынтервалов, а время отыскания необходимого подынтервала было бы также независимо от k. Такие алгоритмы, у которых осуществляется последовательная выборка необходимых коэффициентов полинома соответствующего участка (подынтервала), будем называть полиномо-выборочными алгоритмами.

Рассмотрим сущность метода построения полиномо-выборочных алгоритмов, у которых структура логической схемы не зависит от числа подынтервалов. Как будет показано ниже, построение такой схемы целесообразно для большого числа подынтервалов. Предположим, что аргумент x функции $f(x)$ изменяется в интервале [0; 1]. Разобьем этот интервал на два равных подынтервала:

$$0 \leqslant x < 0{,}5;$$
$$0{,}5 \leqslant x < 1.$$

Для системы счисления с основанием $q = 2$ имеем

$$0 \leqslant x < 0{,}1;$$
$$0{,}1000 \ldots \; 0 \leqslant x < 1.$$

Значение первого двоичного разряда после запятой определяет номер подынтервала: «0» — первый, «1» — второй.

Первый и второй участки функции приблизим полиномами 3-й степени:

$$P_3^1(x) = a_{11}x^3 + a_{12}x^2 + a_{13}x + a_{14};$$
$$P_3^2(x) = a_{21}x^3 + a_{22}x^2 + a_{23}x + a_{24}.$$

Коэффициенты полиномов $P_3^1(x)$ и $P_3^2(x)$ разместим в следующих условных ячейках памяти с условными (относительными) адресами:

$$\langle 0000\rangle = a_{11}; \quad \langle 0001\rangle = a_{21};$$
$$\langle 0010\rangle = a_{12}; \quad \langle 0011\rangle = a_{22};$$
$$\langle 0100\rangle = a_{13}; \quad \langle 0101\rangle = a_{23};$$
$$\langle 0110\rangle = a_{14}; \quad \langle 0111\rangle = a_{24},$$

где скобками $\langle\ \rangle$ обозначено содержимое ячейки.

Функциональная зависимость между кодовым числом аргумента и относительными адресами коэффициентов:

$$\bar{A}a_{j1} := x2^{-[n-1]};$$
$$\bar{A}a_{j2} := \bar{A}a_{j1} + 2A;$$
$$\bar{A}a_{j3} := \bar{A}a_{j2} + 2A;$$
$$\bar{A}a_{j4} := \bar{A}a_{j3} + 2A,$$

где $(:=)$ — знак присвоения (величина $\overline{A}a_{ij}$ $(i = 1, 2, 3, 4)$ — есть целое число, определяемое несколькими разрядами после запятой) значения выражения справа; $\bar{A}$ — относительный адрес; j — номер подынтервала; $2A$ — две адресные единицы (0010).

Если оценить номер подынтервала по двум первым разрядам x после запятой, то можно весь интервал [0; 1] разбить на четыре подынтервала: 00, 01, 10, 11.

Пусть каждый подынтервал аппроксимируется полиномом 3-й степени. Константы полиномов распределим в условных ячейках памяти так, что их относительные адреса:

$$\bar{A}a_{j1} := x2^{-(n-2)};$$
$$\bar{A}a_{j2} := \bar{A}a_{j1} + 4A;$$
$$\bar{A}a_{j3} := \bar{A}a_{j2} + 4A;$$
$$\bar{A}a_{j4} := \bar{A}a_{j3} + 4A,$$

где $4A$ — четыре адресные единицы (0100).

В общем случае если номер подынтервала оценивать по k разрядам, то получим

$$K = 2^k \tag{5.22}$$

равных подынтервала. Каждый участок приблизим полиномом степени p, тогда

$$\bar{A}a_{j_1} := x2^{-(n-k)};$$
$$\bar{A}a_{ij} := \bar{A}a_{j(i-1)} + KA,$$

где $i = 2, 3, \ldots, p + 1$ — порядковый номер коэффициента полинома; KA — число адресных единиц.

Объем долговременной памяти, отводимый под константы,

$$D = \sum_{j=1}^{K} (p_j + 1),$$

или при $p_1 = p_2 = \ldots = p_k = p$

$$D = K(p + 1). \tag{5.23}$$

Для удобства размещения системы констант полиномов в любом месте памяти машины к относительному адресу $\bar{A}a_{j1}$ прибавим адресную константу $(NA)_f$, определяющую начало зоны (в дальнейшем — это номер зоны, равный абсолютному адресу коэффициента a_{11}) размещения системы констант для заданной функции f.

Абсолютные адреса коэффициентов для некоторой функции f:

$$Aa_{j_1} := xq^{-(n-k)} + (NA)_f; \tag{5.24}$$

$$Aa_{ji} := Aa_{j(i-1)} + KA \qquad (i = 2, 3, \ldots, p + 1), \tag{5.25}$$

где q — система счисления.

Зона системы констант разбита на подзоны, число которых равно $p + 1$. В каждой подзоне хранятся коэффициенты с одинаковым порядковым номером, количество которых равно числу подынтервалов K. Номер зоны $N_{\text{зоны}} = (NA)_f$.

Функциональная зависимость (5.24) между кодовым числом аргумента и абсолютным адресом первого коэффициента a_{j1} представляет собой логическую схему, предшествующую вычислению полинома. Как видно, логическая схема содержит операции сдвига и сложения. Объем и время реализации не зависит от числа подынтервалов. Выражение

(5.25) используется непосредственно при вычислении полинома для получения абсолютных адресных коэффициентов $a_{jl} \neq a_{j1}$.

Разбивка интервала изменения аргумента на число подынтервалов $K = q^k$ не всегда является удобным, так как приводит к излишнему объему долговременной памяти, отводимой под константы.

Для одного и того же порядка аппроксимирующего полинома и заданной допустимой погрешности ($\delta_{доп}$) найдем такие K_1 и K_2 и, следовательно, их погрешности δ_1 и δ_2, что при $K_1 = q^{k-1}$ имеем $\delta_1 > \delta_{доп}$, а при $K_2 = q^k$ $\delta_2 < \delta_{доп}$.

Здесь δ_2 удовлетворяет требованию по погрешности и функция разбивается на K_2 подынтервалов.

Для K_1 и K_2 выполняется условие $D_1 < D_2$, где $D_1 = K_1 (p + 1)$, а $D_2 = K_2 (p + 1)$. Но так как $K_2 = qK_1$, то $D_2 = qD_1$.

Если найти такое целое K ($K_1 < K < K_2$), при котором ошибка δ находится в пределах

$$\delta(K_1) > \delta_{доп} \geqslant \delta > \delta(K_2),$$

то объем памяти D (K) под константы уменьшится, при этом

$$D_1 < D < D_2.$$

Если $K - K_1 \ll K_2 - K$, то для $q = 2D/D_2 \approx 0{,}5$, т. е. выигрыш в числе ячеек ДЗУ по сравнению с D_2 близок к 50%.

Преобразуем x так, что при оценке его по первым k разрядам число возможных номеров подынтервалов соответствовало бы K. Для этого разбиваем интервал [0; 1] на K_2 подынтервалов. Находим новый интервал [0; b], на котором размещается K подынтервалов. Тогда

$$b = \frac{K}{K_2} = \frac{K}{q^k},$$

где

$$q^{k-1} < K \leqslant q^k.$$

Преобразуем интервал [0; 1] в интервал [0; b] по формуле

$$x' = bx.$$

Тогда для любого x в интервале [0; 1] будет получено x', не превышающее величины b. Число возможных номеров подынтервалов равно K.

Коэффициенты полиномов распределены в памяти так, что их абсолютные адреса:

$$Aa_{j1} := bxq^{-(n-k)} + (NA)_f;$$
$$Aa_{ji} := Aa_{j(i-1)} + KA$$
$$(i = 2, 3, \ldots, p+1).$$

Отсюда видно, что для $K \neq q^k$ в логическую схему дополнительно необходимо ввести операцию умножения. Параметры логической схемы остаются независимыми от K.

Рассмотрим интервал $r \leqslant |x| < c$. Для получения адреса 1-го коэффициента преобразуем $|x|$ в $[r - c]$ и x'' в [0 — 1]. При этом

$$x'' = \frac{|x| - r}{c - r},$$

тогда

$$Aa_{j1} := b\frac{|x| - r}{c - r} q^{-(n-k)} + (NA)_f, \qquad (5.26)$$

где

$$b = \frac{K}{q^k}, \quad q^{k-1} < K \leqslant q^k.$$

Преобразуем выражение (5.26)

$$Aa_{j1} := \frac{b}{c - r}|x| q^{-(n-k)} + (NA)_f - \frac{b}{c - r} rq^{-(n-k)}.$$

Подставив конкретные значения b, c, r, n, k, $(NA)_f$, получим

$$Aa_{j1} := B|x| q^{-(n-l)} + (NA)_f^{\partial}, \qquad (5.27)$$

где

$$Bq^{-(n-l)} = \frac{b}{c - r} q^{-(n-k)};$$

$$(NA)_f^{\partial} = (NA)_f - \frac{b}{c - r} rq^{-(n-k)}.$$

Зона размещения системы констант имеет номер $(NA)_f$. Константы полиномов с большим порядковым номером раз-

мещены так, что их абсолютные адреса

$$Aa_{jl} := Aa_{j(l-1)} + KA \qquad (i = 2, 3, \ldots, p+1).$$

В выражении (5.27) операцию формирования модуля можно исключить, если $x > 0$. Логическая схема, реализующая выражение (5.27), применима для аппроксимирования функции или в положительной или в отрицательной области изменения аргумента.

Рассмотрим функцию, которую следует аппроксимировать как в положительной, так и в отрицательной областях с одинаковыми границами интервалов. В качестве признака, указывающего на положительный или отрицательный интервал аппроксимации, используется знаковый разряд. Исключив в выражении (5.27) операцию формирования модуля, получим

$$Aa_{j1} := (Bx)\, q^{-(n-l)} + (NA)_f^{\partial}, \tag{5.28}$$

где

$$Bq^{-(n-l)} = \frac{b}{c-r} q^{-(n-k)};$$

$$(NA)_f^{\partial} = (NA)_f - \frac{b}{c-r} rq^{-(n-k)}.$$

В данной логической схеме знаковый разряд произведения Bx сдвигается вместе с цифровыми разрядами.

Определим номер зон систем коэффициентов для положительной и отрицательной областей изменения аргумента.

Обозначив $Bx = X$, запишем

$$X = S_x + |X|,$$

где $S_x = 0$ при $X > 0$ и $S_x = 1$ при $X < 0$.
Тогда

$$Aa_{j1} := (S_x + |X|)\, q^{-(n-l)} + (NA)_f^{\partial} =$$
$$= |X|\, q^{-(n-l)} + (NA)_f^{\partial} + S_x q^{-(n-l)}, \tag{5.29}$$

или

$$Aa_{j1} := B|x|\, q^{-(n-l)} + (NA)_f^{\partial} + S_x q^{-(n-l)}.$$

При $x > 0$ имеем $S_x = 0$ и выражение (5.29) соответствует формуле (5.27). Номер зоны $N_{\text{зоны}}\ (x > 0) = (NA)_f$. При $x < 0$ имеем $S_x = 1$, и

$$N_{\text{зоны}}\ (x < 0) = (NA)_f + q^{-(n-l)}.$$

Это положительное смещение возникло в результате сдвига знака числа $S_x = 1$.

Так как число подынтервалов равно $2K$, то

$$Aa_{jl} := Aa_{j(l-1)} + 2KA.$$

Логическая схема, реализующая выражение (5.28), применяется для аппроксимации функции в положительной или отрицательной областях. При этом номера зон:

$$N_{\text{зоны}}(x > 0) = (NA)_f;$$

$$N_{\text{зоны}}(x < 0) = (NA)_f + q^{-(n-l)}.$$

Если число подынтервалов равно K, то

$$Aa_{jl} := Aa_{j(l-1)} + KA.$$

Если для изображения чисел в машине применяется обратный или дополнительный код, то логическая схема дополняется операцией выделения тех разрядов, по которым определяется номер подынтервала.

Пусть $[x]$ — аргумент, представленный в одном из кодов. Тогда в выражении (5.28), выделив $l + 1$ разряд, получим

$$Aa_{jl} := \{B[x]q^{-(n-l)}\} \wedge \{q^{-[n-(l+1)]} - q^{-n}\} + (NA)_f^{\partial}, \quad (5.30)$$

где

$$Bq^{-(n-l)} = \frac{b}{c-r} q^{-(n-k)};$$

$$(NA)_f^{\partial} = (NA)_f - \frac{b}{c-r} rq^{-(n-k)};$$

$\{q^{-[n(l+1)]} - q^{-n}\}$ — константа выделения $l + 1$ разряда.

Определим размещение системы констант в памяти машины. Обозначив $B[x] = [X]$, запишем, что $[X] = S_X + [X]_{\text{ц}}$, где $[X]_{\text{ц}}$ — цифровые разряды $[X]$.

Отсюда

$$Aa_{jl} := \{(S_X + [X]_{\text{ц}})q^{-(n-l)}\} \wedge \{q^{-[n-(l+1)]} - q^{-n}\} + (NA)_f^{\partial} =$$

$$= \{[X]_{\text{ц}} q^{-(n-l)}\} \wedge \{q^{-[n-(l+1)]} - q^{-n}\} +$$

$$+ S_X q^{-(n-l)} \{q^{-[n-(l+1)]} - q^{-n}\} + (NA)_f^{\partial} = \{[X]_{\text{ц}} q^{-(n-l)}\} \wedge$$

$$\wedge \{q^{-[n-(l+1)]} - q^{-n}\} + S_X q^{-(n-l)} + (NA)_f^{\partial}. \quad (5.31)$$

Для $x > 0$ имеем $S_X = 0$, $[X]_{\text{д}} = X = B \mid x \mid$. После операции выделения получим

$$Aa_{j1} := B \mid x \mid q^{-(n-l)} + (NA)_f^{\partial},$$

что соответствует выражению (5.27).

Номер зоны

$$N_{\text{зоны}}(x > 0) = (NA)_f.$$

Для $x < 0$ имеем $S_X = 1$. Номер подынтервала, оцененный по l разрядам, будет представлен в обратном коде. Пронумеруем подынтервалы в отрицательной области в обратном направлении. Из формулы (5.21) видно, что номера зон ($x < 0$) смещены на величину $2^{-(n-l)}$ в сторону увеличения адресов за счет сдвига знака числа, т. е.

$$N_{\text{зоны}}(x < 0) = (NA)_f + q^{-(n-l)}.$$

Если функция аппроксимируется и в положительной, и в отрицательной областях, то число подынтервалов равно $2K$ и абсолютный адрес

$$Aa_{jl} := Aa_{j(l-1)} + 2KA.$$

Если функция аппроксимируется только в положительной области, то следует применить логическую схему, реализующую выражение (5.28). Для отрицательной области аппроксимирования функции выбирается логическая схема (5.30). При этом

$$Aa_{jl} := Aa_{j(l-1)} + KA.$$

Ранее рассмотренные выражения для получения абсолютных адресов констант полиномов требуют расположения коэффициентов с одинаковым порядковым номером в каждой подзоне. В некоторых случаях желательно размещение коэффициентов $a_{j1}, a_{j2}, \ldots, a_{j(p+1)}$ данного j-го полинома в памяти машины с шагом $1A$. В этом случае зона коэффициентов данной функции разбивается на подзоны коэффициентов полинома данного подынтервала.

Пусть так же, как и ранее, по k разрядам аргумента определим номер подынтервала. Отведем для каждого подынтервала свою подзону коэффициентов. Ячейки каждой подзоны условно пронумеруем, начиная с нуля. В нулевых ячейках разместим коэффициенты a_{j1} в следующих старших номерах — $a_{j2}, a_{j3}, \ldots$

Так как номер подзоны соответствует номеру подынтервала и определяется по k разрядам, то для изображения

номера ячейки в подзоне следует отвести ξ разрядов, следующих после k-го разряда. При этом число разрядов ξ выбирается из условия

$$(p+1) \leqslant q^{\xi}.$$

Сдвигая аргумент x вправо на $[n-(k+\xi)]$ разрядов и «обнуляя» разряды ξ, получаем относительный адрес коэффициента a_{j1}. Для прямого кода представления чисел, если функция аппроксимируется и в положительной, и в отрицательной областях в интервале $r \leqslant |x| < c$, запишем абсолютный адрес

$$Aa_{j1} := \{Bxq^{-[n-(l+\xi)]}\} \wedge \{q^{-[n-(l+\xi+1)]} - q^{-[n-\xi]}\} + (NA)_f^{\partial}, \qquad (5.32)$$

где

$$Bq^{-[n-(l+\xi)]} = \frac{b}{c-r} q^{-[n-(k+\xi)]};$$

$$(NA)_f^{\partial} = (NA)_f - \frac{b}{c-r} rq^{-[n-(k+\xi)]};$$

$\{q^{-[n-(l+\xi+1)]} - q^{-[n-\xi]}$ — константа выделения $l+1$ разряда (выделяется также знаковый разряд).

Номер зоны

$$N_{\text{зоны}}(x>0) = (NA)_f.$$

Для отрицательной области выделенный знаковый разряд дает положительное смещение номера зоны. Тогда

$$N_{\text{зоны}}(x<0) = NA_f + q^{-[n-(l+\xi)]}.$$

Адреса коэффициентов с порядковым номером $(i \neq 1)$ находим из последовательности

$$Aa_{ji} := Aa_{j(i-1)} + 1A \qquad (i = 2, 3, \ldots, p+1). \quad (5.33)$$

Логическая схема, реализующая выражение (5.32), может применяться при аппроксимировании функции в положительной или в отрицательной области.

Выражение (5.32) и размещение констант полиномов в памяти машины справедливы также для машин, в которых числа представляются в обратных или дополнительных кодах. При этом подынтервалы в отрицательной области нумеруются в обратном направлении.

В логических схемах, реализующих выражения (5.28), (5.29), (5.32), длинную операцию умножения иногда

исключают, если величина B представляется как q^{φ}. Тогда число сдвигов уменьшается на φ разрядов.

Если функция аппроксимируется в неравномерных подынтервалах, то в логическую схему вводят некоторую функцию, преобразующую неравномерные подынтервалы в равномерные. При этом для прямого кода представления чисел абсолютный адрес

$$Aa_{j1} := bF\left(\frac{x-r}{c-r}\right) q^{-(n-l)} + (NA)_f,$$

для обратного и дополнительного кодов абсолютный адрес

$$Aa_{j1} := \left\{bF\left(\frac{x-r}{c-r}\right) q^{-(n-l)}\right\} \wedge \{q^{-[n-(k+1)]} - q^{-n}\} + (NA)_f.$$

При реализации повышенной точности вычислений полиномо-выборочными алгоритмами показатели эффективности логической схемы (объем и быстродействие) остаются прежними, так как $K < q^n$. Однако объем долговременной памяти, отводимый под константы, увеличивается в Q раз для Q-значной точности

$$D_Q = QK(p+1).$$

Распределим однозначные слова по отдельным зонам в памяти машины, а i-е коэффициенты разместим согласно логической схемы, реализующей выражение (5.32), и последовательности (5.33). Тогда номера зон:
для $x \geqslant 0$

$$N_{\text{зоны}}(\varphi) = (NA)_f + (\varphi - 1)\,[(Kq^{\xi})\,A],$$

для $x < 0$

$$N_{\text{зоны}}(\varphi) = (NA)_f + q^{-[n-(l+\xi)]} + (\varphi - 1)\,[(Kq^{\xi})\,A]$$
$$(\varphi = 1,\ 2,\ \ldots,\ Q).$$

Абсолютные адреса коэффициентов находим логической схемой, реализующей выражение

$$Aa_{ji1} := \{Bxq^{-[n-(l+\xi)]}\} \wedge \{q^{-[n-(l+\xi)]} - q^{-[n-\xi]}\} + (NA)_f^{\partial}.$$

Адрес старших слов i-х коэффициентов

$$Aa_{ji1} := Aa_{j(l-1)1} + 1A.$$

Адреса младших слов i-х коэффициентов

$$Aa_{jl\varphi} := Aa_{jl(\varphi-1)} + (Kq^{\xi})\,A.$$

3. ВЫБОР ОПТИМАЛЬНОГО ЧИСЛА ПОДЫНТЕРВАЛОВ И СТЕПЕНИ АППРОКСИМИРУЮЩЕГО ПОЛИНОМА

Для выбора оптимального числа подынтервалов следует установить зависимость между числом подынтервалов K, степенью аппроксимирующего полинома p и абсолютной ошибкой аппроксимации δ. Для этого данную функцию на каждом подынтервале аппроксимируем интерполяционным полиномом Лагранжа. Узлы интерполирования выбираем по методу Чебышева. Ошибка такой аппроксимации [62] не превышает значения

$$\delta_{\text{доп}} = \frac{(b-a)^{p+1}}{(p+1)!\,2^{2p+1}} M_{f_{\max}}^{p+1}[a, b], \tag{5.34}$$

где b, a — соответственно верхняя и нижняя границы подынтервала; p — степень полинома; $M_{f_{\max}}^{p+1}[a, b]$ — максимальная по модулю $(p+1)$-я производная функции на интервале аппроксимации $[a, b]$.

Если интервал $[A, B]$, где A — верхняя граница интервала, а B — нижняя граница интервала, разбить на K подынтервалов, то

$$b - a = \frac{B-A}{K}. \tag{5.35}$$

Подставив выражение (5.35) в формулу (5.34), получим

$$\delta_{\text{доп}} = \frac{(B-A)^{p+1}}{(p+1)!\,2^{2p+1}K^{p+1}} M_{f_{\max}}^{p+1}[A, B], \tag{5.36}$$

откуда

$$K = \frac{B-A}{4} \sqrt[p+1]{\frac{2M_{f_{\max}}^{p+1}[A, B]}{\delta_{\text{доп}}(p+1)!}}.$$

Так как число подынтервалов не может быть дробным, то, округляя до большего целого, находим

$$K^0 = \left\{\frac{B-A}{4} \sqrt[p+1]{\frac{2M_{f_{\max}}^{p+1}[A, B]}{\delta_{\text{доп}}(p+1)!}}\right\}^{\uparrow}.$$

Значения $M_{f_{\max}}^{p+1}[A, B]$ для некоторых функций приведены в табл. 1.

Для заданного интервала $[A, B]$ и допустимой ошибки аппроксимации $\delta_{\text{доп}}$ при различных p (см. табл. 1) можно

Таблица 1

Степень полинома	Интервал [0; 1]			Интервал [0,5; 1]		
	sin	cos	exp	ln	$\sqrt{X}$	$1/X$
0	1	0,84175	2,7183	2	0,707	4
1	0,84175	1	2,7183	4	0,707	16
2	1	0,84175	2,7183	16	2,12	96
3	0,84175	1	2,7183	96	10,6	7680
4	1	0,84175	2,7183	768	74,24	—
5	0,84175	1	2,7183	7680	—	—
6	1	0,84175	2,7183	—	—	—

найти такое K^0, при котором $D = K^0 (p + 1)$ удовлетворяет условию $D \leqslant D_{\text{доп}}$.

Представим графическое решение выражения (5.36). Для этого примем, что $M_{f_{\max}}^{p+1} [A, B] = 1$, тогда получим нормированную ошибку

$$\delta_{\text{доп}}^{\text{н}} = \frac{(B - A)^{p+1}}{(p + 1)\,!\, 2^{2p+1} K^{p+1}}.$$

Определим $\delta_{\text{доп}}^{\text{н}}$ для наиболее часто встречающихся интервалов [0; 1] и [0,5; 1].

Если $K = 2^k$, то

$$\delta_{\text{доп}}^{\text{н}} [0;\ 1] = \frac{1}{(p + 1)\,!\, 2^{2p+1+k(p+1)}}. \qquad (5.37)$$

Выражение для расчета $\delta_{\text{доп}}^{\text{н}}$ приведем к виду

$$\delta_{\text{доп}}^{\text{н}} = c \cdot 2^{-r},$$

где $c < 1$ и r для различных p и K выбирают из табл. 2.

Коэффициент c от K (числа подынтервалов) не зависит.

Определим значение нормированной ошибки для интервала [0,5; 1]

$$\delta_{\text{доп}}^{\text{н}} [0{,}5;\ 1] = \frac{(0{,}5)^{p+1}}{(p + 1)\,!\, 2^{2p+1+k(p+1)}} = \delta_{\text{доп}}^{\text{н}} [0;\ 1]\, 2^{-(p+1)}.$$

Таблица 2

Число подынтервалов	Порядок полинома							
	0	1	2	3	4	5	6	7
	Коэффициент (c)							
	1	1	0,66	0,66	0,53	0,71	0,82	0,82
1	1	4	7	11	15	20	25	30
2	2	6	10	15	20	26	32	38
4	3	8	13	19	25	32	39	46
8	4	10	16	23	30	38	46	54
16	5	12	19	27	35	44	53	62
32	6	14	22	31	40	50	60	70
64	7	16	25	35	45	56	67	78
128	8	18	28	39	50	62	74	86
256	9	20	31	43	55	68	81	94
512	10	22	34	47	60	74	88	102

Графики нормированной ошибки для различных интервалов показаны на рис. 11.

Истинное значение ошибки

$$\delta = \delta^{\text{н}}_{\text{доп}} M^{p+1}_{f_{\max}} [A, B],$$

значения $M^{p+1}_{f_{\max}} [A, B]$ выбирают из табл. 1.

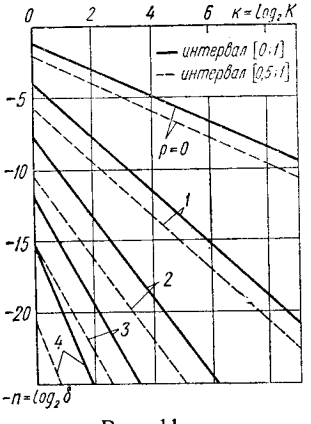


Рис. 11.

Сравнительные оценки методов. Сравним некоторые показатели эффективности логической схемы [52, 98] и логических схем, реализующих выражения (5.28), (5.30), (5.32). Для сравнения выберем следующие показатели: t — время реализации логической схемы; d — объем долговременной памяти, отводимый под программу логической схемы (количество команд). Определим эти показатели как функцию от числа подынтервалов K. Объем долговременной памяти D, отводимый под константы полиномов, исключим, так как для различных методов при одинаковом K величину D в первом приближении можно считать равнозначной.

Известно, что $t \sim \log_2 K$ [52, 98], при этом для определения номера подынтервала необходимо сравнить аргумент функции с границами подынтервалов. Для одного сравнения

требуются следующие команды: посылка x на Σ, вычитание из x границы подынтервала, проверка знака результата. Для вычислительных машин такие операции являются короткими. Примем, что время выполнения одной короткой операции равно $1t_0$, тогда время реализации логической схемы

$$t_1 = 3t_0 \log_2 K.$$

Так как логическая схема содержит $K - 1$ блоков сравнения, в каждом из которых содержится три команды, то объем долговременной памяти

$$d_1 = 3(K - 1).$$

Для реализации выражений (5.28) и (5.30) ($x > 0$) требуется пять команд ($d_2 = 5$): посылка x на Σ, умножение

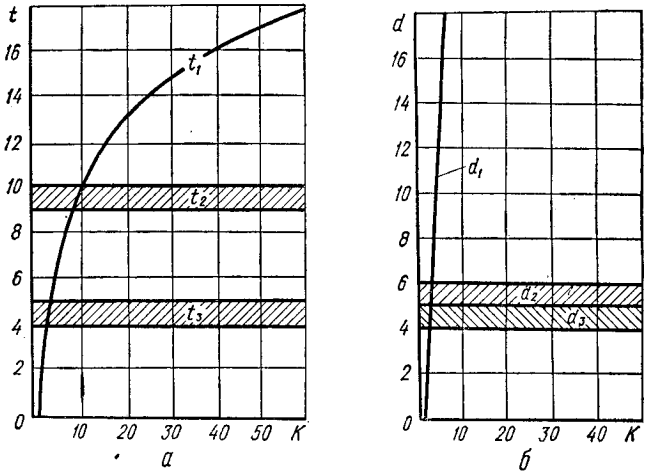


Рис. 12.

$\langle \Sigma \rangle$ на B, сдвиг вправо на $n - l$ разрядов, сложение с $(NA)_i^{\partial}$; запись $\langle \Sigma \rangle$ в ячейку.

Для реализации выражений (5.30) и (5.32) дополнительно вводится команда выделения, тогда $d_2 = 6$. Для большинства вычислительных машин операция умножения — длинная операция. Примем, что $t_{\text{умн}} = 5t_0$. Остальные операции программы — короткие ($1t_0$).

Время реализации выражений (5.28), (5.30), (5.32)

$$t_2 = 9t_0\,(10t_0),\ \text{а}\ d_2 = 5\,(6).$$

Если $B = 2^{\varphi}$, то операцию умножения можно исключить, тогда

$$d_3 = 4\,(5);$$
$$t_3 = 4t_0\,5\,(t_0).$$

Графики зависимостей $t = f(K)$ и $d = f(K)$ показаны на рис. 12, *а* и *б* (области изменений t и d заштрихованы). Из графиков видно, что для $K \geqslant 4$ при $B = 2^{\varphi}$ и $K > 10$ при $B \neq 2^{\varphi}$ целесообразнее применять метод полиномовыборочных алгоритмов, у которого логическая схема, предшествующая вычислению полинома, не зависит от числа подынтервалов.

В классе итерационных процессов методом полиномовыборочных алгоритмов можно отыскать наилучшие константы начального приближения ($p = 0$) в зависимости от того, в каком подынтервале отыскивается значение функции. Ошибка начального приближения может быть уменьшена, если принять за начальное приближение линейное ($p = 1$) или квадратичное ($p = 2$). Применение полиномовыборочных алгоритмов в классе итерационных процессов позволяет уменьшить число итераций за счет лучшего выбора начального приближения.

Глава VI

ПРИЛОЖЕНИЕ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ ДЛЯ ОПИСАНИЯ ПРОЦЕССОВ ДИСПЕТЧЕРИЗАЦИИ ПРИ РЕАЛИЗАЦИИ АЛГОРИТМОВ НА УВС

1. ИНФОРМАЦИОННЫЕ ПОТОКИ АЛГОРИТМОВ КАК ВХОДЯЩИЕ ПОТОКИ ТРЕБОВАНИЙ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Первичной задачей, которая неизбежно возникает при теоретической разработке теории массового обслуживания и применении ее методов, является изучение потока требований, который поступает на обслуживающий прибор.

В подавляющем большинстве работ по теории массового обслуживания [46, 48, 140] рассматривается простейший

поток требований, когда вероятность поступления в промежуток времени t k требований задается формулой

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t},$$

где $\lambda > 0$ — плотность потока требований (параметр потока).

Поступающий поток требований считается таким, что для любой конечной группы непересекающихся отрезков времени числа появившихся на их протяжении требований представляют собой взаимно независимые величины.

Рассмотрение именно простейшего потока требований объясняется следующими обстоятельствами.

1. Для других видов потоков требований не получены пока простые формульные зависимости количественной оценки качества функционирования систем массового обслуживания.

2. Простейший поток требований обслуживать труднее. Поэтому работу систем ставят в более тяжелые условия. Если средства обслуживания рассчитывать на тяжелый случай, то обслуживание системой других случайных потоков требований с одинаковой плотностью поступления требований будет надежнее [49].

3. Простейший поток требований в теории массового обслуживания играет такую же роль, как нормальный закон распределения случайных величин в теории вероятностей. При сложении нескольких случайных потоков требований образуется суммарный поток, который по своим характеристикам приближается к простейшему.

Пуассоновские потоки требований (или весьма близкие к ним по структуре) часто имеют место в действительности, так как в определенном смысле они являются предельными для различных потоков. Например, если накладывать друг на друга (складывать) большое число различных по структуре потоков требований, то суммарный поток в широком классе условий близок к пуассоновскому. С другой стороны, если взять произвольный поток требований и из него случайным образом выбрасывать требования, то после нескольких таких «разрежений» полученный поток требований будет также близок к пуассоновскому. На практике очень часто имеет место сложение или случайное «разрежение» потоков требований, поэтому пуассоновские потоки требований находят широкое применение.

Предельная теорема для суммарного потока требований утверждает сходимость суммы независимых, ординарных, стационарных потоков к простейшему. При этом складываемые потоки требований должны оказывать одинаково малое влияние на суммарный поток. Другими словами, среди суммируемых потоков требований не должно быть потоков с очень большой интенсивностью (по сравнению с суммарной интенсивностью всех остальных); интенсивности складываемых потоков не должны становиться по мере увеличения номера потока исчезающе малыми. Важно отметить, что сходимость суммарного потока требований к простейшему осуществляется очень быстро. Практически можно считать, что сложение четырех-пяти стационарных, ординарных, независимых потоков требований, сравнимых по интенсивности, достаточно для того, чтобы суммарный поток требований был близок к простейшему.

На практике часто потоки требований возникают в результате сложения не строго независимых, а слабозависимых потоков событий. Исследования, проведенные методом статистических испытаний, показывают, что и в этом случае (при достаточном числе слагаемых) суммарный поток требований также оказывается близок к простейшему.

Если складываемые потоки нестационарны, то получаемый суммарный поток требований близок к нестационарному пуассоновскому.

Из всего вышеизложенного следует, что многие потоки требований, возникающие на практике и фигурирующие в задачах массового обслуживания, можно приближенно считать пуассоновскими. Заметим, что пуассоновский поток требований обладает устойчивостью, т. е. при суммировании независимых пуассоновских потоков получается снова пуассоновский, причем интенсивности складываемых потоков суммируются.

Если не предполагать, что потоки требований пуассоновские, то для аналитического исследования требуется сложный математический аппарат. К тому же в большинстве задач прикладного характера замена непуассоновских потоков требований пуассоновскими с теми же интенсивностями приводит к решению, которое мало отличается от истинного, а иногда и вовсе не отличается. При этом погрешность решения, как правило, находится в пределах точности исходных данных, которые зачастую известны приближенно. Специальное моделирование различных задач,

проведенное методом Монте-Карло [34], показало, что в большинстве случаев эта погрешность равна 3—5% и лишь редко доходит до 10—12%, что вполне приемлемо при решении ряда прикладных задач. Данное положение объясняется тем, что потоки требований, протекающие в реальных системах, по своей структуре близки к пуассоновским.

Однако имеются особые условия, при которых погрешность достигает значительных величин. Поэтому при решении сложных задач, когда нет уверенности в том, что замена реальных потоков пуассоновскими приведет к малым ошибкам, необходимо проверять аналитическое решение методом Монте-Карло (методом статистических испытаний). Решение, полученное с помощью пуассоновских систем, рассматривается как первое приближение. Это сокращает расчеты методом Монте-Карло.

В качестве простого критерия отличия реального стационарного потока от пуассоновского можно рассматривать близость математического ожидания и дисперсии числа событий, наступающих на определенном участке времени в реальном потоке.

Учитывая, что в автоматических системах управления потоки от разных датчиков накапливаются в некотором буферном запоминающем устройстве, поэтому суммарный поток близок по своей структуре к простейшему.

Если не выполняется условие стационарности для потоков входной информации и они имеют импульсный характер, то необходимо принимать меры для сглаживания таких потоков, что достигается введением сглаживающих буферных накопителей. В этом случае потоки после сглаживания становятся более равномерными и близкими к стационарным.

Так как эффективность систем автоматизированного управления и обработки информации зависит не только от эффективности самих обрабатываемых алгоритмов, но и от системы диспетчеризации решения основных функциональных алгоритмов, т. е. от способа обработки их, то необходимо иметь некоторый формальный метод, позволяющий оценивать эффективность принятой системы диспетчеризации.

Иногда удается организовать синхронный детерминированный способ приема исходных данных, их обработки в управляющей ЦВМ и выдачи команд на управляемые объекты. Если интервалы времени между поступающими и выдаваемыми сообщениями, а также длительность обработ-

ки каждого сообщения постоянны и равны, а фазы согласованы, то полностью используется быстродействие ЦВМ без снижения эффективности за счет задержек и потерь информации.

Однако, как правило, моменты поступления исходных данных носят случайный характер и длительности обработки каждого сообщения не равны между собой. Это приводит к неизбежным потерям и задержкам информации в управляющей системе. Тогда такие ЦВМ могут рассматриваться как системы массового обслуживания с одним или несколькими входными потоками заявок (алгоритмов).

Отметим различие между потоками различных источников информации и входными потоками заявок для ЦВМ как системы массового обслуживания. При анализе таких систем следует учитывать прежде всего характеристики потоков по типам поступающих заявок и времени их обработки в ЦВМ. Так, если один источник информации передает сообщения нескольких типов, различающихся по времени обработки и величине штрафа вследствие их задержки или потери, то такой источник характеризуется несколькими потоками. И напротив, может встретиться случай, когда от различных источников поступает однотипная информация. Естественно, что абоненты такого типа представляются одним суммарным потоком. Таким образом, входные потоки для управляющих ЦВМ классифицируют по типу поступающей информации, времени ее обработки и возможному штрафу за потерю или задержку сообщений.

Входные потоки информации в управляющую ЦВМ формируются не только внешними источниками, а также в процессе решения некоторых задач. В ЦВМ это приводит к образованию, в общем случае, случайного потока заявок на включение определенных подпрограмм, источником которого является сама ЦВМ. Кроме того, заявки на решение некоторых задач формируются в ЦВМ периодически по счетчику реального времени ЦВМ. Такие заявки образуют регулярный поток заявок.

Отдельный поток заявок составляют сигналы аппаратного контроля ЦВМ и внешних специализированных устройств. Появление этих сигналов вызывает обычно прерывание выполняемых вычислений и включение специальных подпрограмм анализа и исправления сбоев, тестового контроля аппаратуры и контроля информации, хранящейся

в оперативной памяти ЦВМ. Таким образом, в ЦВМ поступает в основном четыре типа входных потоков заявок:

от внешних источников информации;

сформированные при решении основных функциональных задач;

на включение периодических подпрограмм;

на включение подпрограмм, выполняющих анализ аппаратного контроля ЦВМ и внешних специализированных устройств.

Процесс управления основывается на целенаправленной переработке различной информации. В настоящее время все большее значение приобретают количественные методы, используемые при обработке этой информации, и, в частности, оценки качества управления, которые требуют разработки специальных критериев эффективности. Эти критерии должны обеспечить выбор оптимальных способов управления из всех возможных. Теория массового обслуживания оказывает большую помощь при построении таких критериев эффективности.

Для задач передачи и переработки информации требованием на обслуживание можно условно считать поступление новой информации, а обслуживание — ее передачу или переработку. При этом возникает проблема создания такой системы управления, которая обеспечивает надежную переработку всего объема поступающей информации в допустимые сроки и не является при этом избыточной. Эта задача относится к классу задач теории массового обслуживания и может быть успешно решена при использовании методов и аппарата теории массового обслуживания.

2. ОСНОВНЫЕ ТИПЫ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Если система массового обслуживания состоит из одного обслуживающего аппарата, то ее называют одноканальной, если же из нескольких обслуживающих аппаратов, то — многоканальной.

Существует много разновидностей систем массового обслуживания, каждая из которых характеризуется определенной организацией работы. Различают многоканальные системы массового обслуживания упорядоченные и неупорядоченные. В неупорядоченных системах поступившее требование может занять любой из свободных обслуживающих аппаратов, в упорядоченных — строго определенный.

Как многоканальные (упорядоченные и неупорядоченные), так и одноканальные системы массового обслуживания могут быть отказами (с потерями) или с ожиданием.

Системы с отказами характеризуются тем, что требование не может ждать начала обслуживания, или, что то же самое, система обслуживания отказывает требованию, поступившему в тот момент, когда все обслуживающие аппараты заняты. Если требование, поступившее в систему, получило отказ, то, следовательно, оно покидает систему необслуженным. Поэтому подобные обслуживающие системы называются системами с потерями.

Если требование, поступившее в систему массового обслуживания, может ее покинуть только тогда, когда оно полностью обслужено, а требований поступило больше, чем число обслуживающих аппаратов, то требования, поступившие в систему в тот момент, когда все обслуживающие аппараты заняты, образуют очередь. Поэтому такие системы массового обслуживания называются системами с ожиданием.

Системы массового обслуживания с ожиданием и с потерями не исчерпывают всех типов задач теории массового обслуживания. Большое количество задач лежит между задачами этих двух типов, так как организация систем массового обслуживания может быть различной.

Бывают системы массового обслуживания смешанного типа, в которых накладываются дополнительные ограничения:

1) на время ожидания требования в очереди, которое может быть как случайной, так и постоянной величиной, при этом ограничивается только срок ожидания заявки в очереди, а начатое обслуживание доводится до конца;

2) на общее время пребывания заявки в системе;

3) на число заявок в очереди, т. е. на длину очереди.

Кроме того, системы массового обслуживания различаются по числу требований, которые могут одновременно находиться в обслуживающей системе. В некоторых задачах число требований, одновременно находящихся в обслуживающей системе, не может быть больше определенного числа. Однако в ряде задач число требований, находящихся одновременно в обслуживающей системе, может быть настолько большим, что поток требований можно рассматривать как неограниченный. Такие системы принято называть замкнутыми и разомкнутыми соответственно.

При решении задач массового обслуживания большое значение имеет правильный выбор критериев, характеризующих изучаемый процесс. Одна и та же система массового обслуживания характеризуется различными критериями эффективности. Выбор критерия эффективности — очень важный этап исследования, поскольку от того, насколько правильно выбран критерий, зависит оценка самой системы обслуживания. Перечислить все критерии, которые могут или могли бы быть полезными во всех задачах массового обслуживания, невозможно, поэтому ограничимся указанием наиболее существенных и часто используемых.

Под эффективностью обслуживающей системы понимают характеристику уровня выполнения этой системой тех функций, для которых она предназначена.

Выбор критерия эффективности зависит от трех факторов: характеристик качества и надежности системы массового обслуживания; экономических показателей, характеризующих работу системы (стоимость, трудовые затраты обслуживающего персонала, убытки, связанные с несвоевременным обслуживанием); особенностей ситуации, в которой эксплуатируется система (параметры потоков требований, ограничения на длину очереди и др.). В зависимости от условий эксплуатации системы и критерия эффективности выбирается и математическая модель.

Выбор критерия эффективности зависит от исследуемой задачи. В системах массового обслуживания с отказами критериями эффективности обслуживания являются: вероятность отказа очередному требованию в обслуживании; вероятность того, что все обслуживающие аппараты системы не заняты; вероятность того, что заняты S обслуживающих аппаратов; среднее число занятых обслуживающих аппаратов; относительная пропускная способность системы.

Вероятность отказа очередному требованию определяет, в какой степени данная система массового обслуживания способна удовлетворить поступающий поток заявок. Этот критерий не связан с качеством обслуживания внутри системы. Он дает только внешнюю оценку способности системы приступить к обслуживанию поступившего требования.

Среднее число занятых обслуживающих аппаратов характеризует степень загрузки системы массового обслуживания. Более полно загрузка системы характеризуется законом распределения количества занятых аппаратов.

В системах массового обслуживания с ожиданием при условии, что на ожидание никаких ограничений не наложено, эффективность обслуживания характеризуется следующими показателями: вероятностью того, что время ожидания начала обслуживания меньше некоторой фиксированной величины τ; средним временем ожидания начала обслуживания; средним числом требований, ожидающих начала обслуживания, вероятностью того, что все обслуживающие аппараты свободны или заняты; средним числом свободных обслуживающих аппаратов и др. При наличии ограничений на ожидание характеристики эффективности обслуживания изменяются в зависимости от типа ограничений. Если в систему массового обслуживания не может поступить больше k требований, то вызывают интерес прежде всего такие характеристики, как среднее число требований, находящихся в системе массового обслуживания; вероятность того, что число требований, ожидающих начала обслуживания, не больше некоторого числа r; коэффициент простоя обслуживающих аппаратов (отношение среднего числа свободных обслуживающих аппаратов к общему числу обслуживающих аппаратов).

При неограниченном числе обслуживающих аппаратов характеристиками эффективности обслуживания являются: вероятность того, что в установившемся процессе обслуживания занято S обслуживающих аппаратов, независимо от начального состояния системы; вероятность того, что все обслуживающие аппараты свободны; вероятность того, что в определенный момент времени заняты S обслуживающих аппаратов, при условии, что в начальный момент они все свободны; среднее число обслуживающих аппаратов, занятых в определенный момент времени, при условии, что в начальный момент они все свободны.

Наиболее характерными дисциплинами обслуживания заявок в управляющих ЦВМ являются:

бесприоритетные дисциплины обслуживания, при которых выбор заявок для обслуживания из числа ожидающих в очереди осуществляется в некотором заранее установленном порядке без учета типа этих заявок, их относительной важности и времени обслуживания;

приоритетные дисциплины обслуживания, при реализации которых некоторым заявкам предоставляется преимущество в обслуживании перед заявками других типов.

По способу реализации в ЦВМ приоритетные дисциплины в свою очередь подразделяются на:

дисциплины с относительными приоритетами, в которых не допускается прерывание начатого обслуживания;

дисциплины с абсолютными приоритетами, в которых допускается прерывание обслуживания заявок более низкого приоритета при поступлении в ЦВМ заявок с более высокими приоритетами;

— смешанные дисциплины обслуживания, в которых используются как относительные, так и абсолютные приоритеты.

Перечисленные дисциплины могут отличаться друг от друга некоторыми деталями. Так, например, может накладываться ограничение на продолжительность обслуживания заявок каждого типа, на количество последовательно обслуживаемых однотипных заявок и т. п. Классификация дисциплин обслуживания заявок с ожиданием в управляющих ЦВМ показана на рис. 13 [27].

3. СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ С ПОТЕРЯМИ

Особенностью функционирования этих систем является то, что всякое требование, поступившее в систему в некоторый момент времени, либо начинает сразу выполняться, либо теряется, если все обслуживающие приборы заняты, и весь дальнейший процесс обслуживания идет так, как если бы это требование вообще не поступило.

Для систем массового обслуживания с потерями основным показателем является вероятность потери требования и среднее число аппаратов, занятых обслуживанием. Первый критерий характеризует соотношение потока обслуженных требований и входящего потока, т. е. полноту обслуживания входящего потока, второй критерий — степень загрузки системы массового обслуживания.

Простейшие задачи для систем массового обслуживания с потерями были впервые решены А. К. Эрлангом. Им же были выведены формулы оценки функционирования этих систем при условии поступления простейшего потока заявок и для показательного закона распределения времени обслуживания.

Пусть $N(t_0) = i$, т. е. в момент t_0 занято i обслуживающих аппаратов. Тогда последующее течение процесса

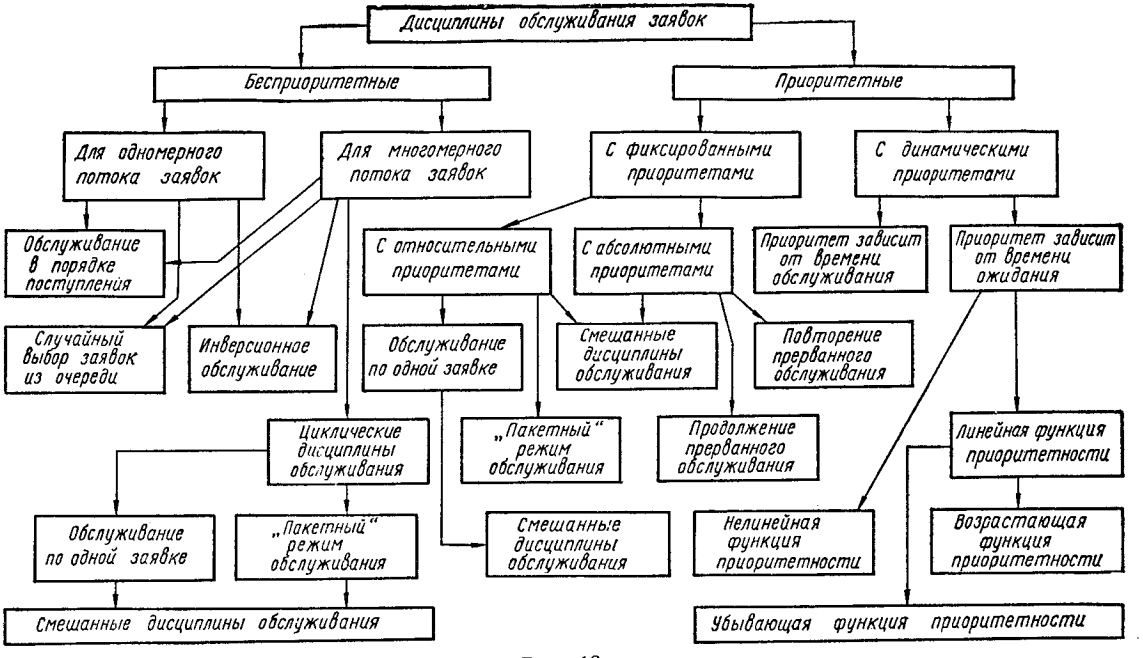

Дисциплины обслуживания заявок
Бесприоритетные
Приоритетные
Для одномерного потока заявок
Для многомерного потока заявок
С фиксированными приоритетами
С динамическими приоритетами
Обслуживание в порядке поступления
Случайный выбор заявок из очереди
Инверсионное обслуживание
С относительными приоритетами
С абсолютными приоритетами
Приоритет зависит от времени обслуживания
Приоритет зависит от времени ожидания
Обслуживание по одной заявке
Смешанные дисциплины обслуживания
Повторение прерванного обслуживания
Циклические дисциплины обслуживания
„Пакетный" режим обслуживания
Продолжение прерванного обслуживания
Линейная функция приоритетности
Обслуживание по одной заявке
„Пакетный" режим обслуживания
Смешанные дисциплины обслуживания
Нелинейная функция приоритетности
Возрастающая функция приоритетности
Смешанные дисциплины обслуживания
Убывающая функция приоритетности

Рис. 13.

независимо от всего, что происходило до момента t_0, определяется следующими факторами.

1. Моментами окончания обслуживания i-х требований, которые обслуживались в момент t_0.

2. Моментами поступления новых требований после t_0.

3. Длительностями обслуживания требований, поступивших после t_0.

Но ни один из этих трех случайных факторов не зависит от того, что происходило до момента t_0. Для первого фактора это вытекает из показательного закона распределения длительности обслуживания. Для второго фактора это следует из того, что поступающий поток требований простейший и, следовательно, не обладает последействием. Наконец, для третьего фактора это очевидно само собой. Таким образом, все три перечисленных фактора не зависят от «прошлого» системы, т. е. от течения процесса до момента t_0, а следовательно, не зависит от прошлого и течение процесса после момента t_0.

Случайные процессы, обладающие этим свойством, называются марковскими процессами или процессами Маркова. Если в некоторый момент t занято i обслуживающих аппаратов, то говорят, что система в момент t находится в состоянии i. Всего для системы возможно $n+1$ различных состояний (если n — число обслуживающих аппаратов). Обозначим через $P_{ik}(t)$ $(t>0,\ 0\leqslant i\leqslant n,\ 0\leqslant k\leqslant n)$ условную вероятность того, что система, находившаяся в некоторый момент в состоянии i, по истечении t единиц времени перейдет в состояние k. Очевидно, всегда

$$P_{ik}(t)\geqslant 0,\quad \sum_{k=0}^{n} P_{ik}(t)=1 \qquad (0\leqslant i\leqslant n;\ 0\leqslant k\leqslant n).$$

Если $t_1>0,\ t_2>0,\ 0<i\leqslant n,\ 0\leqslant k\leqslant n$, то имеет место соотношение

$$P_{ik}(t_1+t_2)=\sum_{r=0}^{n} P_{ir}(t_1)\,P_{rk}(t_2). \tag{6.1}$$

Эта формула справедлива только для процессов Маркова. В самом деле, только для процессов Маркова переходную вероятность $P_{rk}(t_2)$ можно считать независимой от i. Выражение (6.1), иногда называемое уравнением Уэпмана-Колмогорова, лежит в основе всех исследований о процессах Маркова.

Если в начальный момент $t = 0$ система находится в состоянии i, то вероятность застать ее в момент $t > 0$ в состоянии k равна $P_{ik}(t)$. Сделаем относительно начального момента допущение общего характера: в момент $t = 0$ можно считать известными не состояние системы, а лишь начальные вероятности $P_i(0)$ $(0 \leqslant i \leqslant n)$ различных состояний. Определим вероятность застать систему в момент t в состоянии k по формуле полной вероятности

$$P_k(t) = \sum_{i=0}^{n} P_i(0)\, P_{ik}(t). \tag{6.2}$$

Если в уравнении (6.1) умножить обе части на $P_i(0)$ и просуммировать по i от 0 до n, то, учитывая формулу (6.2), получаем

$$P_k(t_1 + t_2) = \sum_{i=0}^{n} P_i(t_1)\, P_{ik}(t_2), \tag{6.3}$$

которая означает, что если в момент времени t_1 в системе занято i аппаратов с вероятностью $P_i(t_1)$ $(i = 1, 2, \ldots, n)$, а условная вероятность перехода к k занятым аппаратам через промежуток времени t_2 равна $P_{ik}(t_2)$, то полная вероятность того, что в момент времени $t_1 + t_2$ занято k аппаратов, равна сумме произведений $P_i(t_1)\, P_{ik}(t_2)$ по всем возможным i.

Следовательно, задача состоит в отыскании вероятности застать систему в том или ином состоянии. Вероятности $P_k(t)$ меняются с течением времени и, кроме того, зависят еще от начальных данных, т. е. от чисел $P_i(0)$. Таким образом, искомые вероятности $P_k(t)$ можно определить лишь при данных t и $P_i(0)$. В приложениях, однако, обычно считают возможным говорить о вероятности застать систему в состоянии k, независимо от выбранного момента времени и от начальных данных. Оказывается, для многих процессов имеет место очень важное свойство, которое заключается в том, что $P_k(t)$ при $t \to \infty$ стремятся к конечным пределам p_k, где p_k — постоянные числа, независящие от начального состояния, в котором находилась система массового обслуживания.

Для того чтобы вероятности $P_k(t)$ при $t \to \infty$ стремились к независящим от начальных данных числам p_k необходимо и достаточно, чтобы к тем же пределам стремились соответственно переходные вероятности $P_{ik}(t)$ при любом значении i.

Согласно теореме Маркова, для любого транзитивного процесса Маркова $P_{ik}(t)$ $(0 \leqslant i \leqslant n;\ 0 \leqslant k \leqslant n)$ предел

$$\lim_{t \to \infty} P_{ik}(t) = p_k$$

существует и не зависит от i, доказывается наличие стационарного решения.

Теперь переходим к классическому методу Эрланга определения величин p_k, существование которых только что доказано. В дальнейшем придется иметь дело с промежутком времени бесконечно малой длины Δt. Условимся обозначать через $0(\Delta t)$ всякую бесконечно малую величину более высокого порядка, чем Δt.

Вероятность поступления одного требования за промежуток времени Δt есть величина $\lambda \Delta t + 0(\Delta t)$, а вероятность поступления более одного вызова — $0(\Delta t)$. С другой стороны, если какой-либо обслуживающий аппарат в данный момент занят, то вероятность оставаться занятым еще в течении времени Δt (или более) для него равна $e^{-\nu \Delta t}$. Если занято k обслуживающих аппаратов, то вероятность того, что все они останутся занятыми в течение промежутка времени Δt, равна $e^{-k\Delta t}$. Вероятность, что в течение промежутка времени Δt один из аппаратов освободится

$$1 - e^{-\nu k \Delta t} = \nu k \Delta t + 0(\Delta t).$$

Отсюда следует, что вероятность наступления в промежутке длины Δt по меньшей мере одного события при $\Delta t \to 0$ асимптотически пропорциональна Δt. Вероятность же наступления в промежутке длины Δt двух или более событий есть величина вида $0(\Delta t)$.

Эти замечания позволяют легко найти выражения переходных вероятностей $P_{ik}(\Delta t)$ при $\Delta t \to 0$. Прежде всего, если $|i - k| > 1$, то переход из состояния i в состояние k требует, очевидно, наступления по меньшей мере двух событий. Поэтому при $\Delta t \to 0$

$$P_{ik}(\Delta t) = 0(\Delta t) \qquad (|i - k| > 1). \tag{6.4}$$

Далее, для перехода из состояния $k < n$ в состояние $k + 1$ требуется поступление одного требования

$$P_{k,k+1}(\Delta t) = \lambda \Delta t + 0 \Delta t \qquad (0 \leqslant k \leqslant n).$$

Чтобы система перешла из состояния $k > 0$ в состояние $k - 1$, требуется освобождение одного из обслуживающих аппаратов. Учитывая, что освобождение двух и более

аппаратов за время Δt имеет порядок $0(\Delta t)$, получаем

$$P_{k,k-1}(\Delta t) = \nu k \Delta t + 0(\Delta t) \qquad (0 < k \leqslant n).$$

Наконец, по формуле (6.4) при $\Delta t \to 0$ находим

$$P_{k,k}(\Delta t) = 1 - P_{k,k+1}(\Delta t) - P_{k,k-1}(\Delta t) =$$
$$= 1 - \lambda \Delta t - \nu k \Delta t + 0(\Delta t) \qquad (0 \leqslant k \leqslant n),$$

где при $k = n$ второй, а при $k = 0$ третий член правой части необходимо заменить нулями:

$$P_{00}(\Delta t) = 1 - \lambda \Delta t + 0(\Delta t);$$
$$P_{kk}(\Delta t) = 1 - \lambda \Delta t - \nu k \Delta t + 0(\Delta t) \qquad (1 \leqslant k \leqslant n-1);$$
$$P_{nn}(\Delta t) = 1 - \nu n \Delta t + 0(\Delta t).$$

Таким образом, для всех вероятностей $P_{ik}(\Delta t)$ установлены очень простые асимптотические выражения с точностью до величины вида $0(\Delta t)$.

Теперь обратимся к уравнению (6.3), по которому при любом постоянном $t \geqslant 0$

$$P_k(t + \Delta t) = \sum_{i=0}^{n} P_i(t) P_{ik}(\Delta t).$$

Подставляя вместо $P_{ik}(\Delta t)$ найденные значения, находим:

$$P_0(t + \Delta t) = P_0(t) P_{00}(\Delta t) + P_1(t) P_{10}(\Delta t) + 0(\Delta t) =$$
$$= (1 - \lambda \Delta t) P_0(t) + \nu \Delta t P_1(t) + 0(\Delta t);$$
$$P_k(t + \Delta t) = P_{k-1}(t) P_{k-1,k}(\Delta t) + P_k(t) P_{kk}(\Delta t) +$$
$$+ P_{k+1}(t) P_{k+1,k}(\Delta t) + 0(\Delta t) = \lambda \Delta t P_{k-1}(t) +$$
$$+ (1 - \lambda \Delta t - \nu k \Delta t) P_k(t) + \nu(k+1) \Delta t P_{k+1}(t) + 0(\Delta t);$$
$$P_n(t + \Delta t) = P_{n-1}(t) P_{n-1,n}(\Delta t) + P_n(t) P_{nn}(\Delta t) + 0(\Delta t) =$$
$$= \lambda \Delta t P_{n-1}(t) + (1 - \nu n \Delta t) P_n(t) + 0(\Delta t).$$

После некоторых преобразований в полученных уравнениях разделим обе части на Δt:

$$\frac{P_0(t + \Delta t) - P_0(t)}{\Delta t} = -\lambda P_0(t) + \nu P_1(t) + \frac{0(\Delta t)}{\Delta t};$$

$$\frac{P_k(t + \Delta t) - P_k(t)}{\Delta t} = \lambda P_{k-1}(t) - (\lambda + \nu k) P_k(t) +$$

$$+ \nu(k+1) P_{k+1}(t) + \frac{0(\Delta t)}{\Delta t} \qquad (0 < k < n);$$

$$\frac{P_n(t + \Delta t) - P_n(t)}{\Delta t} = \lambda P_{n-1}(t) - \nu n P_n(t) + \frac{0(\Delta t)}{\Delta t}.$$

Если $\Delta t \to 0$, то убеждаемся в существовании производных всех функций $P_k(t)$ $(k = 0, 12, \ldots, n)$ и находим:

$$\left.\begin{aligned} &P_0'(t) = -\lambda P_0(t) + \nu P_1(t); \\ &P_k'(t) = \lambda P_{k-1}(t) - (\lambda + \nu k) P_k(t) + \nu (k+1) P_{k+1}(t) \\ &(0 < k < n); \\ &P_n'(t) = \lambda P_{n-1}(t) - \nu n P_n(t). \end{aligned}\right\} \quad (6.5)$$

Систему (6.5) обычно интегрируют при начальных условиях:

$$P_0(0) = 1; \quad P_k(0) = 0 \qquad (k = 1, 2, \ldots, n),$$

что соответствует случаю, когда система в начальный момент при $t = 0$ свободна. Решение системы при этих начальных условиях удовлетворяет нормировочному условию

$$\sum_{k=0}^{n} P_k(t) = 1.$$

Уравнения (6.5) называются уравнениями Эрланга. Они справедливы и для случая, когда потоки требований не являются простейшими, а представляют собой нестационарные пуассоновские потоки. В этом случае $\lambda = \lambda(t)$, $\nu = \nu(t)$ являются некоторыми функциями времени.

Хотя задача интегрирования системы уравнений (6.5) принципиально разрешима, нет необходимости искать решения системы дифференциальных уравнений, так как было доказано, что для любого k $(0 \leqslant k \leqslant n)$ существует предел

$$\lim_{t \to \infty} P_k(t) = p_k.$$

Тогда при $t \to \infty$ система дифференциальных уравнений превращается в систему алгебраических уравнений:

$$\begin{aligned} &0 = -\lambda p_0 + p_1; \\ &0 = \lambda p_{k-1} - (\lambda + \nu k) p_k + \nu (k+1) p_{k+1} \qquad (0 < k < n); \\ &0 = \lambda p_{n-1} - \nu n p_n. \end{aligned}$$

Эта простая система линейных уравнений вместе с нормировочным условием $\sum_{k=0}^{n} p_k = 1$ служит для однозначного определения искомых величин p_k.

Если положить

$$\lambda p_{k-1} - \nu k p_k = z_k \qquad (1 \leqslant k < n),$$

то $z_1 = \lambda p_0 - \nu p_1$, но из первого уравнения следует, что эта величина равна нулю, поэтому $z_1 = 0$.

Преобразуя второе уравнение, получаем

$$(\lambda p_{k-1} - \nu k p_k)\,[\lambda p_k - \nu (k+1)\, p_{k+1}] = 0,$$

т. е.

$$z_k - z_{k+1} = 0 \qquad (1 \leqslant k \leqslant n-1).$$

Последнее уравнение дает $z_n = 0$.

Таким образом:

$$z_1 = 0;$$
$$z_k - z_{k+1} = 0 \qquad (1 \leqslant k \leqslant n-1);$$
$$z_n = 0.$$

Подставляя $z_1 = 0$ во второе уравнение при $k = 1$, получаем, что $z_2 = 0$. Аналогично приходим к выводу, что $z_1 = z_2 = \ldots = z_n = 0$. Это означает, что

$$\lambda p_{k-1} = \nu k p_k$$

и

$$p_k = \frac{\lambda}{\nu k}\, p_{k-1} \qquad (k = 1, 2, \ldots, n).$$

Отсюда находим

$$p_1 = \frac{\lambda}{\nu}\, p_0; \quad p_2 = \frac{\lambda}{2\nu}\, p_1 = \left(\frac{\lambda}{\nu}\right)^2 \frac{1}{1 \cdot 2}\, p_0;$$

$$p_3 = \frac{\lambda}{3\nu}\, p_2 = \left(\frac{\lambda}{\nu}\right)^3 \frac{1}{1 \cdot 2 \cdot 3}\, p_0, \text{ и т. д.}$$

Легко доказать, что этот закон имеет место при любом k

$$p_k = \frac{p_0}{k!} \left(\frac{\lambda}{\nu}\right)^k \qquad (k = 1, 2, \ldots, n). \tag{6.6}$$

Для определения p_0 воспользуемся нормировочным условием. Тогда

$$p_0 \sum_{k=0}^{n} \frac{1}{k!} \left(\frac{\lambda}{\nu}\right)^k = 1,$$

откуда

$$p_0 = \frac{1}{\sum\limits_{k=0}^{n} \frac{1}{k!}\left(\frac{\lambda}{\nu}\right)^k}. \tag{6.7}$$

Чаще всего в формулах используется параметр $\rho = \frac{\lambda}{\nu}$. Находим

$$p_k = \frac{\rho^k}{k! \sum\limits_{k=0}^{n} \frac{1}{k!}\rho^k}. \tag{6.8}$$

Используя формулу (6.8), определяем вероятность отказа в обслуживании вновь поступившего требования в систему массового обслуживания

$$p_n = \frac{\rho^n}{n! \sum\limits_{k=0}^{n} \frac{\rho^k}{k!}} = p_0 \frac{\rho^n}{n!}. \tag{6.9}$$

На практике необходимо часто определять среднее число занятых обслуживанием приборов

$$M_з = \sum_{k=1}^{n} k p_k = \sum_{k=1}^{n} \frac{\rho^k}{(k-1)!} p_0. \tag{6.10}$$

Зависимости (6.6) — (6.10), характеризующие эффективность функционирования систем массового обслуживания с отказами, были получены при условии, что время обслуживания распределено по показательному закону. Справедливость формул Эрланга для абсолютно непрерывного закона распределения показана в работе [126]. Но в ней не приводятся доказательства единственности и эргодичности стационарного распределения, а лишь даны краткие указания на возможность такого доказательства. Б. А. Севастьянов доказал эргодическую теорему для марковских процессов [128].

В результате получено стационарное распределение, из которого, в частности, выводятся хорошо известные формулы Эрланга и показана их неизменность при любом распределении времени обслуживания, но конечном и постоянном значении его математического ожидания. Этот резуль-

тат позволяет значительно расширить область применения формул Эрланга для решения многих практических задач.

Вывод дифференциальных уравнений для вероятностей состояний (6.5) намного облегчается, если использовать определенный методический прием. Сущность его состоит в следующем. Составляют граф возможных состояний системы и стрелками показывают возможные переходы системы из одного состояния в другое. Возле каждой стрелки указывают интенсивность потоков событий, переводящих систему из состояния в состояние по данной стрелке. Граф состояний, на котором проставлены не только стрелки переходов но и интенсивность соответствующих потоков событий, будем называть размеченным графом состояний.

Если составлен размеченный граф состояний, то для составления дифференциальных уравнений для вероятностей $P_i\ (t)$ $(i = 0, 1, \ldots, n)$ можно использовать простое мнемоническое правило.

Производная $dP_i\ (t)/dt$ вероятности пребывания системы в состоянии x_i равна алгебраической сумме нескольких членов, число членов этой суммы равно числу стрелок на графе состояний системы, соединяющих состояние x_i с другими состояниями. Если стрелка направлена в состояние x_i, то член берется со знаком плюс; если стрелка направлена из состояния x_i, то со знаком минус. Каждый член суммы равен произведению вероятности того состояния, из которого направлена стрелка, на интенсивность потока событий, переводящего систему по данной стрелке.

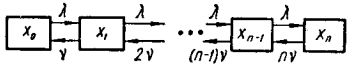


Рис. 14.

Составим размеченный граф состояний системы массового обслуживания с отказами.

Поясним порядок определения интенсивностей потоков событий (рис. 14). Когда система находится в состоянии x_0, на нее действует поток заявок с интенсивностью λ, переводящий систему в состояние x_1. Если система находится в состоянии x_1, то на нее действует уже два потока событий: 1) поток заявок с интенсивностью λ, который стремится перевести систему в состояние x_2 и 2) поток обслуживаний, который стремится перевести систему в состояние x_0. Интенсивность этого потока равна ν.

Если система находится в состоянии x_n, то на нее действует только один поток событий с плотностью $n\nu$, переводящий систему справа налево в состояние x_{n-1}.

В соответствии с мнемоническим правилом составления системы дифференциальных уравнений для вероятностей состояний получим:

$$\left.\begin{aligned} &P_0'(t) = -\lambda P_0(t) + \nu P_1(t); \\ &P_k'(t) = \lambda P_{k-1}(t) - (\lambda + \nu k) P_k(t) + \nu (k+1) P_{k+1}(t) \\ &(0 < k < n); \\ &P_n'(t) = \lambda P_{n-1}(t) - \nu n P_n(t). \end{aligned}\right\} \quad (6.11)$$

Система дифференциальных уравнений (6.11), полученная с помощью размеченного графа состояний, полностью совпадает с системой уравнений (6.5), полученной аналитически. Данная система уравнений Эрланга справедлива и в том случае, когда параметр поступающего потока требований меняется с течением времени.

Так как уравнения системы (6.11) имеют чисто локальную природу (относятся к некоторому определенному моменту времени t), то проведенный вывод этих уравнений остается в полной силе и в случае переменного параметра, только на место постоянного λ становится «мгновенное значение» $\lambda(t)$ этого параметра.

Используя формулы Эрланга, можно оценить функционирование в стационарном режиме простейших систем массового обслуживания с отказами при условии поступления в них пуассоновского потока требований. Однако на практике в реальных системах массового обслуживания поток требований может длиться ограниченное время, в течение которого система не успевает войти в стационарный режим. Поэтому при решении практических задач важно определить, при каких условиях процесс обслуживания в системе можно считать установившимся. Если же по условиям его нельзя считать установившимся, то какую ошибку вычислений можно получить при использовании формул, полученных для стационарного режима работы этой системы.

Очевидно, что вначале, сразу после включения системы в работу, протекающий в ней процесс еще не будет стационарным. В системе массового обслуживания (как и в любой динамической системе) возникает переходный, нестационарный процесс. Однако, спустя некоторое время, этот переходный процесс постепенно затухает и система переходит на стационарный, установившийся режим, вероят-

ностные характеристики которого уже не будут зависеть от времени.

Поэтому при оценке функционирования систем массового обслуживания на практике необходимо оценить влияние начального момента обслуживания на процесс затухания и определить те условия, при которых возможно с достаточной точностью сделать предположение о стационарности процесса работы исследуемых систем.

Приведем вывод зависимостей, которые учитывают начальный период работы системы массового обслуживания. Для простоты решения рассмотрим однолинейную систему с отказами. Пусть на обслуживающий прибор поступает в ограниченный отрезок времени простейший поток заявок с интенсивностью λ. Время обслуживания одной заявки случайное и распределено по показательному закону с параметром ν. За начало функционирования системы принимаем момент поступления первой заявки. Тогда, в начальный момент времени t_0 должно выполняться условие $P_1(0) = 1$.

Вероятность того, что обслуживающий прибор занят обслуживанием в момент времени $t + \Delta t$, слагается из вероятностей следующих двух несовместных событий с точностью до бесконечно малых более высокого порядка, чем Δt:

в момент времени t обслуживающий прибор был свободен, а за время Δt поступила заявка на обслуживание $\lambda \Delta t P_0(t)$;

в момент времени t обслуживающий прибор был занят обслуживанием и за время Δt не освободился от обслуживания $P_1(t)(1 - \nu \Delta t)$.

Тогда

$$P_1(t + \Delta t) = P_1(t)(1 - \nu \Delta t) + \lambda \Delta t P_0(t).$$

Если перенести $P_1(t)$ в левую часть и разделить на Δt, то получим

$$\frac{P_1(t + \Delta t) - P_1(t)}{\Delta t} = -\nu P_1(t) + \lambda P_0(t) + 0(\Delta t).$$

При $\Delta t \to 0$ это равенство превращается в дифференциальное уравнение

$$P'(t) = -\nu P_1(t) + \lambda P_0(t).$$

Учитывая нормирующее условие

$$P_0(t) + P_1(t) = 1,$$

уравнение приводим к линейному уравнению первого порядка

$$P_1'(t) = \lambda - (\lambda + \nu) P_1(t). \tag{6.12}$$

Для интегрирования полученного уравнения воспользуемся методом вариации произвольной постоянной. Решение этого уравнения целесообразно искать в виде

$$P_1(t) = U(t) e^{-(\lambda+\nu)t}, \tag{6.13}$$

где $U(t)$ — искомая функция t.

Совместное решение уравнений (6.12) и (6.13) дает результат

$$U(t) = \frac{\lambda}{\lambda+\nu} e^{-(\lambda+\nu)t} + k. \tag{6.14}$$

Учитывая начальное условие $P_1(0) = 1$, из формул (6.13) и (6.14) получаем

$$k = \frac{\nu}{\lambda+\nu}.$$

Отсюда решение уравнения (6.12) определяем в виде

$$P_1(t) = \frac{\lambda}{\lambda+\nu} + \frac{\nu}{\lambda+\nu} e^{-(\lambda+\nu)t}. \tag{6.15}$$

В частном случае, при $t \to \infty$, выражение (6.15) превращается в формулу Эрланга для одноканальной системы

$$P_1^э = \frac{\lambda}{\lambda+\nu}.$$

Вероятность того, что обслуживающий аппарат свободен, определяется следующим образом:

$$P_0(t) = 1 - P_1(t).$$

Математическое ожидание числа заявок, обслуженных за время t_n при конечном времени функционирования системы, может быть определено как произведение среднего времени $T_{ср}$, в течение которого обслуживающий аппарат был занят обслуживанием, на среднюю плотность обслуживания одной заявки ν:

$$M_t = T_{ср}\nu.$$

Среднее время

$$T_{ср} = \int_0^{t_n} P_1(t)\, dt.$$

Тогда

$$M_t = \nu \int_0^{t_n} P_1(t)\, dt.$$

Откуда, учитывая выражение (6.15), получаем

$$M_t = \frac{\rho}{\rho + 1} T + \frac{1}{(1+\rho)^2} [1 - e^{-(1+\rho)T}], \qquad (6.16)$$

где $\rho = \frac{\lambda}{\nu}$; $T = \frac{t_n}{\bar{t}_{обс}}$ — время поступления потока заявок в систему, выраженное в единицах $\bar{t}_{обс}$.

В стационарном режиме математическое ожидание числа заявок, обслуженных за время t_n,

$$M_{t\text{стац}} = \frac{\rho}{\rho + 1} T.$$

Абсолютная ошибка в определении среднего числа обслуженных заявок при предположении стационарности процесса

$$\Delta M_t = \frac{1}{(\rho + 1)^2} [1 - e^{-(\rho+1)T}]. \qquad (6.17)$$

Относительная ошибка зависимости

$$\Delta M_{t\text{отн}} = \frac{\Delta M_t}{M_{t\text{стац}}} = \frac{1}{T\rho\,(\rho + 1)} [1 - e^{-(\rho+1)T}]. \qquad (6.18)$$

Анализ уравнения (6.18) показывает, что с увеличением ρ процесс устанавливается значительно быстрее. Так, при $\rho \geqslant 2$ уже при длительности процесса обслуживания более $2\bar{t}_{обс}$ ошибка не превышает 10%. Получение аналитических зависимостей для $n > 1$ при неустановившемся процессе сопровождается громоздкими вычислениями, поэтому для значений $n > 1$ ошибки $\Delta M_{t\text{отн}}$ целесообразнее получить методом статистических испытаний.

Таким образом, при определении пропускных способностей систем массового обслуживания с отказами для процессов продолжительностью более (2—3) $\bar{t}_{обс}$ можно при расчетах с достаточной степенью точности использовать зависимости, полученные для стационарного процесса. Если точность полученных результатов необходимо повысить или продолжительность процесса мала, то следует при расчетах использовать формулы, выведенные для процессов конечной продолжительности.

Так как при решении различных прикладных задач во многих случаях требуется учитывать нестационарность процесса, то для оценки погрешности от допущения стационарности процесса, при проведении расчетов можно использовать вывод о том, что при заданных параметрах длительности процесса T и производительности системы ρ максимальная ошибка будет для одноканальной системы. Поэтому ошибка одноканальной системы может являться как бы верхним пределом. Прежде чем проводить расчеты, необходимо оценить ошибку от допущения стационарности процесса, а затем решать, какими зависимостями пользоваться.

4. СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ С ОЖИДАНИЕМ

Системы массового обслуживания с ожиданием распространены наиболее широко. Они могут быть чистого или смешанного типа. В чистой системе с ожиданием число мест в очереди и время ожидания в ней ничем не ограничены; каждая заявка рано или поздно будет обслужена. Для такой системы понятие «отказ» не имеет смысла.

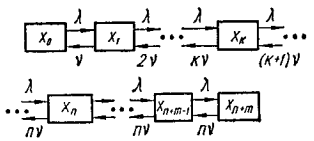


Рис. 15.

В системе с ожиданием смешанного типа возможны как отказы, так и ожидания заявки в очереди. Отказы могут быть связаны или с ограниченным числом мест в очереди, или с ограниченным временем ожидания, которым располагает заявка.

Рассмотрим некоторую обслуживающую систему, состоящую из n однотипных обслуживающих аппаратов. Если в момент поступления требования имеется хотя бы один свободный обслуживающий аппарат, то он немедленно приступает к обслуживанию этого требования. Если же все обслуживающие аппараты заняты, то очередное требование становится в очередь при условии, что в ней стоит меньше m требований. Если в очереди стоит m ранее поступивших требований, то очередное требование покидает систему обслуживания, или что то же самое, обслуживающая система отказывает требованию в обслуживании, если в ней находится $m + n$ требований. Очевидно, что при $m \to \infty$ эта задача превращается в задачу с неограниченным ожиданием начала обслуживания, а при $m = 0$ совпадает с задачей обслуживания с потерями.

Состояние рассматриваемой системы будем связывать с числом заявок, находящихся в системе (обслуживаемых и ожидающих в очереди):

x_k — в системе имеется k заявок ($k = 0, 1, 2, \ldots, n$), они обслуживаются k каналами, очереди нет; x_{n+r} — в системе имеется $n + r$ заявок ($r = 1, 2, \ldots, m$), n из них обслуживается в n каналах и r заявок находится в очереди.

Таким образом, система имеет $n + m + 1$ состояний. Граф состояний рассматриваемой системы показан на рис. 15.

Пользуясь мнемоническим правилом, составляем дифференциальные уравнения для вероятностей состояний системы:

$$\begin{aligned}
&P_0'(t) = -\lambda P_0(t) + \nu P_1(t);\\
&P_k'(t) = \lambda P_{k-1}(t) - (\lambda + k\nu) P_k(t) + (k+1)\nu P_{k+1}(t)\\
&\qquad\qquad (1 \leqslant k \leqslant n);\\
&P_k'(t) = \lambda P_{k-1}(t) - (\lambda + n\nu) P_k(t) + n\nu P_{k+1}(t)\\
&\qquad\qquad (n \leqslant k \leqslant m+n);\\
&P_{m+n}(t) = \lambda P_{m+n-1}(t) - n\nu P_{m+n}(t).
\end{aligned} \tag{6.19}$$

Рассмотрим стационарное состояние системы, при котором $t \to \infty$, а $P_k(t) \to 0$ и $P_k(t) \to p_k$. В этом случае уравнения состояний запишем в виде:

$$\left.\begin{aligned}
&0 = -\lambda p_0 + \nu p_1;\\
&0 = \lambda p_{k-1} - (\lambda + k\nu) p_k + (k+1)\nu p_{k+1} \qquad (1 \leqslant k \leqslant n);\\
&0 = \lambda p_{k-1} - (\lambda + n\nu) p_k + n\nu p_{k+1} \qquad (n \leqslant k \leqslant m+n);\\
&0 = \lambda p_{m+n-1} - n\nu p_{m+n}.
\end{aligned}\right\} \tag{6.20}$$

Нормировочное условие

$$\sum_{k=0}^{m+n} p_k = 1.$$

Для решения полученной алгебраической системы уравнений введем обозначения:

$$z_k = \lambda p_{k-1} - k\nu p_k \text{ при } 1 \leqslant k < n;$$
$$z_k = \lambda p_{k-1} - n\nu p_k \text{ при } n \leqslant k \leqslant m+n.$$

Система уравнений (6.20) в этих обозначениях принимает вид:

$$z_1 = 0; \quad z_k - z_{k+1} = 0 \text{ при } k \geqslant 1.$$

Отсюда получаем

$$k\nu p_k = \lambda p_{k-1} \text{ при } 1 \leqslant k < n;$$
$$n\nu p_k = \lambda p_{k-1} \text{ при } n \leqslant k \leqslant m + n.$$

Тогда

$$p_k = \frac{\rho^k}{k!} p_0 \qquad (1 \leqslant k < n); \tag{6.21}$$

$$p_k = \frac{\rho^k}{n!\, n^{k-n}} p_0 \qquad (n \leqslant k \leqslant m + n). \tag{6.22}$$

где $\rho = \frac{\lambda}{\nu}$.

Найдем вероятность отсутствия требований в системе из условия

$$\sum_{k=0}^{m+n} p_k = 1.$$

Подставляя в это уравнение значения p_k из формул (6.21) и (6.22), получаем

$$p_0 \left[\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \sum_{k=n}^{m+n} \left(\frac{\rho}{n} \right)^{k-n} \right] = 1.$$

Выражения, полученные для p_k, позволяют вычислить вероятность того, что очередное требование получит отказ в обслуживании

$$p_{m+n} = \frac{p_0}{n!\, n^m} \rho^{m+n}. \tag{6.23}$$

Определим вероятность того, что все обслуживающие аппараты будут заняты,

$$\Pi = \sum_{k=n}^{m+n} p_k.$$

Подставим в это выражение значение p_k из формулы (6.22)

$$\Pi = \sum_{k=n}^{n+m} \frac{p_0}{n!\, n^{k-n}} \rho^k = \frac{p_0}{n!} \rho^n \sum_{k=n}^{n+m} \left(\frac{\rho}{n} \right)^{k-n}.$$

Учитывая, что

$$\frac{p_0}{n!}\rho^n = p_n$$

и вычисляя сумму как сумму членов геометрической прогрессии со знаменателем $\frac{\rho}{n}$, получаем

$$\Pi = p_n \frac{1-\left(\frac{\rho}{n}\right)^{m+1}}{1-\frac{\rho}{n}},$$

откуда находим

$$p_n = \Pi \frac{1-\frac{\rho}{n}}{1-\left(\frac{\rho}{n}\right)^{m+1}}. \qquad (6.24)$$

Важнейшей характеристикой качества обслуживания для систем массового обслуживания с ожиданием является длительность ожидания требованием начала обслуживания. Длительность ожидания представляет собой случайную величину, которую обозначим через β. Запишем вероятность того, что время ожидания β превысит время t, по формуле полной вероятности следующим образом:

$$P\{\beta > t\} = \sum_{k=n}^{m+n-1} p_k P_k\{\beta > t\}, \qquad (6.25)$$

где $P_k\{\beta > t\}$ — условная вероятность того, что время ожидания превзойдет величину t при условии, что в момент поступления требования в очереди уже находилось k требований.

Обозначим через $q_s(t)$ вероятность обслуживания за время t ровно S требований. Поток обслуженных требований при условии, что все обслуживающие аппараты заняты обслуживанием и имеется очередь, будет простейшим. Вероятность того, что за время t закончится обслуживание ровно S требований

$$q_S(t) = \frac{(\nu n t)^S}{S!} e^{-\nu n t}.$$

Тогда

$$P_k\{\beta > t\} = \sum_{S=0}^{k-n} q_S(t).$$

Подставим в это равенство значения $q_S(t)$

$$P_k\{\beta > t\} = \sum_{S=0}^{k-n} \frac{(\nu t)^S}{S!} e^{-\nu t}.$$

Выражение (6.25) перепишем

$$P\{\beta > t\} = \sum_{k=n}^{m+n-1} p_k \sum_{S=0}^{k-n} \frac{(\nu t)^S}{S!} e^{-\nu t},$$

или, изменив порядок суммирования,

$$P\{\beta > t\} = e^{-\nu t} \sum_{S=0}^{m-1} \frac{(\nu t)^S}{S!} \sum_{k=S}^{m-1} p_{n+k}.$$

Вместо p_{n+k} подставим его значение

$$P\{\beta > t\} = e^{-\nu t} \sum_{S=0}^{m-1} \frac{(\nu t)^S}{S!} \sum_{k=S}^{m-1} \frac{p_0}{n!\, n^k} \rho^{k+n} =$$

$$= e^{-\nu t} \frac{p_0}{n!} \rho^n \sum_{S=0}^{m-1} \frac{(\nu t)^S}{S!} \sum_{k=S}^{m-1} \left(\frac{\rho}{n}\right)^k.$$

Так как

$$\frac{p_0}{n!} \rho_0^n = p_n.$$

Просуммировав вторую сумму как сумму членов геометрической прогрессии со знаменателем $\frac{\rho}{n}$ и подставив значение p_n из формулы (6.24), получим

$$P\{\beta > t\} = \frac{\Pi e^{-\nu t}}{1 - \left(\frac{\rho}{n}\right)^{m+1}} \sum_{S=0}^{m-1} \frac{\nu t}{S!} \left[\left(\frac{\rho}{n}\right)^S - \left(\frac{\rho}{n}\right)^m\right]. \quad (6.26)$$

Находим математическое ожидание длины очереди

$$M_1 = \sum_{k=n}^{m+n} (k-n)\, p_k.$$

Подставив значение p_k из формулы (6.22), имеем

$$M_1 = \frac{p_0}{n!} \rho^n \sum_{k=n}^{m+n} (k-n) \left(\frac{\rho}{n}\right)^{k-n},$$

или, заменив индекс суммирования на $k - n$, получим

$$M_1 = p_n \sum_{k=0}^{m} k \left(\frac{\rho}{n}\right)^k.$$

Сумму вычисляем по формуле суммы членов ряда

$$S_n = \frac{1}{1-a}\left(\frac{a - a^{n+1}}{1-a} - na^{n+1}\right),$$

где $a = \frac{\rho}{n}$.

Следовательно,

$$M_1 = \frac{p_n}{\left(1 - \frac{\rho}{n}\right)^2}\left[\frac{\rho}{n} - (m+1)\left(\frac{\rho}{n}\right)^{m+1} + m\left(\frac{\rho}{n}\right)^{m+2}\right]. \tag{6.27}$$

Среднее число требований, находящихся в обслуживающей системе,

$$M_2 = \sum_{k=1}^{m+n} k p_k.$$

Подставляя в это выражение значения p_k из формул (6.21) и (6.22), получаем

$$M_2 = \sum_{k=1}^{n-1} \frac{k}{k!}\rho^k p_0 + \sum_{k=n}^{m+n} \frac{k}{n!\, n^{k-n}}\rho^k p_0.$$

Изменяя индекс суммирования во второй сумме на $k - n$, находим

$$M_2 = p_0 \sum_{k=1}^{n-1} \frac{1}{(k-1)!}\rho^k + p_n \sum_{k=0}^{m} (n+k)\left(\frac{\rho}{n}\right)^k =$$

$$= M_1 + \frac{1 - \left(\frac{\rho}{n}\right)^{m+1}}{1 - \frac{\rho}{n}} n p_n + p_0 \sum_{k=1}^{n-1} \frac{1}{(k-1)!}\rho^k. \tag{6.28}$$

Среднее число аппаратов, свободных от обслуживания,

$$M_3 = \sum_{k=0}^{n-1} (n-k) p_k,$$

или, подставляя значение p_k из формулы (6.21),

$$M_3 = \sum_{k=0}^{n-1} \frac{n-k}{k!}\rho^k p_0. \tag{6.29}$$

Как уже указывалось, система с ожиданием может иметь или не иметь ограничения на ожидание. Рассмотрим теперь систему без ограничений. Все рассуждения, как и раньше, будем вести для простейшего потока требований и показательного распределения времени обслуживания.

Для системы массового обслуживания с ожиданием без ограничений имеет место бесчисленное множество дифференциальных уравнений, соответствующих бесчисленному множеству вероятностей:

$$\left.\begin{aligned} &P_0'(t) = -\lambda P_0(t) + \nu P_1(t); \\ &P_k'(t) = \lambda P_{k-1}(t) - (\lambda + k\nu) P_k(t) + (k+1)\nu P_{k+1}(t) \\ &(1 \leqslant k \leqslant n); \\ &P_k'(t) = \lambda P_{k-1}(t) - (\lambda + n\nu) P_k(t) + n\nu P_{k+1}(t) \qquad (k \geqslant n). \end{aligned}\right\} \tag{6.30}$$

Легко заметить, что первые n уравнений в данной системе полностью совпадают с первыми n уравнениями Эрланга. Действительно, до тех пор, пока не будет занят последний свободный обслуживающий аппарат, работа системы ничем не отличается от работы системы массового обслуживания с отказами. После того как последний обслуживающий аппарат будет занят, в системах массового обслуживания с отказами может наступить только освобождение одного из обслуживающих аппаратов, в то время как в системах с ожиданием без ограничений может произойти или освобождение одного из обслуживающих аппаратов, или образование очереди из одного поступившего требования.

Так же, как и уравнения Эрланга, уравнения (6.30) имеют предельные решения. Перейдем в системе (6.30) к пределу, когда $t \to \infty$, а $P_k'(t) \to 0$ и $P_k(t) \to p_k$:

$$\begin{aligned} 0 &= -\lambda p_0 + \nu p_1; \\ 0 &= \lambda p_{k-1} - (\lambda + k\nu) p_k + (k+1)\nu p_{k+1} \qquad (1 \leqslant k \leqslant n); \\ 0 &= \lambda p_{k-1} - (\lambda + n\nu) p_k + n\nu p_{k+1} \qquad (k \geqslant n). \end{aligned}$$

Общие решения данной системы алгебраических уравнений имеют вид:

$$p_k = \frac{\rho_k}{k!} p_0 \qquad (1 \leqslant k < n); \tag{6.31}$$

$$p_k = \frac{\rho^k}{n!\, n^{k-n}} p_0 \qquad (k \geqslant n). \tag{6.32}$$

Используя нормировочное условие

$$\sum_{k=0}^{\infty} p_k = 1 \qquad (k = 0,\ 1,\ 2,\ \ldots),$$

находим вероятность отсутствия требований в системе

$$p_0 = \frac{1}{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \sum\limits_{k=n}^{\infty} \left(\frac{\rho}{n}\right)^{k-n}}. \qquad (6.33)$$

Поскольку сумма

$$\sum_{k=n}^{\infty} \left(\frac{\rho}{n}\right)^{k-n}$$

является суммой членов геометрической прогрессии со знаменателем $\frac{\rho}{n}$, то эта сумма при $k \to \infty$ неограниченно возрастает, т. е. при $\frac{\rho}{n} \geqslant 1$ предельное значение вероятности того, что в системе массового обслуживания находится ровно k требований равно нулю. Число требований в очереди в этом случае неограниченно увеличивается.

Очевидно, что система массового обслуживания с ожиданием без ограничений должна проектироваться таким образом, чтобы выполнялось неравенство $\frac{\rho}{n} < 1$. При этом сумма имеет предел

$$\sum_{k=n}^{\infty} \left(\frac{\rho}{n}\right)^{k-n} = \frac{1}{1 - \frac{\rho}{n}}.$$

Вероятность того, что в системе массового обслуживания с ожиданием без ограничений все обслуживающие аппараты свободны

$$p_0 = \frac{1}{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\nu\rho^n}{(n-1)!\,(n\nu - \lambda)}}. \qquad (6.34)$$

Найдем вероятность того, что в системе массового обслуживания с ожиданием без ограничений занято ровно S обслуживающих аппаратов, подставив выражение (6.34)

в формулу (6.31)

$$p_S = \frac{\frac{\rho^S}{S!}}{\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\nu\rho^n}{(n-1)!\,(n\nu-\lambda)}} \qquad (1 \leqslant S \leqslant n-1). \quad (6.35)$$

Определим вероятность того, что в системе массового обслуживания с ожиданием без ограничений длина очереди равна r из формулы (6.32), подставив вместо k сумму $n + r$, а вместо p_0 выражение (6.34)

$$p_{n+r} = \frac{\frac{\rho^{n+r}}{n!\,n^r}}{\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\nu\rho^n}{(n-1)!\,(n\nu-\lambda)}} \qquad (r \geqslant 0). \quad (6.36)$$

Вероятность того, что все обслуживающие аппараты заняты,

$$p_з = \sum_{k=n}^{\infty} p_k = \frac{\frac{\nu\rho^n}{(n-1)!\,(n\nu-\lambda)}}{\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\nu\rho^n}{(n-1)!\,(n\nu-\lambda)}}. \quad (6.37)$$

В частности, для одноканальной системы массового обслуживания с ожиданием без ограничений, получаем следующие характеристики:

вероятность того, что единственный обслуживающий аппарат свободен

$$p_0 = 1 - \rho;$$

вероятность того, что обслуживающий аппарат занят

$$p_з = \rho;$$

вероятность того, что длина очереди равна r

$$p_{1+r} = \rho^{1+r}(1 + \rho).$$

Используя полученные характеристики, можно вычислить также: среднюю длину очереди, среднее число требований, находящихся в системе массового обслуживания, среднее число занятых обслуживающих аппаратов, среднее время ожидания в очереди, вероятность того, что время ожидания в очереди превышает длительность t и некоторые

другие. Приведем основные соотношения для получения перечисленных характеристик.

Средняя длина очереди, представляющая собой математическое ожидание числа требований, ожидающих начала обслуживания,

$$M_{\text{о}} = \sum_{r=0}^{\infty} r p_{n+r}$$

или, после подстановки значения p_{n+r},

$$M_{\text{о}} = p_n \sum_{r=0}^{\infty} r \left(\frac{\rho}{n}\right)^r,$$

где вероятность того, что все обслуживающие аппараты заняты, но очереди нет,

$$p_n = \frac{\rho^n}{n!} p_0.$$

Вычисляя сумму в выражении для $M_{\text{о}}$, окончательно получаем

$$M_{\text{о}} = \frac{n\nu\lambda p_n}{(\nu n - \lambda)^2} = \frac{\nu\lambda p_0 \rho^n}{(n-1)!\,(n\nu - \lambda)^2}. \qquad (6.38)$$

Среднее число требований, находящихся в системе (включая требования в очереди)

$$M_{\text{т}} = \sum_{k=1}^{n-1} k p_k + \sum_{k=n}^{\infty} k p_k.$$

Подставляя значения p_k из формулы (6.31) и (6.32), находим значение $M_{\text{т}}$ в виде:

$$M_{\text{т}} = M_{\text{о}} + p_0 \left[\frac{n\nu\rho^n}{(n-1)!\,(n\nu - \lambda)} + \sum_{k=1}^{n-1} \frac{\rho^k}{k-1} \right]. \qquad (6.39)$$

Среднее число занятых обслуживающих аппаратов в системе

$$M_{\text{а}} = \sum_{k=1}^{n} k p_k.$$

Используя формулы (6.31) и (6.32), перепишем $M_{\text{а}}$ в следующем виде:

$$M_{\text{а}} = \sum_{k=1}^{n-1} k p_k + n p_n.$$

После подстановки p_k и p_n имеем

$$M_a = \sum_{k=1}^{n-1} \frac{k\rho^k}{k!} p_0 + \frac{n\rho^n}{n!} p_0.$$

Окончательно запишем

$$M_a = \sum_{k=1}^{n} \frac{k\rho^k}{k!} p_0. \qquad (6.40)$$

Докажем, что вероятность того, что время ожидания в очереди превышает интервал времени t связана с вероятностью того, что все обслуживающие аппараты заняты $p_з$, соотношением

$$P\{t_{ож} > t\} = p_з e^{-(n\nu-\lambda)t} \qquad (t \geqslant 0).$$

Среднее время ожидания в очереди начала обслуживания

$$T_{ср} = M(t_{ож}) = -\int_0^\infty t dP(t_{ож} > t).$$

Подставив в эту формулу значение $P(t_{ож} > t)$ и проинтегрировав по частям, получим

$$T_{ср} = \frac{p_з}{n\nu - \lambda}. \qquad (6.41)$$

Для установившегося процесса получены зависимости, при помощи которых можно оценить качество обслуживания систем массового обслуживания с ожиданием. Однако на практике возможны случаи, в которых система массового обслуживания работает ограниченное время и процесс еще полностью не успевает установиться. Переходные режимы работы, характерные для работы систем массового обслуживания в течение относительно небольших промежутков времени, изменяют характеристики оценок функционирования систем, которые отличаются от характеристик для стационарных условий. Поэтому необходимо проверить, можно ли пользоваться зависимостями, полученными для установившегося процесса, а если можно, то каких погрешностей следует при этом ожидать.

Рассмотрим влияние нестационарности на вероятности состояний системы с ограниченной длиной очереди требований. Система дифференциальных уравнений, описываю-

щая все вероятностные состояния, имеет вид:

$$P_0'(t) = -\lambda P_0(t) + \nu P_1(t);$$

$$P_k'(t) = \lambda P_{k-1}(t) - (\lambda + k\nu) P_k(t) + (k+1)\nu P_{k+1}(t)$$
$$(1 \leqslant k < n);$$

$$P_k'(t) = \lambda P_{k-1}(t) - (\lambda + n\nu) P_k(t) + n\nu P_{k+1}(t)$$
$$(n \leqslant k < m+n);$$

$$P_{m+n}'(t) = \lambda P_{m+n-1}(t) - n\nu P_{m+n}(t).$$

Решение системы дифференциальных уравнений для n канальной системы очень громоздко, поэтому для простоты расчетов приведем решение только для одноканальной системы ($n = 1$), когда в очереди может находиться не более одной заявки ($m = 1$). В этом случае имеем следующую систему дифференциальных уравнений:

$$\left.\begin{aligned} P_0'(t) &= -\lambda P_0(t) + \nu P_1(t); \\ P_1'(t) &= \lambda P_0(t) - (\lambda + \nu) P_1(t) + \nu P_{1+1}(t); \\ P_{1+1}'(t) &= \lambda P_1(t) - \nu P_{1+1}(t). \end{aligned}\right\} \qquad (6.42)$$

Предположим, что с некоторого момента времени ($t = 0$) ожидается поступление заявок. Полученную систему (6.42) проинтегрируем при следующих условиях: $P_0(0) = 1$, $P_1(0) = P_2(0) = 0$.

Система дифференциальных уравнений (6.42) линейная с постоянными коэффициентами и ее можно проинтегрировать методом операционного исчисления. Не останавливаясь на промежуточных выкладках, запишем следующие решения:

$$P_0(t) = \frac{\nu^2}{rq} - \frac{\lambda}{2r} e^{rt} - \frac{\lambda}{2q} e^{qt}; \qquad (6.43)$$

$$P_1(t) = \frac{\lambda\nu}{rq} + \frac{\lambda(-\lambda + \sqrt{\lambda\nu})}{2r\sqrt{\lambda\nu}} e^{rt} + \frac{\lambda(\lambda + \sqrt{\lambda\nu})}{2q\sqrt{\lambda\nu}} e^{qt}; \qquad (6.44)$$

$$P_{1+1} = \frac{\lambda^2}{rq} + \frac{\lambda^2}{2r\sqrt{\lambda\nu}} e^{rt} - \frac{\lambda^2}{rq\sqrt{\lambda\nu}} e^{qt}, \qquad (6.45)$$

где

$$r = -\lambda - \nu + \sqrt{\lambda\nu};$$

$$q = -\lambda - \nu - \sqrt{\lambda\nu}.$$

Для стационарного процесса из формул (6.43) — (6.45) получим для одноканальной системы с ограничением на длину очереди при $m = 1$ следующие вероятности состояний, которые не зависят от t и являются постоянными:

$$p_0 = \frac{\nu^2}{\nu^2 + \lambda\nu + \lambda^2};$$

$$p_1 = \frac{\lambda\nu}{\nu^2 + \lambda\nu + \lambda^2};$$

$$p_{1+1} = \frac{\lambda^2}{\nu^2 + \lambda\nu + \lambda^2}.$$

Сравнивая значения основных параметров функционирования системы, полученные для неустановившегося и стационарного режимов работы, определим абсолютные погрешности:

$$M_0 = p_0 - P_0(t) = \frac{\lambda}{2r} e^{rt} + \frac{\lambda}{2q} e^{qt}; \qquad (6.46)$$

$$M_1 = p_1 - P_1(t) = \frac{\lambda(\lambda - \sqrt{\lambda\nu})}{2r\sqrt{\lambda\nu}} e^{rt} - \frac{\lambda(\lambda + \sqrt{\lambda\nu})}{2q\sqrt{\lambda\nu}} e^{qt}; \qquad (6.47)$$

$$M_{1+1} = p_{1+1} - P_{1+1}(t) = \frac{\lambda^2}{2q\sqrt{\lambda\nu}} e^{qt} - \frac{\lambda^2}{2r\sqrt{\lambda\nu}} e^{rt}. \qquad (6.48)$$

Относительные погрешности

$$\Delta M_0 = \frac{M_0}{p_0} = \frac{M_0(\nu^2 + \lambda\nu + \lambda^2)}{\nu^2}; \qquad (6.49)$$

$$\Delta M_1 = \frac{M_1}{p_1} = \frac{M_1(\nu^2 + \lambda\nu + \lambda^2)}{\lambda\nu}; \qquad (6.50)$$

$$\Delta M_{1+1} = \frac{M}{p_{1+1}} = \frac{M_{1+1}(\nu^2 + \lambda\nu + \lambda^2)}{\lambda^2}. \qquad (6.51)$$

Анализ результатов показывает, что использование формул, справедливых для стационарных условий, оправдано только в том случае, когда продолжительность процесса достаточно велика. При малом времени работы системы массового обслуживания применение формул (6.31) — (6.37) приводит к ошибкам, величина которых увеличивается с уменьшением времени. Для одноканальной системы при длительности процесса $T \geqslant 4\bar{t}_{\text{обс}}$ при $\rho \leqslant 0{,}5$ абсолютная ошибка не превышает 1,32%.

Необходимость рассмотрения функционирования системы массового обслуживания в неустановившемся режиме появляется тогда, когда время ее функционирования меньше, чем (2—4) средних значения времени обслуживания требования системы. Расчеты количественных характеристик функционирования систем массового обслуживания с ожиданием в различных режимах работы показали, что с увеличением числа обслуживающих аппаратов процесс установления режима в многоканальной системе проходит при прочих одинаковых условиях быстрее, чем в одноканальной.

Формулы для определения параметров функционирования системы в неустановившемся режиме для одноканальной системы получены в работах [81, 126]. Сравнивая эти формулы с формулами, рассчитанными для стационарных условий, можно оценить погрешность использования последних. Эта погрешность рассматривается как верхняя граница возможного отклонения результатов расчетов параметров функционирования систем для неустановившегося режима при использовании зависимостей, полученных для стационарных условий функционирования многоканальных систем массового обслуживания с ожиданием.

5. СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ С НЕОГРАНИЧЕННЫМ ЧИСЛОМ ОБСЛУЖИВАЮЩИХ АППАРАТОВ

Если число обслуживающих аппаратов бесконечно, то рассматриваемую систему массового обслуживания нельзя причислять к системам с потерями, так как потери становятся невозможными.

Будем считать, что в систему на обслуживание поступает простейший поток требований с параметром λ и время обслуживания подчинено показательному закону с параметром ν. Граф состояний данной системы массового обслуживания показан на рис. 16. Те рассуждения, которые привели нас к системе уравнений Эрланга, сохраняются почти полностью и в случае бесконечного числа обслуживающих аппаратов. Различие состоит, очевидно, лишь в том, что группа уравнений системы Эрланга, построенная для $0 < k < n$, теперь справедлива для любого $k > 0$.

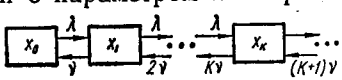


Рис. 16.

В соответствии с мнемоническим правилом составления системы дифференциальных уравнений для вероятностей состояний при неограниченном числе обслуживающих аппаратов получим:

$$P_0'(t) = -\lambda P_0(t) + \nu P_1(t);$$

$$P_k'(t) = \lambda P_{k-1}(t) - (\lambda + k\nu) P_k(t) + \nu(k+1) P_{k+1}(t).$$

Однако метод, которым мы пришли к формулам Эрланга, в случае бесконечного пучка оказывается неприменим, так как теорема Маркова, на которой он основан, существенным образом предполагает число состояний конечным. Систему массового обслуживания с неограниченным числом обслуживающих аппаратов легко исследовать методом производящих функций. Этот метод основан на введении вспомогательной функции, называемой производящей,

$$\Phi(t, x) = \sum_{k=0}^{\infty} P_k(t) x^k.$$

Если эта функция известна, то для определения исходных вероятностей $P_k(t)$ требуется разложить производящую функцию в ряд по степеням x, так как коэффициент при x^k равен $P_k(t)$. Опуская математические выкладки, приведем формулы в конечном виде.

Вероятность того, что в системе в момент времени t занято k обслуживающих аппаратов, при условии, что в начальный момент все они свободны,

$$P_k(t) = \frac{1}{k!}\left(\frac{\lambda}{\nu}\right)^k (1 - e^{-\nu t})^k e^{-\frac{\lambda}{\nu}(1 - e^{-\nu t})} \quad (k = 0, 1, 2, \ldots).$$

Среднее число аппаратов, занятых в момент времени t, при условии, что в начальный момент все они свободны,

$$M = \frac{\lambda}{\nu}(1 - e^{-\nu t}).$$

Вероятность того, что в установившемся процессе занято k обслуживающих аппаратов, независимо от начального состояния системы,

$$p_k' = \frac{1}{k!}\left(\frac{\lambda}{\nu}\right)^k e^{-\frac{\lambda}{\nu}} \qquad (k = 0, 1, 2, \ldots).$$

Вероятность того, что в установившемся режиме все обслуживающие аппараты свободны,

$$p_0 = e^{-\frac{\lambda}{\nu}}.$$

Задачи подобного типа встречаются при проектировании технических устройств и систем, когда большое значение имеет правильный выбор элементов.

6. ВЕРОЯТНОСТИ ПОТЕРЬ АЛГОРИТМОВ В СИСТЕМАХ УПРАВЛЕНИЯ С НЕСКОЛЬКИМИ ВЫЧИСЛИТЕЛЯМИ

Систему управления на базе цифровой вычислительной машины, в которую поступают на обработку сообщения, можно рассматривать как систему массового обслуживания с n однотипными обслуживающими аппаратами (вычислителями) и ограниченным числом мест ожидания m (емкость буферного накопителя). Предположим, что поток заявок, поступающий в такую систему, простейший с параметром λ. Закон распределения времени обслуживания определяется статистическими методами анализа численных значений времени обслуживания реальной системы. Наиболее широко распространен показательный закон распределения, при котором значительно упрощаются все расчеты по сравнению с произвольным законом распределения времени обслуживания.

При стационарных условиях функционирования систем массового обслуживания с отказами доказана справедливость формул Эрланга для любых законов распределения времени обслуживания [128]. Это предположение для других видов систем массового обслуживания, получивших наиболее широкое применение, проверено методом статистических испытаний [113]. Для этого рассчитывали вероятности состояний систем в стационарном режиме в широком диапазоне изменения основных параметров систем.

Сравнение результатов расчетов для различных законов распределения времени обслуживания (показательного, равномерного, усеченного нормального, Релея) показало их хорошее совпадение. Поэтому в дальнейшем для простоты будем предполагать, что время обслуживания одной заявки одним обслуживающим аппаратом (вычислителем)

распределено по показательному закону с параметром ν.

Для системы с n вычислителями, зная величину $\rho = \frac{\lambda}{\nu}$ и допустимую вероятность потери сообщения $P_{\text{пот}}$, определяем необходимую емкость буферного накопителя m

$$P_{\text{пот}} = \frac{\dfrac{\rho^{n+m}}{n!\, n^m}}{\sum\limits_{k=0}^{n-1} \dfrac{\rho^k}{k!} + \dfrac{\rho^n}{n!} \sum\limits_{k=n}^{n+m} \left(\dfrac{\rho}{n}\right)^{k-n}}. \qquad (6.52)$$

Покажем, что при значениях $\rho < n$, можно получить сколь угодно малую величину вероятности потери сообщения, увеличивая емкость буферного накопителя [22]. Для этого определим $\lim\limits_{m\to\infty} P_{\text{пот}}$. Выражение (6.52) перепишем в ином виде:

$$P_{\text{пот}} = \frac{\dfrac{\rho^{n+m}}{n!\, n^m}}{\sum\limits_{k=0}^{n-1} \dfrac{\rho^k}{k!} + \dfrac{\rho^n}{n!} \sum\limits_{k=n}^{n+m} \left(\dfrac{\rho}{n}\right)^{k-n}} =$$

$$= \frac{\dfrac{\rho^n}{n!}\left(\dfrac{\rho}{n}\right)^m}{\sum\limits_{k=0}^{n-1} \dfrac{\rho^k}{k!} + \dfrac{\rho^n}{n!} \sum\limits_{j=0}^{m} \left(\dfrac{\rho}{n}\right)^j} =$$

$$= \frac{\dfrac{\rho^n}{n!}\left(\dfrac{\rho}{n}\right)^m}{\sum\limits_{k=0}^{n-1} \dfrac{\rho^k}{k!} + \dfrac{\rho^n}{n!} \cdot \dfrac{1 - \left(\dfrac{\rho}{n}\right)^{m+1}}{1 - \dfrac{\rho}{n}}}.$$

Тогда

$$\lim_{m\to\infty} P_{\text{пот}} = \frac{\dfrac{\rho^n}{n!} \lim\limits_{m\to\infty} \left(\dfrac{\rho}{n}\right)^m}{\sum\limits_{k=0}^{n-1} \dfrac{\rho^k}{k!} + \dfrac{\rho^n}{n!} \lim\limits_{m\to\infty} \dfrac{1 - \left(\dfrac{\rho}{n}\right)^{m+1}}{1 - \dfrac{\rho}{n}}} =$$

$$= \frac{\frac{\rho^n}{n!} \cdot 0}{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!\left(1-\frac{\rho}{n}\right)} \cdot 1} = 0.$$

Таким образом, при неограниченном увеличении емкости буферного накопителя вероятность потери стремится к нулю при $\rho < n$, а следовательно, увеличивая число m, можно уменьшить $P_{\text{пот}}$.

Аналогичный результат получается и при $\rho = n$

$$P_{\text{пот}} = \frac{\frac{\rho^n}{n!}\left(\frac{\rho}{n}\right)^m}{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \sum\limits_{j=0}^{m} \left(\frac{\rho}{n}\right)^j} =$$

$$= \frac{\frac{\rho^n}{n!}\left(\frac{\rho}{n}\right)^m}{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!}(m+1)}.$$

Тогда

$$\lim_{\substack{m\to\infty \\ \rho=n}} P_{\text{пот}} = \frac{\frac{\rho^n}{n!} \lim\limits_{m\to\infty}\left(\frac{\rho}{n}\right)^m}{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \lim\limits_{m\to\infty}(m+1)} =$$

$$= \frac{\frac{\rho^n}{n!} \cdot 1}{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \cdot \infty} =$$

$$= \frac{\frac{\rho^n}{n!} \cdot 1}{\frac{\rho^n}{n!}\left(\frac{\sum\limits_{k=0}^{n-1} \frac{\rho^k}{k!}}{\frac{\rho^n}{n!}} + \infty\right)} = 0.$$

Если же $\rho > n$, то получить вероятность потери сколь угодно малой, увеличивая только емкость буферного накопителя, не удается. В этом случае вероятность потери заявки при $m \to \infty$ стремится к некоторой постоянной величине, меньше которой получить $P_{\text{пот}}$ при $\rho > n$ и неизменном числе вычислителей n невозможно. Определим величину, к которой стремится $P_{\text{пот}}$ при $m \to \infty$. Используя выражение (6.52) и учитывая, что $\rho \geqslant n$, находим

$$P_{\text{пот}} = \frac{\dfrac{\rho^n}{n!}\left(\dfrac{\rho}{n}\right)^m}{\displaystyle\sum_{k=0}^{n-1}\frac{\rho^n}{k!} + \frac{\rho^n}{n!}\cdot\frac{1-\left(\dfrac{\rho}{n}\right)^{m+1}}{1-\dfrac{\rho}{n}}} =$$

$$= \frac{\dfrac{\rho^n}{n!}\left(\dfrac{\rho}{n}\right)^m\left(\dfrac{n}{\rho}\right)^m}{\left(\dfrac{n}{\rho}\right)^m\displaystyle\sum_{k=0}^{n-1}\frac{\rho^n}{k!} + \frac{\rho^n}{n!}\cdot\frac{\left(\dfrac{n}{\rho}\right)^m - \left(\dfrac{n}{\rho}\right)^m\left(\dfrac{\rho}{n}\right)^{m+1}}{1-\dfrac{\rho}{n}}} =$$

$$= \frac{\dfrac{\rho^n}{n!}}{\dfrac{n}{\rho}\displaystyle\sum_{k=0}^{n-1}\frac{\rho^n}{k!} + \frac{\rho^n}{n!}\cdot\frac{\left(\dfrac{n}{\rho}\right)^m - \dfrac{\rho}{n}}{1-\dfrac{\rho}{n}}}.$$

Тогда

$$\lim_{\substack{m\to\infty\\ \rho>n}} P_{\text{пот}} = \frac{\dfrac{\rho^n}{n!}}{\displaystyle\sum_{k=0}^{n-1}\frac{\rho^n}{k!}\lim_{m\to\infty}\left(\frac{n}{\rho}\right)^m + \frac{\rho^n}{n!}\lim_{m\to\infty}\frac{\left(\dfrac{n}{\rho}\right)^m - \dfrac{\rho}{n}}{1-\dfrac{\rho}{n}}} =$$

$$= \frac{\dfrac{\rho^n}{n!}}{\displaystyle\sum_{k=0}^{n-1}\frac{\rho^n}{k!}0 + \frac{\rho^n}{n!}\cdot\frac{\dfrac{\rho}{n}}{\dfrac{\rho}{n}-1}} = \frac{1}{\dfrac{\rho}{\rho-n}} = 1-\frac{n}{\rho}. \qquad (6.53)$$

Следовательно, при $\rho > n$, только увеличивая емкость буферного накопителя m, невозможно получить вероятность потери сообщений меньше величины $1 - \frac{n}{\rho}$. Если же требуется обеспечить $P_{\text{пот}}$ меньше этой величины, то необходимо увеличить число вычислителей и довести это число до такой величины, чтобы

$$P_{\text{пот}} \leqslant 1 - \frac{n}{\rho},$$

или чтобы $\rho < n$. В этом случае возникает задача правильного выбора числа вычислителей и емкости буферного накопителя для построения оптимальной системы управления.

7. ВЫБОР ОПТИМАЛЬНОГО КОЛИЧЕСТВА ВЫЧИСЛИТЕЛЕЙ И ЕМКОСТИ БУФЕРНЫХ НАКОПИТЕЛЕЙ

Все большее развитие управляющих систем выдвинуло ряд интересных задач, которые могут быть решены методами теории массового обслуживания. Особенность работы таких систем заключается в том, что заявки поступают вначале не на обслуживающие приборы, а в буферный накопитель. По мере функционирования этих систем через некоторые интервалы времени заявки могут быть взяты из буферного накопителя для обслуживания. Примером такой системы может служить некоторая управляющая вычислительная система (УВС), у которой имеется блок сбора (буферная память) и блок обработки информации (вычислитель). Поступающая в систему информация записывается в буферную память, если в ней имеется место, иначе информация теряется. Из буферной памяти она поступает на блок обработки информации, если в нем закончится процесс обработки предшествующего массива информации.

Основной задачей в системах массового обслуживания является определение количественных показателей функционирования и их зависимостей от параметров входящего потока и структуры системы (ее состава и функциональных связей). Решение этой задачи дает возможность найти в системе слабые звенья, определить их влияние на эффективность обслуживания и найти пути их улучшения, или при заданных характеристиках потока требований и критерии качества обслуживания дать предложения о структуре

системы, которая обеспечит выполнение поставленной задачи.

Пусть система состоит из буферного накопителя заявок и обслуживающего прибора. Заявки поступают в накопитель в случайные моменты времени, распределенные по пуассоновскому закону с параметром λ заявок на единицу времени. Объем накопителя ограничен и равен m заявкам (условным единицам). Если поступившая заявка застанет накопитель полностью заполненным, то заявка теряется. Если прибор обработки свободен от обслуживания, то он обращается к накопителю и принимает на обслуживание очередную заявку. Время обработки сообщения прибором случайное и распределено по показательному закону с параметром ν заявок на единицу времени. В этом случае возникает задача определения минимального объема буферной памяти УВС, в которую поступает непрерывный поток сообщений. Емкость буферной памяти должна быть такой, чтобы обеспечить вероятность потери сообщений, не превышающую заданную величину допустимой вероятности потери $P_{\text{пот}}$.

При уменьшении требуемой величины $P_{\text{пот}}$, необходимо соответствующим образом изменить параметры УВС, чтобы вероятность потери сообщений оставалась меньше $P_{\text{пот}}$. Этого можно добиться увеличением производительности вычислительного устройства, увеличением числа вычислителей или количества ячеек памяти буферного накопителя.

Производительность можно увеличить за счет повышения частотных характеристик элементов структуры машины, т. е. за счет сокращения такта вычислительного устройства.

Однако нас интересует случай, когда дальнейшее уменьшение такта невозможно.

Когда дальнейшее увеличение быстродействия вообще невозможно, необходимо увеличивать либо количество вычислителей, либо объем буферного накопителя, чтобы потеря сообщений, поступающих в систему управления, была меньше или равна $P_{\text{пот}}$.

Увеличивая объем буферного накопителя (см. гл. VI,6), не всегда можно получить желаемый результат. Поэтому необходимо ввести критерий, по которому можно судить о качестве построенной системы. Одним из таких критериев для УВС служит количество условных единиц оборудования, из которых состоит система [145]. Чем меньше коли-

чество оборудования Q, тем лучше система. Следовательно, необходимо выбрать такое число вычислителей и емкость буферной памяти, чтобы система, удовлетворяя поставленным требованиям, содержала минимальное количество условных единиц оборудования.

Если предположить, что устройство обработки сообщений состоит из A условных единиц оборудования, а одна ячейка памяти буферного накопителя из B таких же единиц, то прибор обработки сообщений (арифметическое устройство) значительно сложнее по своей конструкции и количеству оборудования одной ячейки буферного накопителя, т. е. $A \gg B$. Для обеспечения вероятности потери сообщения меньше $P_{\text{пот}}$ обычно увеличивают емкость буферной памяти, оставляя неизменным количество вычислителей.

Но при значениях $\rho > 1$ невозможно получить вероятность потери меньше определенной величины только за счет увеличения числа мест ожидания (см. гл. VI,6). Поэтому для обеспечения заданной вероятности потери сообщений необходимо увеличить либо быстродействие вычислителя, либо увеличить их количество.

Кроме того, если сообщения, поступающие на обработку в УВС, реализуются достаточно простыми алгоритмами, и УВС является узкоспециализированной, т. е. решается определенный круг задач, то вычислительное устройство может быть упрощено, а следовательно, и уменьшится число условных единиц оборудования, входящих в его состав. В этом случае имеет смысл говорить о соизмеримости по количеству оборудования, вычислителя и условной ячейки памяти буферного накопителя, т. е. одно устройство обработки эквивалентно по количеству оборудования некоторому конечному числу условных ячеек памяти. Чтобы получить УВС с минимальным количеством единиц оборудования, обеспечивающую вероятность потери сообщений меньше или равную $P_{\text{пот}}$, не всегда целесообразно увеличивать емкость буферной памяти, оставляя неизменным число вычислителей, так как при определенных условиях выгоднее построить вычислительную систему, состоящую из нескольких вычислителей, сократив за счет этого емкость буферной памяти. Вся система в целом в этом случае содержит меньшее количество условных единиц оборудования, чем система с одним устройством обработки сообщений (вычислителем) и большим числом ячеек памяти буферного накопителя. Таким образом, при разработке УВС необходимо

правильно выбрать количество вычислителей и емкость буферной памяти при заданных параметрах потока входной информации, быстродействии вычислителя и допустимой вероятности потери заявки $P_{\text{пот}}$.

Эта задача может быть решена как задача теории массового обслуживания. Действительно, УВС может быть представлена как система массового обслуживания, в которой роль обслуживающих аппаратов выполняют вычислительные устройства, а ячейки памяти отождествляются с местами ожидания. Тогда для решения поставленной задачи необходимо исследовать систему массового обслуживания, состоящую из n обслуживающих аппаратов с ограниченной длиной очереди m при условии поступления простейшего потока сообщений с параметром λ для показательного закона распределения времени обслуживания с параметром ν.

Пусть задано, что один обслуживающий аппарат, содержащий A условных единиц оборудования, эквивалентен по количеству оборудования a местам ожидания, т. е. если одно место ожидания содержит B условных единиц оборудования, то

$$a = \frac{A}{B}.$$

Если при определенном количестве обслуживающих аппаратов n необходимо иметь m мест ожидания для обеспечения вероятности потери требования меньше $P_{\text{пот}}$ и отношение $\frac{m}{n} > a$, то чтобы получить систему с меньшим количеством оборудования, необходимо увеличить число обслуживающих аппаратов. Следовательно, требуется найти такие значения n и m, чтобы система обслуживания имела минимальное количество оборудования.

Определим вероятность потери требования в системе массового обслуживания, состоящей из n обслуживающих аппаратов и m мест ожидания,

$$P_{\text{пот}} = \frac{\dfrac{\rho^{n+m}}{n!\, n^m}}{\displaystyle\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \sum_{k=n}^{n+m} \left(\frac{\rho}{n}\right)^{k-n}},$$

где $\rho = \frac{\lambda}{\nu}$.

Сумму $\sum_{k=n}^{n+m} \left(\frac{\rho}{n}\right)^{k-n}$ найдем как сумму членов геометрической прогрессии со знаменателем $\frac{\rho}{n}$:

$$\sum_{k=n}^{n+m} \left(\frac{\rho}{n}\right)^{k-n} = \frac{1 - \frac{\rho}{n}\left(\frac{\rho}{n}\right)^{m}}{1 - \frac{\rho}{n}}.$$

Тогда

$$P_{\text{пот}} = \frac{\frac{\rho^{n+m}}{n!\, n^m}}{\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!} \cdot \frac{\left(1 - \frac{\rho}{n}\right)^{m+1}}{1 - \frac{\rho}{n}}}.$$

Откуда имеем:

$$P_{\text{пот}} \sum_{k=0}^{n-1} \frac{\rho^k}{k!} + P_{\text{пот}} \frac{\rho^n}{n!} \cdot \frac{1 - \left(\frac{\rho}{n}\right)^{m+1}}{1 - \frac{\rho}{n}} = \frac{\rho^{n+m}}{n!\, n^m};$$

$$P_{\text{пот}}\, n! \left(1 - \frac{\rho}{n}\right) \sum_{k=0}^{n-1} \frac{\rho^k}{k!} + P_{\text{пот}}\, \rho^n \left[1 - \left(\frac{\rho}{n}\right)^{m+1}\right] = \frac{\rho^{n+m}\left(1 - \frac{\rho}{n}\right)}{n^m};$$

$$P_{\text{пот}} \left[n! \left(1 - \frac{\rho}{n}\right) \sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \rho^n\right] = \frac{\rho^{n+m}\left(1 - \frac{\rho}{n}\right)}{n^m} +$$

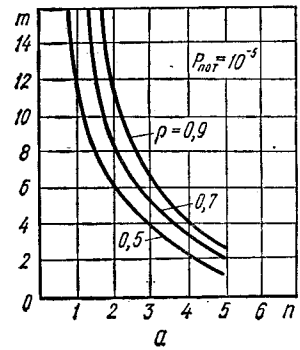
$$+ P_{\text{пот}} \rho^n \left(\frac{\rho}{n}\right)^{m+1} = \frac{\rho^m}{n^m}\left(\rho^n - \frac{\rho^{n+1}}{n} + P_{\text{пот}} \frac{\rho^{n+1}}{n}\right) =$$

$$= P_{\text{пот}} \left[n! \left(1 - \frac{\rho}{n}\right) \sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \rho^n\right].$$

Или, решив уравнение относительно m, получим

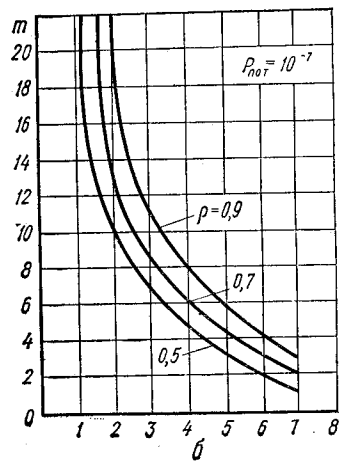
$$m = \frac{\ln \dfrac{P_{\text{пот}}\left[n!\left(1-\dfrac{\rho}{n}\right)\sum\limits_{k=0}^{n-1}\dfrac{\rho^k}{k!}+\rho^n\right]}{\rho^n\left[\dfrac{\rho}{n}(P_{\text{пот}}-1)+1\right]}}{\ln\dfrac{\rho}{n}}. \qquad (6.54)$$

Выражение (6.54) позволяет определить, какое количество мест ожидания потребуется при определенном числе обслуживающих аппаратов, если заданы допустимая вероятность потери требований $P_{\text{пот}}$ и ρ. Требуется построить такую систему массового обслуживания, которая обеспечивала бы вероятность потери требования меньше или равную $P_{\text{пот}}$ и имела минимальное количество оборудования. Следовательно, необходимо найти величины m и n, при которых выполняются поставленные условия.

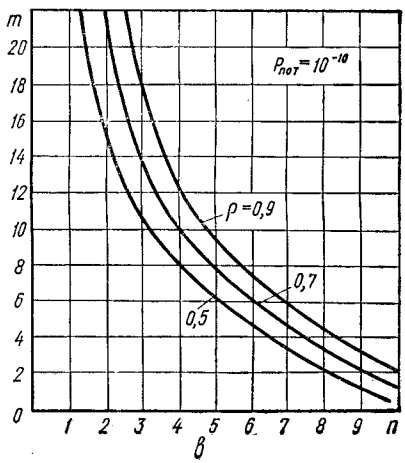


а

Введем понятие $N_{\text{пр}}$ — приведенное количество оборудования,



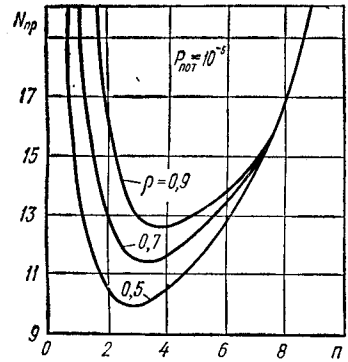
б



в

Рис. 17.

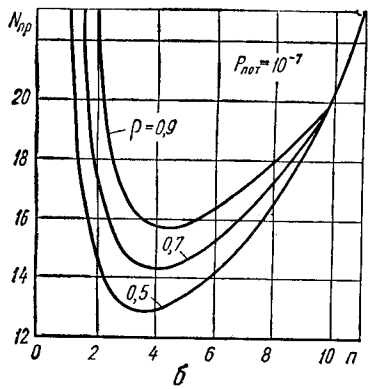
т. е. число условных единиц оборудования, содержащееся в данной системе обслуживания. Примем, что одно место ожидания состоит из одной условной единицы оборудования, тогда обслуживающий аппарат содержит a условных единиц. Определим величину $N_{\text{пр}}$ для системы массового обслуживания, состоящей из n обслуживающих аппаратов и m мест ожидания:



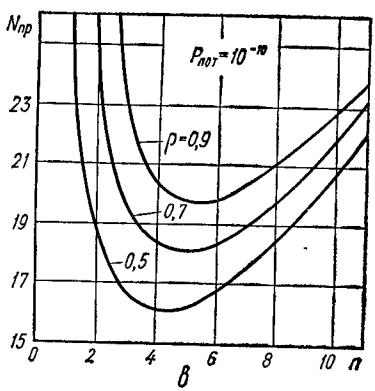
а

$$N_{\text{пр}} = an + m.$$

Подставим вместо m его значение из формулы (6.54)



б



в

Рис. 18.

$$N_{\text{пр}} = an + \frac{\ln \dfrac{P_{\text{пот}}\left[n!\left(1-\dfrac{\rho}{n}\right)\displaystyle\sum_{k=0}^{n-1}\frac{\rho^k}{k!}+\rho^n\right]}{\rho^n\left[\dfrac{\rho}{n}\left(P_{\text{пот}}-1\right)+1\right]}}{\ln\dfrac{\rho}{n}}. \tag{6.55}$$

Таким образом, для получения системы массового обслуживания, имеющей минимальное количество оборудования

необходимо найти такие значения m и n, при которых величина $N_{\text{пр}}$ минимальна для заданных $P_{\text{пот}}$, a и ρ.

На рис. 17 и 18 показаны графики зависимостей $m = f(n)$ и $N_{\text{пр}} = f(n)$ для $a = 2$. По этим кривым можно определить величины n и m, при которых система имеет минимальное количество оборудования.

Например, требуется определить число обслуживающих аппаратов и количество мест ожидания для системы массового обслуживания с параметром обслуживания для одного обслуживающего аппарата $\rho = 0{,}9$, $P_{\text{пот}} = 10^{-7}$ и $a = 2$. Прежде всего, по рис. 18, *б* находим минимальное $N_{\text{пр}}$ для $\rho = 0{,}9$ и $n = 5$. По рис. 17, *б* для $\rho = 0{,}9$ определяем необходимое количество мест ожидания, т. е. $m = 6$. Таким образом, рассматриваемая система обслуживания содержит минимальное количество оборудования при $n = 5$ и $m = 6$.

Предлагаемый метод определения числа вычислителей и емкости буферного накопителя может быть использован не только для получения УВС, содержащих минимальное количество оборудования, а также для получения систем, оптимальных по стоимости, весу и т. д., если предположить, что ячейка памяти по стоимости (весу) в A раз дешевле (легче) одного вычислителя.

Глава VII

УПРАВЛЯЮЩАЯ ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА С НЕСКОЛЬКИМИ ИНФОРМАЦИОННЫМИ ПОТОКАМИ КАК СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ С ПРИОРИТЕТАМИ

1. КРИТЕРИИ ПРИОРИТЕТОВ ДЛЯ УВС

В последние годы большое развитие получили математические методы оптимального управления, а также оптимального синтеза систем управления. В теории массового обслуживания обычно рассматривается лишь однолинейная система без мест для ожидания или с неограниченным числом мест для ожидания в установившемся режиме.

При проектировании систем управления возникает ряд задач по обслуживанию нескольких входящих информаци-

онных потоков с приоритетами. Оптимальная организация вычислений зависит от числа этих потоков, объема памяти и быстродействия УВС, наличия стандартных программ и требований к точности вычислений.

Наиболее важный вопрос — как по информации о состоянии системы выбирать требования на обслуживание, чтобы минимизировать суммарный штраф за функционирование системы в стационарном режиме.

Такие задачи возникают в системах управления производством посредством УВС, в системах сбора и обработки информации, в системах управления движением и т. д. Пусть в некоторую систему управления в случайные моменты времени поступают сигналы о необходимости выполнения управляющей вычислительной машиной некоторых программ для выработки управляющих воздействий. Требуется выбрать оптимальную последовательность выполнения программ.

Возникающие при этом задачи массового обслуживания нескольких потоков требований различных типов весьма разнообразны. Действительно, по-разному решаются задачи, если требование первого типа застает все аппараты занятыми.

Могут быть системы массового обслуживания, в которых требование старшего ранга не прерывает уже начатых обслуживаний, но становится впереди всех требований низших рангов. Такое преимущество действует в пределах буферного накопителя, и дисциплина ожидания ограничивается определением правил выбора требования, обслуживание которого начнется в момент, когда освободится один из обслуживающих аппаратов, если при этом ожидают обслуживания несколько требований. Это относительные преимущества (приоритеты). Введение относительных преимуществ дает возможность уменьшить время ожидания некоторых требований, рассматриваемых как срочные. Но при такой дисциплине ожидания требование первоочередной категории, заставшее аппарат занятым даже требованием с меньшим приоритетом, должно дожидаться окончания текущего обслуживания. Можно ускорить обработку первоочередных требований, упразднив это ограничение. При этом обслуживание требований низших рангов прерывается при поступлении требований более высоких рангов. Это абсолютные преимущества (приоритеты). Вторая схема в свою очередь подразделяется на две: в первой из них

при возобновлении обслуживания прерванного требования учитывается время, ранее затраченное на его обслуживание, во второй — это время теряется, и обслуживание начинается сначала. Это встречается при работе управляющих вычислительных машин, когда одни отказы прерывают вычисления, но не требуют после своего исправления возобновлять ранее сделанные расчеты; отказы же другого типа вносят ошибку в ранее полученные вычисления и требуют проведения всех расчетов заново.

Если рассматривать случай, когда все требования одинаковы, т. е. когда предполагается, что стоимость ожидания и распределение длительности обслуживания одинаковы для всех требований, то, очевидно, нет никакого смысла вводить абсолютные преимущества. Как правило, признаком, по которому определяется дисциплина обслуживания, является порядок поступления требований, но встречаются также и другие дисциплины обслуживания, при которых поступление требований принимается только частично или даже вовсе не принимается во внимание. Изучать системы массового обслуживания такого рода важно, так как для большого числа задач учитывают возможность выхода из рабочего состояния самих обслуживающих аппаратов. Так в сложных вычислительных устройствах из-за отказа того или иного элемента может наступить отказ всего устройства, в результате чего прекращается обслуживание поступающих требований до обнаружения неисправности и ее ликвидации.

Имеется большое число задач, в которых необходимо учитывать возможность выхода из рабочего состояния обслуживающего прибора: 1) в одних случаях требования остаются ожидать окончания восстановления, как бы долго ни продолжалось ожидание; 2) требование теряется мгновенно, как только обслуживающий аппарат потерял способность работать; 3) время пребывания требования в системе массового обслуживания не превосходит некоторого времени t. Различие может происходить и по иным причинам: 1) обслуживающий аппарат выходит из рабочего состояния только во время работы; 2) выходит из строя как во время работы, так и во время ожидания требования; 3) вероятность выхода из строя в период рабочего состояния больше, чем в период ожидания работы. Требование, которое обслуживалось на испортившемся аппарате, может теряться независимо от того, имеются или не имеются в системе

свободные обслуживающие аппараты, но может быть и иная система, когда требование с испортившегося прибора переходит на любой из свободных. Все эти задачи могут быть интерпретированы как задачи массового обслуживания с преимуществом, если рассматривать поток поломок, как поток наивысшего приоритета.

Рассмотрим систему массового обслуживания, в которую поступают несколько потоков требований. Будем предполагать, что требования различных потоков обладают отличительными признаками, которые приписаны каждому потоку:

определенная стоимость единицы времени ожидания данного требования, или, по крайней мере, «показатель срочности», который характеризует срочность обработки рассматриваемого требования (этот признак обусловлен причинами, не связанными с работой самой системы);

особый закон распределения длительности обслуживания, отличающийся от закона, относящегося к случайно выбранному требованию;

два вышеуказанных признака одновременно.

Требованиями могут быть сообщения, каждому из которых приписывается определенная степень срочности; тогда продолжительности обслуживания соответствуют каждой из указанных степеней срочности. Сообщения классифицируются по продолжительности обслуживания на однородные или неоднородные.

Задача состоит в выборе оптимальной дисциплины ожидания при определенном критерии (например, средней стоимости ожидания для совокупности требований), т. е. оптимального управления системой массового обслуживания.

Определение оптимального управления даже в простейших случаях является достаточно сложной задачей, не поддающейся аналитическим расчетам. Поэтому иногда разумно отказаться от поисков оптимального управления системы массового обслуживания, а ограничиться управлением, близким к оптимальному, или, что то же самое, искать оптимальное управление в более узком классе управлений.

В системах массового обслуживания с несколькими входящими информационными потоками таким более узким классом управлений является выбор оптимальной последовательности приоритетов. При правильном выборе последовательности приоритетов система с приоритетами лучше,

чем система без них, но последняя может оказаться лучше системы с приоритетами, если последовательность приоритетов выбрана неправильно.

Поэтому, прежде чем переходить к исследованию систем массового обслуживания с приоритетами, рассмотрим выбор последовательности приоритетов для систем массового обслуживания с ограниченным числом мест для ожидания. Предположим, что требования разбиты на конечное число категорий z, с каждой из которых связывается определенная степень срочности; требования одной и той же категории имеют один и тот же закон распределения длительности обслуживания, но этот закон не обязательно одинаков для различных категорий требований. Входящий поток требований одинаковой категории — простейший с интенсивностью λ_i.

Если средняя продолжительность обслуживания одна и та же для всех входящих потоков, то можно показать, что средняя длительность ожидания не зависит от присвоенных требованиям приоритетов и равна среднему времени ожидания в упорядоченной очереди. Эффект преимущества состоит только в уменьшении времени ожидания требований с большим преимуществом в ущерб требованиям с меньшим преимуществом. Поэтому присваивают приоритеты в порядке убывания срочности.

Если же средняя длительность обслуживания неодинакова для всех категорий требований, следует ожидать, что средняя длительность ожидания зависит от присвоенных требованиям приоритетов. Можно показать, что введение преимуществ улучшает общее функционирование системы, если более высокие приоритеты присваиваются требованиям, имеющим в среднем меньшую длительность обслуживания.

Внешние причины, которые не связаны с функционированием системы и заставляют приписывать тем или иным категориям большую или меньшую срочность, могут быть сведены к различию в стоимости времени ожидания. Отсюда следует, что потери, связанные с образованием очереди, зависят только от среднего времени ожидания. Тогда, изменяя порядок приоритетов, можно изменять средние потери до тех пор, пока в системе не установится такой порядок приоритетов, что меньшему значению отношения

$$\frac{\text{среднее значение длительности обслуживания}}{\text{стоимость задержки в течение единицы времени}}$$

соответствует высший приоритет. Тогда получим последовательность приоритетов, минимизирующую средние потери. Анализ известных систем приоритетов для двух категорий требований показывает, что выбор преимуществ является выгодным, если средняя стоимость ожидания для совокупности требований имеет меньшее значение, чем эта же величина, полученная при перемене порядка приоритетов. Показано, что большее преимущество нужно предоставлять той категории требований, у которой произведение $c_i \mu_i$ имеет большее значение, где μ_i — интенсивность обслуживания требований i-го потока; c_i — стоимость времени ожидания. В этом случае стоимость получается меньше стоимости единственной упорядоченной очереди.

В других случаях с целью оптимального функционирования вводится экономический критерий, т. е. ценность заявки каждого потока характеризуется некоторой величиной α_i. Тогда выбранная последовательность приоритетов является оптимальной, если в установившемся режиме суммарная ценность всех полностью обслуженных в единицу времени заявок будет максимальной, т. е. необходимо максимизировать выражение

$$\sum_{i=1}^{N} \alpha_i \lambda_i \vartheta_i,$$

где ϑ_i — вероятность полного обслуживания заявки из i-го входящего потока.

Поставим задачу определения критерия приоритета при условиях обеспечения минимальной длины очереди с одной стороны и минимальной вероятности потери требований с другой стороны. Пусть на некоторое обслуживающее устройство поступает r потоков требований $L_1, L_2, \ldots, L_r$. Заявкам потока L_i $(i = 1, 2, \ldots, r)$ присваиваем i-й приоритет. Будем говорить, что заявки потока L_i имеют более высокий приоритет по сравнению с заявками потока L_j, если $i < j$.

Определенный интерес представляет образование очередей в системах массового обслуживания с ожиданием при одном обслуживающем аппарате. Образование очереди при обслуживании с преимуществом может быть организовано следующими основными способами.

1. Система обслуживания, в которую поступает r потоков заявок $L = \{L_1, L_2, \ldots, L_r\}$ имеет m мест ожидания. Причем все количество мест ожидания разбито на группы

$m_1, m_2, \ldots, m_r$, каждая из которых служит для приема требований определенного потока $L_i \in L$, т. е. группа мест ожидания m_i служит для приема требований потока с приоритетом i, т. е. потока L_i. В первую очередь обслуживаются требования наивысшего приоритета, т. е. требования, ожидающие обслуживания на местах ожидания группы m_1. Если же группа мест ожидания m_1 свободна, то обслуживающий аппарат приступает к обслуживанию требований, находящихся в группе m_2, т. е. после окончания обслуживания требований из группы m_i аппарат приступает к обслуживанию требований, ожидающих в группе m_{i+1}. При такой организации обслуживания и образования очереди не все группы $m_1, m_2, \ldots, m_r$ будут использоваться с одинаковой нагрузкой. Так, например, в какой-то момент времени вся группа мест ожидания m_1 может оказаться занятой, и требования потока L_1 будут теряться, в то время как другая группа мест ожидания имеет свободные места, но требования потока L_1 не могут поступать на места m_i.

2. Система имеет m мест ожидания. Обслуживающий аппарат всегда приступает к обслуживанию требования, стоящего на первом месте. Возникает вопрос, каким образом размещать требования в очереди. Так как они различаются по времени, необходимому для их обслуживания, и требуется получить минимальную очередь, то может оказаться целесообразной следующая система преимуществ. Из очереди на обслуживание поступает первое требование, для которого время обслуживания минимально. Такая система приоритетов обеспечивает уменьшение длины очереди. Возможна другая система преимуществ, при которой в первую очередь обслуживаются требования, обладающие минимальным допустимым временем ожидания. В этом случае уменьшается вероятность потери требований за счет ожидания, но значительно увеличивается очередь.

Если же к системе обслуживания одновременно предъявляются требования уменьшения длины очереди и вероятности потери требований за счет времени ожидания, то необходимо выбрать такую систему преимуществ, которая смогла бы удовлетворить в какой-то мере этим требованиям.

С точки зрения получения минимальной очереди необходимо обслуживать требования, обладающие минимальным временем обслуживания. Эта система преимуществ ха-

рактеризуется коэффициентом

$$K_1 = \frac{t_{max}}{t},$$

где t — время, необходимое для обслуживания поступившего требования; t_{max} — время, необходимое для обслуживания требования, обладающего самым длительным временем обслуживания.

В первую очередь обслуживаются требования, имеющие K_{1max}. В этом случае очередь минимальна. Чтобы уменьшить вероятность потери требований за счет ожидания, необходимо в первую очередь обслуживать требования, обладающие минимальным допустимым временем ожидания. Эта система преимуществ характеризуется коэффициентом

$$K_2 = \frac{\tau_{max}}{\tau},$$

где τ — допустимое время ожидания поступившего требования; τ_{max} — допустимое время ожидания для требований, обладающего максимальным допустимым временем ожидания.

Согласно этой системе преимуществ в первую очередь обслуживаются требования, имеющие K_{2max}. Для учета обеих систем преимуществ необходимо выбрать такую систему приоритетов, которая включала бы в себя оба коэффициента K_1 и K_2.

Если взять сумму K_1 и K_2, то такая система преимуществ не полностью отвечает двум поставленным требованиям, так как в некоторых случаях более важно получить минимальную длину очереди, а в других случаях более важно не потерять требование за счет времени ожидания.

Введем следующие понятия и обозначения: P — допустимая вероятность потери данного требования; Q — допустимая вероятность увеличения длины очереди. Тогда величины $1/P$ и $1/Q$ характеризуют веса коэффициентов K_1 и K_2 соответственно.

Коэффициент, характеризующий систему преимуществ, удовлетворяющую двум поставленным требованиям,

$$K = \frac{1}{Q} K_1 + \frac{1}{P} K_2 = \frac{1}{Q}\left(K_1 + \frac{Q}{P} K_2\right).$$

Требования, поступающие в систему массового обслуживания, располагаются в очереди в порядке убывания величины K.

Рассмотрим на примере, что дает предлагаемая система приоритетов. Пусть на информационно-логическую машину поступает шесть типов потоков информации ($r = 6$). Каждый поток является простейшим с плотностью соответственно:

$$\lambda_1 = 0{,}25; \quad \lambda_2 = 0{,}15; \quad \lambda_3 = 0{,}4;$$
$$\lambda_4 = 2; \quad \lambda_5 = 7{,}5; \quad \lambda_6 = 1{,}3.$$

Время обработки поступающей информации распределено по показательному закону со средним временем соответственно:

$$t_1 = 0{,}2; \quad t_2 = 0{,}4; \quad t_3 = 0{,}25;$$
$$t_4 = 0{,}1; \quad t_5 = 0{,}05; \quad t_6 = 0{,}15.$$

При обработке поступающей информации управляющая программа выбирает порядок обслуживания информации. При составлении управляющей программы требуется определить порядок приоритетов среди потоков, чтобы в результате работы машины было обработано наибольшее количество поступающей информации при одновременном сокращении длины очереди (уменьшении объема буферной памяти).

Определим параметр времени обслуживания для каждого потока

$$\nu_i = \frac{1}{t_i},$$

т. е. $\nu_1 = 5$; $\nu_2 = 2{,}5$; $\nu_3 = 4$; $\nu_4 = 10$; $\nu_5 = 20$; $\nu_6 = 6{,}7$.

Тогда величина $\rho_i = \lambda_i/\nu_i$ $(i = 1 \div 6)$ для каждого потока:

$$\rho_1 = 0{,}05; \quad \rho_2 = 0{,}06; \quad \rho_3 = 0{,}1;$$
$$\rho_4 = 0{,}2; \quad \rho_5 = 0{,}375; \quad \rho_6 = 0{,}195.$$

Допустимые вероятности потери требований каждого потока:

$$P_{\text{пот}}^{\lambda_1} = 8 \cdot 10^{-9}; \; P_{\text{пот}}^{\lambda_2} = 2 \cdot 10^{-8}; \quad P_{\text{пот}}^{\lambda_3} = 5 \cdot 10^{-8};$$
$$P_{\text{пот}}^{\lambda_4} = 3 \cdot 10^{-9}; \quad P_{\text{пот}}^{\lambda_5} = 5 \cdot 10^{-9}; \quad P_{\text{пот}}^{\lambda_6} = 1 \cdot 10^{-8},$$

а величина допустимой вероятности увеличения длины очереди

$$Q = 10^{-8}.$$

Допустимое время ожидания обслуживания требования каждого потока:

$$\tau_1 = 2{,}0; \quad \tau_2 = 1{,}6; \quad \tau_3 = 4;$$
$$\tau_4 = 1{,}25; \quad \tau_5 = 2{,}6; \quad \tau_6 = 1.$$

Рассчитаем длину очереди, вводя различные системы приоритетов для раздельных очередей по каждому потоку сообщений. Используем формулу (7.7) *

$$m_i = \frac{\ln \dfrac{P_{\text{пот}}^{\lambda_i} \prod\limits_{n=1}^{i-1} p_0^{\lambda_n}}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n} - \rho_i (1 - P_{\text{пот}}^{\lambda_i})}}{\ln \dfrac{\rho_i}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n}}},$$

где $P_{\text{пот}}^{\lambda_i}$ — допустимая вероятность потери требования потока, обладающего i-м приоритетом; $p_0^{\lambda_i}$ — вероятность того, что в системе отсутствуют требования данного потока,

$$p_0^{\lambda_i} = \frac{1 - \dfrac{\rho_i}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n}}}{1 - \left(\dfrac{\rho_i}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i+1}}.$$

Примем за систему приоритетов нумерацию потоков, т. е. наивысший приоритет имеет поток с параметром λ_1, следующий приоритет — поток с параметром λ_2 и т. д. Тогда, используя формулы (7.7) и (7.6), получаем значения необходимого числа мест ожидания для каждого потока. Результаты расчетов приведены в табл. 3.

Таблица 3

Поток с параметром λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
Необходимое число мест ожидания m_i	6	7	8	15	40	166

* Вывод формулы приводится в параграфе 3.

Общее число мест ожидания для всех потоков равно сумме полученных значений m_i для каждого потока:

$$m = \sum m_i = 242.$$

Пусть требуется получить минимальную длину очереди. Система приоритетов характеризуется коэффициентом K_1, причем наивысший приоритет присваивается потоку, имеющему максимальное значение K_1. Определим значения коэффициентов K_1 (табл. 4).

Таблица 4

Поток с параметром λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
Коэффициент K_1	2	1	1,6	4	8	2,67

Согласно введенной системе приоритетов потокам присваиваются приоритеты и рассчитывается необходимое число мест ожидания (табл. 5).

Таблица 5

Номер приоритета	1	2	3	4	5	6
Поток с параметром λ_i	λ_5	λ_4	λ_6	λ_1	λ_3	λ_2
Необходимое число мест ожидания m_i	18	17	23	13	28	57

Общее число мест ожидания для всех потоков при выбранной системе приоритетов

$$m = \sum m_i = 156.$$

Если требуется уменьшить вероятность потери за счет ожидания, то система приоритетов характеризуется коэффициентом K_2 и более высокий приоритет имеют требования, у которых коэффициент K_2 наибольший. Найдем значения коэффициентов K_2 для всех потоков (табл. 6).

Таблица 6

Поток с параметром λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
Коэффициент K_2	2	2,5	1	3,2	1,54	4

По табл. 6 задаем приоритеты и рассчитываем число мест ожидания (табл. 7).

Таблица 7

Номер приоритета	1	2	3	4	5	6
Поток с параметром λ_l	λ_6	λ_4	λ_2	λ_1	λ_5	λ_3
Необходимое число мест ожидания m_l	11	14	8	8	64	83

Общее число мест ожидания

$$m = \sum m_l = 188.$$

Для удовлетворения одновременно двум вышепоставленным условиям необходимо ввести систему приоритетов, которая характеризуется коэффициентом

$$K = K_1 + \frac{Q}{P_{\text{пот}}^{\lambda_l}} K_2. \qquad (7.1)$$

Поток, имеющий большую величину K, обладает более высоким приоритетом. Вычислим значения K для каждого потока (табл. 8)

Таблица 8

Поток с параметром λ_l	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
Коэффициент K	4,5	2,25	1,8	14,7	11,08	6,05

По величине коэффициента K распределим приоритеты и вычислим число мест ожидания (табл. 9).

Таблица 9

Номер приоритета	1	2	3	4	5	6
Поток с параметром λ_l	λ_4	λ_5	λ_6	λ_1	λ_2	λ_3
Необходимое число мест ожидания m_l	12	25	23	13	16	83

Общее число мест ожидания

$$m = \sum m_l = 172.$$

Если задавать приоритеты по меньшей величине допустимой вероятности потери, то потоки будут обслуживаться в порядке, указанном в табл. 10.

Таблица 10

Номер приоритета	1	2	3	4	5	6
Поток с параметром λ_i	λ_4	λ_5	λ_1	λ_6	λ_2	λ_3

Число мест ожидания для требований каждого потока приведено в табл. 11.

Таблица 11

Поток с параметром λ_i	λ_4	λ_5	λ_1	λ_6	λ_2	λ_3
Необходимое число мест ожидания m_i	12	25	19	28	16	83

Полное число мест ожидания

$$m = \sum m_i = 183.$$

Таким образом, были рассмотрены пять различных систем приоритетов и найдено необходимое число мест ожидания. Как видно из примера, система массового обслуживания, в которой критерием приоритета является величина коэффициента (7.1), имеет меньшее число мест ожидания, чем системы, у которых критериями приоритетов является либо порядковый номер потока, либо коэффициент K_2, либо допустимая вероятность потери требования. И только система, в которой за критерий приоритета принят коэффициент K_1, имеет меньшее число мест ожидания, но в такой системе вероятность потери требований больше, чем в системе с критерием K.

Следовательно, предлагаемая система приоритетов по величине K наилучшим образом удовлетворяет одновременно двум условиям: получение минимальной длины очереди и минимальной вероятности потери требований за счет времени ожидания. Применение критерия K для организации обработки сообщений в УВС позволит уменьшить объем буферной памяти и обработать большее число сообщений по сравнению с бесприоритетной системой. Это в свою очередь приводит к уменьшению веса и габаритных размеров вычис-

лительных систем, а также увеличивает количество информации на выходе системы управления, что приводит к более эффективному управлению.

2. ОБСЛУЖИВАНИЕ НЕСКОЛЬКИХ ИНФОРМАЦИОННЫХ ПОТОКОВ АЛГОРИТМОВ НА УВМ С АБСОЛЮТНЫМИ ПРИОРИТЕТАМИ

При проектировании УВС и систем связи возникает ряд задач по обслуживанию нескольких входящих информационных потоков с приоритетами. Анализ систем массового обслуживания при ограничениях на число мест ожидания, даже в предположении экспоненциального распределения времени обслуживания, наталкивается на значительные трудности [59].

Поставим задачу исследования УВС, обрабатывающей несколько потоков информации с приоритетами, как системы массового обслуживания. Для этого рассмотрим однолинейную систему массового обслуживания с ограниченным числом мест ожидания, абсолютными приоритетами и дообслуживанием прерванного требования.

Пусть система состоит из одного обслуживающего аппарата (вычислителя) $n = 1$, который одновременно может обслуживать только одно требование (решать одну задачу) и m мест ожидания (m ячеек памяти буферного запоминающего устройства). Если поступившее требование застает свободный аппарат или хотя бы одно свободное место ожидания, то оно остается в системе, в противном случае происходит потеря требования.

Предположим, что в систему массового обслуживания поступают r потоков требований (r типов потоков информации). Требования потоков L_i ($i = 1, 2, \ldots, r$) будем называть требованиями приоритетов i. Считаем, что требования потока L_i имеют более высокий приоритет по сравнению с требованиями потока L_j, если $i < j$. Требования высшего приоритета обладают преимуществом перед требованиями низшего приоритета. Требования одного приоритета обслуживаются в том порядке, в котором они поступают. Если во время обслуживания некоторого требования поступает требование более высокого приоритета, то можно представить случаи, когда обслуживание прерывается и сразу же начинается обслуживание поступившего требования более высокого приоритета или, когда такое прерывание не

происходит, т. е. случаи абсолютного или относительного приоритетов. Исследуем следующую схему обслуживания с преимуществом. Если во время обслуживания очередного требования поступает требование более высокого приоритета, то обслуживание прерывается и начинается обслуживание поступившего требования. Когда система освободится от требований более высокого приоритета, чем прерванное, последнее дообслуживается.

Допустим, что:

1) потоки L_1, L_2, ..., L_r независимы;

2) моменты поступления требований приоритета i ($i = 1, 2, \ldots, r$) образуют простейший поток с параметром λ_i;

3) длительности обслуживания требований (всех потоков) есть независимые случайные величины;

4) функция распределения времени обслуживания требования приоритета i имеет вид

$$F(t) = 1 - e^{-\nu_i t} .$$

За основные вероятностные характеристики обслуживания такой системы принимаются: время ожидания начала обслуживания для требования приоритета i; время пребывания в системе требования приоритета i; длина очереди для требований каждого приоритета, при условии, что для каждого потока L_i отведено m_i мест ожидания.

Условимся обозначать через $P_k^{\lambda_i}(t)$ вероятность того, что в момент t система находится в состоянии k, т. е. имеется всего k требований потока L_i. Выведем систему дифференциальных уравнений для $P_k^{\lambda_i}(t)$.

Воспользуемся формулой

$$P_k^{\lambda_i}(t + \Delta t) = \sum_{j=0}^{m} P_j^{\lambda_i}(t)\, P_{jk}^{\lambda_i}(\Delta t) \qquad (7.2)$$

и вычислим вероятности перехода системы из состояния j, когда имеется j требований, в состояние k за время Δt, т. е. $P_{jk}^{\lambda_i}(\Delta t)$ для каждого потока L_i.

Известно, что вероятность перехода рассматриваемой системы из состояния j в состояние k за время Δt для $|j - k| \geqslant 2$ есть бесконечно малая величина более высокого порядка, чем Δt, т. е.

$$P_{jk}^{\lambda_i}(\Delta t) = 0(\Delta t) \text{ при } |j - k| \geqslant 2.$$

Определим остальные переходные вероятности для каждого потока L_i с параметром λ_i и параметром времени обслуживания ν_i, учтя, что обслуживание ведется с преимуществом, т. е. поток L_i обладает i-м приоритетом. Обозначим вероятность того, что в системе обслуживания отсутствуют требования потока L_i через время Δt, при условии, что в момент t система обслуживания была свободна от требований потока L_i, через $P_{00}^{\lambda_i}(\Delta t)$. Так как вероятность поступления хотя бы одного требования потока L_i за время Δt равна $\lambda_i \Delta t + 0(\Delta t)$, то получаем

$$P_{00}^{\lambda_i}(\Delta t) = 1 - \lambda_i \Delta t - 0(\Delta t).$$

Вероятность того, что из имеющихся k требований потока L_i ни одно не покинет систему за время Δt и ни одно новое не поступит за это время, может быть найдена как произведение вероятностей следующих двух событий. Первое — ни одно требование потока L_i не поступит за время Δt, вероятность чего равна $1 - \lambda_i \Delta t - 0(\Delta t)$; второе — за время Δt не закончится обслуживание ни одного требования.

Тогда для потока L_1 с параметром λ_1 и параметром времени обслуживания ν_1 получаем значение переходной вероятности

$$P_{kk}^{\lambda_1}(\Delta t) = [1 - \lambda_1 \Delta t - 0(\Delta t)]\,[e^{-\nu_1 \Delta t}] =$$
$$= [1 - \lambda_1 \Delta t - 0(\Delta t)]\,[1 - \nu_1 \Delta t + 0(\Delta t)] =$$
$$= 1 - \lambda_1 \Delta t - \nu_1 \Delta t + 0(\Delta t).$$

Для потока L_2 вероятность $P_{kk}^{\lambda_2}(\Delta t)$ может быть найдена аналогично $P_{kk}^{\lambda_1}(\Delta t)$, как произведение вероятностей следующих событий. Первое — ни одно требование потока L_2 не поступит за время Δt в систему обслуживания, вероятность чего равна $1 - \lambda_2 \Delta t + 0(\Delta t)$; второе — за время Δt не закончится обслуживание ни одного требования. Это событие может наступить, если имеет место одно из трех следующих несовместных событий:

а) если в системе в момент времени t имеется хотя бы одно требование потока L_1, т. е. более высокого приоритета. Обозначим вероятность того, что в системе в момент времени отсутствуют требования потока L_1 через $P_0^{\lambda_1}(t)$. Вероятность того, что в системе имеется хотя бы одно требование

потока L_1 в момент времени t

$$P(a) = 1 - P_0^{\lambda_1}(t);$$

б) если в момент времени t отсутствуют требования потока L_1, но за время Δt поступит хотя бы одно требование этого потока. При поступлении требования потока L_1 обслуживающий аппарат прекращает обслуживание требования потока L_2 и приступает к обслуживанию требований потока L_1. Вероятность этого события найдем как произведение вероятностей составляющих событий

$$P(б) = [P_0^{\lambda_1}(t)]\,[1 - P_{00}^{\lambda_1}(\Delta t)];$$

в) если в момент времени t отсутствуют требования потока L_1, за время Δt в систему не поступит ни одного требования потока L_1, и обслуживающий аппарат приступил к обслуживанию требования потока L_2, но не закончил обслуживание этого требования за время Δt. Вероятность такого события по теореме умножения вероятностей

$$P(в) = [P_0^{\lambda_1}(t)]\,[P_{00}^{\lambda_1}(\Delta t)]\,e^{-\nu_2 \Delta t}.$$

Таким образом, вероятность того, что за время Δt не закончится обслуживание ни одного требования потока L_2, найдем как сумму вероятностей $P(a) + P(б) + P(в)$:

$$[1 - P_0^{\lambda_1}(t)] + [P_0^{\lambda_1}(t)]\,[1 - P_{00}^{\lambda_1}(\Delta t)] + [P_0^{\lambda_1}(t)] \times$$
$$\times\,[P_{00}^{\lambda_1}(\Delta t)]\,e^{-\nu_2 \Delta t} = 1 - P_0^{\lambda_1}(t) + P_0^{\lambda_1}(t) - P_0^{\lambda_1}(t)\,P_{00}^{\lambda_1}(\Delta t) +$$
$$+ P_0^{\lambda_1}(t)\,P_{00}^{\lambda_1}(\Delta t) - P_0^{\lambda_1}(t)\,P_{00}^{\lambda_1}(\Delta t)\,\nu_2\,\Delta t + 0(\Delta t) =$$
$$= 1 - P_0^{\lambda_1}(t)\,\nu_2 \Delta t + 0(\Delta t).$$

Вероятность того, что из имеющихся k требований потока L_2 ни одно не покинет систему за время Δt и ни одно не поступит за это время,

$$P_{kk}^{\lambda_2}(\Delta t) = [1 - \lambda_2 \Delta t - 0(\Delta t)]\,[1 - P_0^{\lambda_1}(t)\,\nu_2\,\Delta t +$$
$$+ 0(\Delta t)] = 1 - \lambda_2 \Delta t - P_0^{\lambda_1}(t)\,\nu_2 \Delta t + 0(\Delta t).$$

Рассуждая аналогично, найдем $P_{kk}^{\lambda_3}(\Delta t)$ для потока L_3, как произведение вероятностей следующих событий. Первое — ни одно требование потока L_3 за время Δt не поступит в систему обслуживания, вероятность чего равна $1 - \lambda_3 \Delta t - 0(\Delta t)$; второе — за время Δt не закончится обслуживание ни одного требования потока L_3.

Это событие имеет место в следующих случаях:

а) в системе в момент времени t имелось хотя бы одно требование потока, имеющего более высокий приоритет по сравнению с потоком L_3, т. е. требования потоков L_1 или L_2. Вероятность этого события найдем по формуле вероятности суммы двух событий (имеется хотя бы одно требование потока L_1 или потока L_2):

$$P'(a) = 1 - P_0^{\lambda_1}(t) + 1 - P_0^{\lambda_2}(t) - [1 - P_0^{\lambda_1}(t)] \times$$
$$\times [1 - P_0^{\lambda_2}(t)] = 1 - P_0^{\lambda_1}(t) + 1 - P_0^{\lambda_2}(t) - 1 +$$
$$+ P_0^{\lambda_2}(t) + P_0^{\lambda_1}(t) - P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) = 1 - P_0^{\lambda_1}(t) P_0^{\lambda_2}(t);$$

б) в системе в момент времени t отсутствуют требования потоков более высоких приоритетов, но за это время Δt поступит хотя бы одно требование потока с приоритетом выше приоритета потока L_3, т. е. либо требование потока L_1, либо потока L_2. Вероятность этого события

$$P'(б) = P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) \{1 - P_{00}^{\lambda_1}(\Delta t) + 1 - P_{00}^{\lambda_2}(\Delta t) -$$
$$- [1 - P_{00}^{\lambda_1}(\Delta t)][1 - P_{00}^{\lambda_2}(\Delta t)]\} =$$
$$= P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) [1 - P_{00}^{\lambda_1}(\Delta t) + 1 - P_{00}^{\lambda_2}(\Delta t) -$$
$$- 1 + P_{00}^{\lambda_1}(\Delta t) + P_{00}^{\lambda_2}(\Delta t) - P_{00}^{\lambda_1}(\Delta t) P_{00}^{\lambda_1}(\Delta t) =$$
$$= P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) \{1 - [1 - \lambda_1 \Delta t + 0(\Delta t)][1 - \lambda_2 \Delta t + 0(\Delta t)]\} =$$
$$= P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) (\lambda_1 + \lambda_2) \Delta t + 0(\Delta t);$$

в) в момент времени t в системе не было требований потоков более высокого приоритета, чем приоритет потока L_3, за время Δt не поступило ни одного требования этих потоков и система приступила к обслуживанию требования потока L_3, но за время Δt не закончила обслуживание этого требования. Вероятность такого события равна произведению вероятностей составляющих событий

$$P'(в) = P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) P_{00}^{\lambda_1}(\Delta t) P_{00}^{\lambda_2}(\Delta t) e^{-\nu_3 \Delta t} =$$
$$= P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) [1 - \lambda_1 \Delta t + 0(\Delta t)][1 - \lambda_2 \Delta t + 0(\Delta t)] e^{-\nu_3 \Delta t} =$$
$$= P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) - (\lambda_1 + \lambda_2) P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) \Delta t -$$
$$- P_0^{\lambda_2}(t) P_0^{\lambda_1}(t) \nu_3 \Delta t + 0(\Delta t).$$

Вероятность того, что ни одно требование потока L_3 не будет обслужено за время Δt, найдется как вероятность

суммы трех рассмотренных выше событий:

$$P'(a + б + в) = P'(a) + P'(б) + P'(в) =$$

$$= 1 - P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) + P_0^{\lambda_1}(t) P_0^{\lambda_2}(t)(\lambda_1 + \lambda_2)\Delta t +$$

$$+ 0(\Delta t) + P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) - (\lambda_1 + \lambda_2) P_0^{\lambda_1}(t) P_0^{\lambda_2}(t)\Delta t -$$

$$- P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) \nu_3 \Delta t + 0(\Delta t) = 1 - P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) \nu_3 \Delta t + 0(\Delta t).$$

Таким образом, вероятность того, что за время Δt в систему не поступит ни одного требования потока L_3, и ни одно требование этого же потока не будет обслужено за время Δt,

$$P_{kk}^{\lambda_3}(\Delta t) = [1 - \lambda_3 \Delta t - 0(\Delta t)] \times$$

$$\times [1 - P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) \nu_3 \Delta t + 0(\Delta t) =$$

$$= 1 - \lambda_3 \Delta t - P_0^{\lambda_1}(t) P_0^{\lambda_2}(t) \nu_3 \Delta t + 0(\Delta t).$$

Аналогично найдем выражение для вероятностей $P_{kk}^{\lambda_1}(\Delta t)$, $P_{kk}^{\lambda_2}(\Delta t)$ и т. д. Тогда для любого потока L_i ($i = 1, 2, \ldots, r$) можно записать выражение для определения вероятности $P_{kk}^{\lambda_i}(\Delta t)$ в следующем виде:

$$P_{kk}^{\lambda_i}(\Delta t) = 1 - \lambda_i \Delta t - \nu_i \Delta t \prod_{n=1}^{i-1} P_0^{\lambda_n}(t) + 0(\Delta t).$$

Вероятность перехода системы от k к $k + 1$ требованиям, находящимся в системе обслуживания через время Δt, для каждого потока L_i с приоритетом i может быть найдена как произведение вероятностей двух событий. Первое — за время Δt в систему поступит одно требование потока L_i, вероятность чего равна $\lambda_i \Delta t + 0(\Delta t)$; второе — за время Δt не закончится обслуживание ни одного требования, вероятность чего

$$1 - \nu_i \Delta t \prod_{n=1}^{i-1} P_0^{\lambda_n}(t) + 0(\Delta t).$$

Вероятность перехода системы от k к $k + 1$ требованиям, находящимся в системе обслуживания через время Δt для потока L_i,

$$P_{k,k+1}^{\lambda_i}(\Delta t) = [\lambda_i \Delta t + 0(\Delta t)] \left[1 - \nu_i \Delta t \prod_{n=1}^{i-1} P_0^{\lambda_n}(t) + 0(\Delta t)\right] =$$

$$= \lambda_i \Delta t + 0(\Delta t).$$

Вероятность перехода системы от k к $k-1$ требованиям, находящимся в системе обслуживания через время Δt, равна произведению вероятностей следующих двух событий. Первое — за время Δt не поступит ни одного требования потока L_i, второе — за время Δt будет обслужено только одно требование. Обозначим вероятность перехода системы от k к $k-1$ требованиям за время Δt для требований потока L_i через $P^{\lambda_i}_{k,\,k-1}(\Delta t)$. Для получения выражения $P^{\lambda_i}_{k,\,k-1}(\Delta t)$ в общем виде определим последовательно вероятности $P^{\lambda_1}_{k,\,k-1}(\Delta t)$, $P^{\lambda_2}_{k,\,k-1}(\Delta t)$, $P^{\lambda_3}_{k,\,k-1}(\Delta t)$ и т. д.

Для потока L_1 с параметром λ_1 и параметром времени обслуживания ν_1 имеем

$$P^{\lambda_1}_{k,k-1}(\Delta t) = [1 - \lambda_1 \Delta t + 0(\Delta t)]\,[1 - e^{-\nu_1 \Delta t}] =$$
$$= [1 - \lambda_1 \Delta t + 0(\Delta t)]\,[\nu_1 \Delta t + 0(\Delta t)] = \nu_1 \Delta t + 0(\Delta t).$$

Вероятность того, что из k требований потока L_2, находящихся в системе обслуживания в момент времени t, останется $k+1$ требований этого потока, равна произведению вероятностей следующих событий. Первое — за время Δt в систему не поступит ни одного требования потока L_2, вероятность чего

$$1 - \lambda_2 \Delta t - 0(\Delta t);$$

второе — за время Δt будет обслужено одно требование потока L_2, вероятность этого события равна произведению вероятностей следующих событий:

а) в системе в момент времени t отсутствуют требования потока L_1, вероятность чего равна $P_0^{\lambda_1}(t)$;

б) за время Δt в систему не поступит ни одного требования потока L_1, вероятность чего равна

$$1 - e\lambda_1 \Delta t - 0(\Delta t);$$

в) обслуживающий аппарат закончит за время Δt обслуживание одного требования потока L_2, вероятность чего равна

$$1 - e^{-\nu_2 \Delta t}.$$

Вероятность того, что за время Δt закончится обслуживание одного требования потока L_2 и система перейдет в состояние $k-1$ из состояния k определится как произведение вероятностей трех вышерассмотренных событий,

т. е.

$$P_0^{\lambda_1}(t)\,[1 - \lambda_1 \Delta t - 0(\Delta t)]\,[1 - e^{-\nu_2 \Delta t}] = \\ = P_0^{\lambda_1}(t)\,\nu_2 \Delta t + 0(\Delta t).$$

Вероятность перехода системы от k к $k+1$ требованиям за время Δt для потока L_2

$$P_{k,k-1}^{\lambda_2}(\Delta t) = [P_0^{\lambda_1}(t)\,\nu_2 \Delta t + 0(\Delta t)]\,[1 - \lambda_2 \Delta t - 0(\Delta t)] = \\ = P_0^{\lambda_1}(t)\,\nu_2 \Delta t + 0(\Delta t).$$

Аналогично найдем

$$P_{k,k-1}^{\lambda_3}(\Delta t) = P_0^{\lambda_1}(t)\,\nu_3 \Delta t + 0(\Delta t),$$

или для потока L_i $(i = 1, 2, \ldots, r)$ имеем

$$P_{k,k-1}^{\lambda_i}(\Delta t) = \nu_i \Delta t \prod_{n=1}^{i-1} P_0^{\lambda_n}(t) + 0(\Delta t).$$

Теперь, используя полученные значения переходных вероятностей

$$P_{00}^{\lambda_i}(\Delta t),\ P_{kk}^{\lambda_i}(\Delta t),\ P_{k,k+1}^{\lambda_i}(\Delta t),\ P_{k,k-1}^{\lambda_i}(\Delta t)$$

и подставляя их в выражение (7.2), записываем исходную систему уравнений для определения $P_k^{\lambda_i}(t + \Delta t)$ $(k = 0, 1, 2, \ldots, m_i)$ для потока L_i

$$P_0^{\lambda_i}(t + \Delta t) = P_0^{\lambda_i}(t)\,P_{00}^{\lambda_i}(\Delta t) + P_1^{\lambda_i}(t)\,P_{10}^{\lambda_i}(\Delta t) + 0(\Delta t).$$

При $0 < k < m_i$

$$P_k^{\lambda_i}(t + \Delta t) = P_{k-1}^{\lambda_i}(t)\,P_{k-1,k}^{\lambda_i}(\Delta t) + P_k^{\lambda_i}(t)\,P_{kk}^{\lambda_i}(\Delta t) + \\ + P_{k+1}^{\lambda_i}(t)\,P_{k+1,k}^{\lambda_i}(\Delta t) + 0(\Delta t).$$

При $k = m_i$

$$P_{m_i}^{\lambda_i}(t + \Delta t) = P_{m_i-1}^{\lambda_i}(t)\,P_{m_i-1,m_i}^{\lambda_i}(\Delta t) + P_{m_i}^{\lambda_i}(t)\,P_{m_i m_i}^{\lambda_i}(\Delta t) + \\ + 0(\Delta t).$$

После подстановки значений переходных вероятностей получим систему уравнений для определения вероятностей $P_k^{\lambda_i}(t + \Delta t)$ $(k = 0, 1, 2, \ldots, m_i)$. При подстановке необхо-

димо учесть, что

$$P^{\lambda_i}_{m_i m_i}(\Delta t) = 1 - \nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_n}_0(t) + 0(\Delta t).$$

Тогда

$$P^{\lambda_i}_0(t+\Delta t) = P^{\lambda_i}_0(t)[1 - \lambda_i \Delta t + 0(\Delta t)] +$$

$$+ P^{\lambda_i}_1(t)\left[\nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_n}_0(t) + 0(\Delta t)\right];$$

$$P^{\lambda_i}_k(t+\Delta t) = P^{\lambda_i}_{k-1}(t)[\lambda_i \Delta t + 0(\Delta t)] +$$

$$+ P^{\lambda_i}_k(t)\left[1 - \lambda_i \Delta t - \nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_n}_0(t) + 0(\Delta t)\right] +$$

$$+ P^{\lambda_i}_{k+1}(t)\left[\nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_n}_0(t) + 0(\Delta t)\right] \qquad (0 < k < m);$$

$$P^{\lambda_i}_{m_i}(t+\Delta t) = P^{\lambda_i}_{m_i-1}(t)[\lambda_i \Delta t + 0(\Delta t)] +$$

$$+ P^{\lambda_i}_{m_i}(t)\left[1 - \nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_n}_0(t) + 0(\Delta t)\right].$$

Преобразуем эту систему:

$$P^{\lambda_i}_0(t+\Delta t) = P^{\lambda_i}_0(t) - P^{\lambda_i}_0(t)\lambda_i \Delta t +$$

$$+ P^{\lambda_i}_1(t)\nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_n}_0(t) + 0(\Delta t);$$

$$P^{\lambda_i}_k(t+\Delta t) = P^{\lambda_i}_{k-1}(t)\lambda_i \Delta t + P^{\lambda_i}_k(t) - P^{\lambda_i}_k(t)\lambda_i \Delta t -$$

$$- P^{\lambda_i}_k(t)\Delta t \nu_i \prod_{n=1}^{i-1} P^{\lambda_0}_0(t) + P^{\lambda_i}_{k+1}(t)\nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_n}_0(t) + 0(\Delta t)$$

$$(0 < k < m_i);$$

$$P^{\lambda_i}_{m_i}(t+\Delta t) = P^{\lambda_i}_{m_i-1}(t)\lambda_i \Delta t +$$

$$+ P^{\lambda_i}_{m_i}(t) - P^{\lambda_i}_{m_i}(t)\nu_i \Delta t \prod_{n=1}^{i-1} P^{\lambda_0}_0(t) + 0(\Delta t).$$

Или

$$\frac{P_0^{\lambda_i}(t+\Delta t)-P_0^{\lambda_i}(t)}{\Delta t}=$$

$$=-\lambda_i P_0^{\lambda_i}(t)+\nu_i P_1^{\lambda_i}(t)\prod_{n=1}^{i-1}P_0^{\lambda_n}(t)+\frac{0(\Delta t)}{\Delta t};$$

$$\frac{P_k^{\lambda_i}(t+\Delta t)-P_k^{\lambda_i}(t)}{\Delta t}=\lambda_i P_{k-1}^{\lambda_i}(t)-$$

$$-P_k^{\lambda_i}(t)\left[\lambda_i+\nu_i\prod_{n=1}^{i-1}P_0^{\lambda_n}(t)\right]+$$

$$+\nu_i P_{k+1}^{\lambda_i}(t)\prod_{n=1}^{i-1}P_0^{\lambda_n}(t)+\frac{0(\Delta t)}{\Delta t}\qquad(0<k<m_i);$$

$$\frac{P_{m_i}^{\lambda_i}(t+\Delta t)-P_{m_i}^{\lambda_i}(t)}{\Delta t}=\lambda_i P_{m_i-1}^{\lambda_i}(t)-$$

$$-P_{m_i}^{\lambda_i}(t)\,\nu_i\prod_{n=1}^{i-1}P_0^{\lambda_n}(t)+\frac{0(\Delta t)}{\Delta t}.$$

Переходя к пределу при $\Delta t\to 0$, получаем однородную систему m_i+1 обыкновенных линейных дифференциальных уравнений для определения вероятностей состояний системы $P_k^{\lambda_i}(t)$ $(k=0,1,2,\ldots,m_i)$ для потока L_i $(i=1,2,\ldots,r)$:

$$\left.\begin{aligned}
&[P_0^{\lambda_i}(t)]'=-\lambda_i P_0^{\lambda_i}(t)+P_1^{\lambda_i}(t)\,\nu_i\prod_{n=1}^{i-1}P_0^{\lambda_n}(t);\\
&[P_k^{\lambda_i}(t)]'=\lambda_i P_{k-1}^{\lambda_i}(t)-\left[\lambda_i+\nu_i\prod_{n=1}^{i-1}P_0^{\lambda_n}(t)\right]P_k^{\lambda_i}(t)+\\
&+P_{k+1}^{\lambda_i}(t)\,\nu_i\prod_{n=1}^{i-1}P_0^{\lambda_n}(t)\qquad(0<k<m_i);\\
&[P_{m_i}^{\lambda_i}(t)]'=\lambda_i P_{m_i-1}^{\lambda_i}(t)-P_{m_i}^{\lambda_i}(t)\,\nu_i\prod_{n=1}^{i-1}P_0^{\lambda_n}(t).
\end{aligned}\right\}\qquad(7.3)$$

Таким образом, получена система дифференциальных уравнений для определения вероятностей состояний систе-

мы при одном обслуживающем аппарате и при обслуживании с преимуществом нескольких потоков информации. Причем данная система уравнений описывает состояния системы по каждому потоку требований L_l. Это позволяет использовать полученные уравнения для исследования УВС, когда для каждого потока сообщений L_l с приоритетом i имеются раздельные буферные накопители. Такой случай может возникнуть, например, если управляющая вычислительная машина получает информацию от нескольких датчиков, каждый из которых имеет свою буферную память для хранения информации, подлежащей обработке.

Нетрудно заметить, что, имея полученную систему уравнений, можно легко получить уравнения для определения вероятностей состояний системы массового обслуживания с ограниченным числом мест ожидания, на которую поступает поток требований, не обладающих приоритетами, т. е. обслуживание ведется без преимуществ. Такая система обслуживания является частным случаем системы с приоритетами. Действительно, если $i = 1$, получим хорошо известную систему уравнений для одного обслуживающего аппарата:

$$P_0'(t) = -\lambda P_0(t) + \nu P_1(t);$$

$$P_k'(t) = \lambda P_{k-1}(t) - (\lambda + \nu) P_k(t) + \nu P_{k+1}(t)$$

$$(0 < k < m_l);$$

$$P_{m_1}'(t) = \lambda P_{m_1-1}(t) - \nu P_{m_1}(t),$$

где $m_1 = m + 1$, так как всего в системе находится $m + 1$ требование.

Следовательно, получена система дифференциальных уравнений для определения вероятностей состояний системы обслуживания, одновременно описывающая систему обслуживания с преимуществом, а также систему обслуживания без приоритетов с ограниченным числом мест ожидания для одного обслуживающего аппарата. Кроме того, при подстановке в уравнения (7.3) значения $m = 0$ получаем систему уравнений, описывающую задачи обслуживания с потерями.

Система (7.3) состоит из конечного числа уравнений и может быть проинтегрирована известным методом. Но, как правило, в теории массового обслуживания изучают установившееся решение. Существование таких решений

устанавливается так называемыми эргодическими теоремами, которые доказывают, что для любого k $(0 \leqslant k \leqslant m)$ существует предел

$$\lim_{t \to \infty} P_k(t) = p_k,$$

не зависящий от начального состояния системы. Найдем это предельное решение. Перейдем к пределу при $t \to \infty$ в системе (7.3). Легко видеть, что

$$\lim_{t \to \infty} P_k'(t) = 0,$$

так как в противном случае $|P_k(t)| \to \infty$ при $t \to \infty$, а это невозможно по смыслу величины $P_k(t)$. Поэтому пределы левых частей системы (7.3) равны нулю. Переходя к пределу, получаем:

$$0 = -\lambda_i p_0^{\lambda_i} + p_1^{\lambda_i} \nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n};$$

$$0 = \lambda_i p_{k-1}^{\lambda_i} - \left[\lambda_i + \nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}\right] p_k^{\lambda_i} + p_{k+1}^{\lambda_i} \nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n};$$

$$0 = \lambda_i p_{m_i-1}^{\lambda_i} - p_{m_i}^{\lambda_i} \nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}.$$

Эта система алгебраических уравнений вместе с нормировочным условием $\sum_{k=0}^{m_i} p_k^{\lambda_i} = 1$ служит для однозначного определения искомых значений p_k.

Для решения полученной системы уравнений введем обозначения

$$z_k^{\lambda_i} = \lambda_i p_{k-1}^{\lambda_i} - p_k^{\lambda_i} \nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}.$$

Система уравнений с учетом обозначений принимает такой вид:

$$z_1^{\lambda_i} = 0;$$

$$z_k^{\lambda_i} - z_{k+1}^{\lambda_i} = 0 \qquad (0 < k < m_i);$$

$$z_{m_i}^{\lambda_i} = 0,$$

откуда видно, что при всех значениях $1 \leqslant k \leqslant m_i$

$$z_k^{\lambda_i} = 0,$$

и, следовательно,

$$\lambda_i p_{k-1}^{\lambda_i} = p_k^{\lambda_i} \nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}.$$

Из этого уравнения находим

$$p_k^{\lambda_i} = \frac{\lambda_i}{\nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}} p_{k-1}^{\lambda_i}.$$

Это соотношение позволяет легко получить выражения для $p_k^{\lambda_i}$ через $p_0^{\lambda_i}$

$$p_1^{\lambda_i} = \frac{\lambda_i}{\nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}} p_0^{\lambda_i};$$

$$p_2^{\lambda_i} = \frac{\lambda_i}{\nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}} p_1^{\lambda_i} = \frac{\lambda_i}{\nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}} \cdot \frac{\lambda_i}{\nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}} p_0^{\lambda_i} =$$

$$= \left(\frac{\lambda_i}{\nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^2 p_0^{\lambda_i} \text{ и т. д.,}$$

откуда видно, что

$$p_k^{\lambda_i} = \left(\frac{\lambda_i}{\nu_i \prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^k p_0^{\lambda_i} \qquad (1 \leqslant k \leqslant m_i),$$

или, обозначив через $\rho = \frac{\lambda_i}{\nu_i}$, получим

$$p_k^{\lambda_i} = \left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^k p_0^{\lambda_i} \qquad (1 \leqslant k \leqslant m_i). \qquad (7.4)$$

Вероятность отсутствия требований потока L_i в системе, т. е. вероятность $p_0^{\lambda_i}$, найдем из условия

$$\sum_{k=0}^{k=m_i} p_k^{\lambda_i} = 1.$$

Подставляя в это уравнение значение p_k из формулы (7.4), получаем

$$p_0^{\lambda_l} \sum_{k=0}^{m_l} \left(\frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}} \right)^k = 1,$$

или

$$p_0^{\lambda_l} = \frac{1}{\sum_{k=0}^{m_l} \left(\frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}} \right)^k}.$$

Сумма может быть найдена как сумма членов геометрической прогрессии со знаменателем $\frac{\rho_l}{\prod_{n=0}^{n=l-1} p_0^{\lambda_n}}$, т. е.

$$\sum_{k=0}^{m_l} \left(\frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}} \right)^k = \frac{1 - \left(\frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}} \right)^{m_l+1}}{1 - \frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}}}.$$

Тогда

$$p_0^{\lambda_l} = \frac{1 - \frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}}}{1 - \left(\frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}} \right)^{m_l+1}}, \qquad (7.5)$$

$$p_k^{\lambda_l} = \left(\frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}} \right)^k \cdot \frac{1 - \frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}}}{1 - \left(\frac{\rho_l}{\prod_{n=1}^{l-1} p_0^{\lambda_n}} \right)^{m_l+1}}. \qquad (7.6)$$

Таким образом, получены выражения, позволяющие определить вероятности состояний системы по каждому потоку требований. Применим их для расчета некоторых параметров УВС.

3. РАСЧЕТ ЕМКОСТИ БУФЕРНОЙ ПАМЯТИ УВС С МНОЖЕСТВОМ ВХОДЯЩИХ ПОТОКОВ АЛГОРИТМОВ

При проектировании цифровых УВС возникает задача определения технических характеристик или числа отдельных устройств, входящих в состав таких систем. Анализируя с помощью математических методов теории массового обслуживания или метода Монте-Карло, определяют среднюю длину очереди, среднее время ожидания или другие параметры, характеризующие задержки выполнения алгоритмов. Затем приступают к синтезу, т. е. к определению технических характеристик или числа устройств, исходя из минимально возможного значения какого-либо критерия [151].

Такая постановка задачи позволяет обосновать требования, предъявляемые к УВС, и спроектировать ее с наименьшими затратами. Одной из задач, возникающих при проектировании УВС, является определение объема буферной памяти.

Пусть УВС выполняет совокупность программ, различающихся приоритетами, в режиме прерываний. Заявки на выполнение программ могут образовать очередь, а прерванные программы затем довыполняются с места прерывания. Под программой понимается собственно программа и соответствующая ей исходная информация.

Рассмотрим систему, состоящую из одного обслуживающего аппарата, на который поступают r простейших потоков заявок. Примем, что в системе ожидают не более m заявок всех видов в случае общей очереди и не более m_i заявок каждого потока в случае раздельных очередей. В системах передачи и обработки информации под местом для ожидания подразумевается специальный блок памяти для каждого принятого сообщения или зона памяти, куда записывается принятое сообщение, так как для теории массового обслуживания не играет роли способ выделения мест для ожидания (структурный или программный).

В случае общей очереди все блоки памяти или зоны содержат одинаковое количество ячеек памяти, а в случае раздельных очередей одинаковыми должны быть блоки памяти или зоны, выделенные для одного типа заявок.

Использовав формулы (7.5) и (7.6), определим необходимое количество мест ожидания для системы массового обслуживания. Для этого требуется найти условия, при которых допустимо превышение длины очереди над объемом памяти, а также должны быть заданы параметр потока требований λ и быстродействие обслуживающего аппарата, которое характеризуется параметром ν. Очевидно, объем буферного запоминающего устройства должен быть такой, чтобы сохранить информацию с вероятностью не меньше $1 - P_{\text{пот}}$, где $P_{\text{пот}}$ — заданная допустимая вероятность потери требования. Для определения объема памяти необходимо найти такое число m, для которого вероятность того, что очередь будет больше m, была бы не больше $P_{\text{пот}}$, т. е. вероятность того, что очередная заявка получит отказ, равна вероятности того, что в системе ожидают m заявок, была бы меньше или равна $P_{\text{пот}}$, т. е. $p_m \leqslant P_{\text{пот}}$.

В зависимости от способа организации очередей (распределения поступающих заявок в буферной памяти) могут встретиться два случая.

Случай 1. Система обслуживания имеет раздельные очереди по каждому потоку требований L_i. Количество мест ожидания, отведенное для потока L_i, равно m_i, т. е. информация, поступающая на обработку в УВС, выдается группой датчиков, каждый из которых имеет свой объем буферной памяти.

Определим величину m_i, исходя из того, что вероятность потери требования потока L_i ($i = 1, 2, \ldots, r$) не больше допустимой вероятности потери требования данного потока, т. е.

$$p_{m_i}^{\lambda_i} \leqslant P_{\text{пот}}^{\lambda_i}.$$

Предельное значение m_i найдем из соотношения

$$P_{\text{пот}}^{\lambda_i} = \left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i} p_0^{\lambda_i}.$$

Подставим значение $p_0^{\lambda_i}$,

$$P_{\text{пот}}^{\lambda_i} = \left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i} \cdot \frac{1 - \frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}}}{1 - \left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i+1}}.$$

Разрешим это уравнение относительно m_i:

$$\frac{P_{\text{пот}}^{\lambda_i}}{1 - \frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}}} = \left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i} + \frac{P_{\text{пот}}^{\lambda_i}}{1 - \frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}}} \left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i+1},$$

или

$$\left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i} = \frac{\frac{P_{\text{пот}}^{\lambda_i}}{1 - \frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}}}}{1 + \frac{P_{\text{пот}}^{\lambda_i} \rho_i}{\left(1 - \frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right) \prod_{n=1}^{i-1} p_0^{\lambda_n}}}.$$

Преобразуем последнее выражение

$$\left(\frac{\rho_i}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{m_i} = \frac{P_{\text{пот}}^{\lambda_i} \prod_{n=1}^{i-1} p_0^{\lambda_n}}{\prod_{n=1}^{i-1} p_0^{\lambda_n} - \rho_i (1 - P_{\text{пот}}^{\lambda_i})}.$$

Прологарифмируем полученное выражение и получим

$$m_i = \frac{\ln \dfrac{P_{\text{пот}}^{\lambda_i} \prod\limits_{n=1}^{i-1} p_0^{\lambda_n}}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n} - \rho_i (1 - P_{\text{пот}}^{\lambda_i})}}{\ln \dfrac{\rho_i}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n}}}. \qquad (7.7)$$

При расчете следует учитывать, что поток L_1 обладает наивысшим абсолютным приоритетом по отношению ко всем остальным. Поэтому обслуживающий аппарат можно считать одним местом ожидания для потока L_1. Тогда формула для расчета величины m_1 принимает следующий вид:

$$m_1 = \frac{\ln \dfrac{P_{\text{пот}}^{\lambda_1}}{1 - \rho_1 (1 - P_{\text{пот}}^{\lambda_1})}}{\ln \rho_1} - 1. \qquad (7.8)$$

Общую емкость m буферной памяти при обслуживании с раздельными очередями для r потоков найдем как сумму m_i

$$m = \frac{\ln \dfrac{P_{\text{пот}}^{\lambda_i} \prod\limits_{n=1}^{i-1} p_0^{\lambda_n}}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n} - \rho_i (1 - P_{\text{пот}}^{\lambda_i})}}{\ln \dfrac{\rho}{\prod\limits_{n=1}^{i-1} p_0^{\lambda_n}}} - 1. \qquad (7.9)$$

В выражения (7.7) — (7.9) подставляют заданные значения допустимых вероятностей потери требования для потоков L_i. Количество мест ожидания для каждого потока должно быть больше m_i, чтобы выполнялось неравенство

$$p_{m_i} \leqslant P_{\text{пот}}^{\lambda_i}.$$

Случай 2. Система массового обслуживания имеет общую очередь для потоков всех приоритетов, причем количество мест ожидания равно m. Заявки поступают в систему массового обслуживания и становятся в очередь в зависи-

мости от приоритета, т. е. поступившее требование располагается впереди всех требований, ранее поступивших, но обладающих более низким приоритетом. Тогда, если требования всех потоков обладают одинаковой допустимой вероятностью потери $P_{\text{пот}}$, определить необходимое количество мест ожидания (объем буферного накопителя) можно, исходя из вероятности потери заявки потока с наименьшим приоритетом L_r, которую определим по формуле

$$p_{m+1} = P_{\text{пот}} = \sum_{k_1+k_2+\cdots+k_r=m+1} p_{k_1}^{\lambda_1} p_{k_2}^{\lambda_2} \cdots p_{k_r}^{\lambda_r} =$$

$$= \sum_{\Sigma k_i = m+1} \prod_{i=1}^{r} p_{k_i}^{\lambda_i} \qquad (i = 1, 2, \ldots, r). \qquad (7.10)$$

Сравним два вышеуказанных способа организации очередей с точки зрения необходимого количества мест ожидания при заданной допустимой вероятности потери требований $P_{\text{пот}}^{\lambda_i}$, если известны параметры входящих потоков λ_i и время обслуживания ν_i. Расчеты проведем для двух независимых простейших потоков требований с параметрами λ_1 и λ_2, учитывая, что поток L_1 имеет абсолютный приоритет перед потоком L_2.

Для упрощения сравнения двух способов организации очередей предполагаем, что допустимая вероятность потери требования для любого потока равна $P_{\text{пот}}$, т. е. имеет одно и то же значение для потока L_1 и потока L_2.

Рассчитаем емкость накопителя для двух потоков, поступающих в систему массового обслуживания с раздельными очередями. Используем для этого формулу [24]:

$$m_1 = \frac{\ln \dfrac{P_{\text{пот}}}{1 - \rho_1 (1 - P_{\text{пот}})}}{\ln \rho_1} - 1; \qquad (7.11)$$

$$m_2 = \frac{\ln \dfrac{P_{\text{пот}} p_0^{\lambda_1}}{p_0^{\lambda_1} - \rho_2 (1 - P_{\text{пот}})}}{\ln \dfrac{\rho_2}{p_0^{\lambda_1}}}, \qquad (7.12)$$

где

$$p_0^{\lambda_1} = \frac{1 - \rho_1}{1 - \rho_1^{m_1+1}}. \qquad (7.13)$$

Полная емкость накопителя

$$m = m_1 + m_2, \qquad (7.14)$$

где m_1 и m_2 — расчетные значения, округленные в большую сторону до целого числа.

В случае общей очереди для двух потоков воспользуемся формулой (7.10)

$$p_{m+1} = P_{\text{пот}} = \sum_{k_1+k_2=m+1} p_{k_1}^{\lambda_1} p_{k_2}^{\lambda_2}.$$

Подставляя значения $p_{k_1}^{\lambda_1}$ и $p_{k_2}^{\lambda_2}$ из формулы (7.4) и изменяя пределы суммирования, получаем

$$p_{m+1} = \sum_{k_1=0}^{m+1} \rho_1^{k_1} p_0^{\lambda_1} \left(\frac{\rho_2}{p_0^{\lambda_1}}\right)^{m+1-k_1} \frac{1 - \frac{\rho_2}{p_0^{\lambda_1}}}{1 - \left(\frac{\rho_2}{p_0^{\lambda_1}}\right)^{m+2-k_1}}, \qquad (7.15)$$

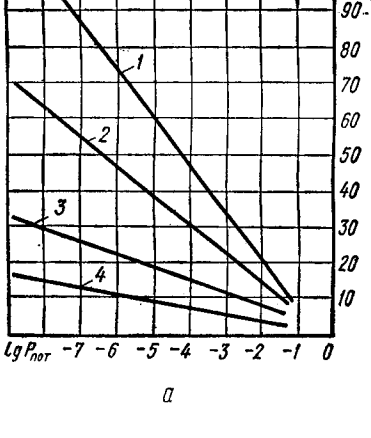
Таблица 12

Коэффициенты		Количество мест ожидания	Вероятность потери требования								
ρ_1	ρ_2		10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
0,5	0,4	m_1	2	5	8	12	15	18	22	25	28
		m_2	4	13	24	35	45	55	65	76	96
		m	6	18	32	47	60	73	87	101	114
0,4	0,4	m_1	2	4	6	9	11	14	16	19	22
		m_2	3	9	15	20	26	32	38	43	49
		m	5	13	21	29	37	46	54	62	71
0,2	0,3	m_1	1	2	4	5	7	8	9	11	12
		m_2	2	5	7	9	12	14	16	19	21
		m	3	7	11	14	19	22	25	30	33
0,05	0,1	m_1	0	1	2	3	3	4	5	6	6
		m_2	1	2	4	5	6	7	8	9	10
		m	1	3	6	8	9	11	13	15	16

где

$$p_0^{\lambda_1} = \frac{1-\rho_1}{1-\rho_1^{m+1}}.$$

Решение полученного уравнения относительно m представляет некоторые трудности, и поэтому ограничимся вычислением величины p_{m+1} как функции от m ($m = 1, 2, \ldots$). По зависимостям $p_{m+1} = f(m)$ определим необходимое количество мест ожидания m, обеспечивающее вероятность потери меньше $P_{пот}$ при следующих значениях ρ_1 и ρ_2:



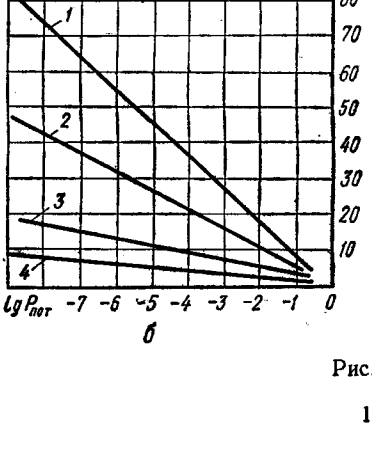
а

Кривая *1* $\rho_1 = 0{,}5$; $\rho_2 = 0{,}4$;

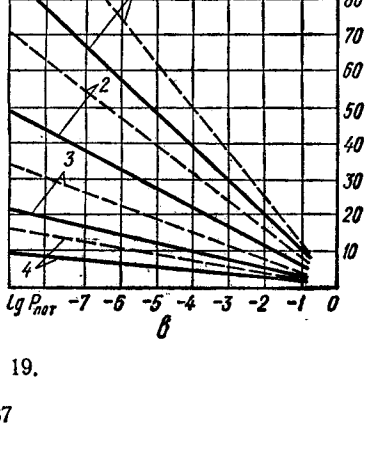
Кривая *2* $\rho_1 = 0{,}4$; $\rho_2 = 0{,}4$;

Кривая *3* $\rho_1 = 0{,}2$; $\rho_2 = 0{,}3$;

Кривая *4* $\rho_1 = 0{,}05$; $\rho_2 = 0{,}1$.



б



в

Рис. 19.

1. Для раздельных очередей. Задаваясь значениями $P_{\text{пот}} = 10^{-1} \div 10^{-9}$, вычисляем по формулам (7.11) — — (7.14) значения m, m_1 и m_2. Полученные результаты сводим в табл. 12 и строим график зависимостей $m = f(\lg P_{\text{пот}})$ для заданных значений ρ_1 и ρ_2 (рис. 19, *а*).

2. Для общей очереди. Задаваясь значениями $m = 1, 2, \ldots$, вычисляем вероятность потери требования p_{m+1} по формуле (7.15), которая в предельном случае равна допустимой вероятности потери, и строим график зависимостей (рис. 19, *б*).

Сравнивая результаты (рис. 19, *в*), полученные для раздельных очередей (пунктирные линии) и для общей очереди (сплошные линии), видим, что для обеспечения вероятности потери требований меньше допустимой при заданных значениях ρ_1 и ρ_2, в случае общей очереди потребуется меньшее количество мест ожидания (число ячеек буферного накопителя), чем в случае раздельных очередей.

Так, например, если $P_{\text{пот}} = 10^{-8}$ и $\rho_1 = 0{,}5$; $\rho_2 = 0{,}4$, то по рис. 19, *в* определяем для раздельных очередей $m = 101$, а для общей очереди $m = 75$.

Таким образом, анализ результатов показывает, что с точки зрения получения меньшего объема буферной памяти целесообразно реализовать случай общей очереди. Эти результаты можно использовать не только для расчета емкости буферной памяти, но также и для определения других параметров систем управления, в которых используются вычислительные машины. Например, можно рассчитать необходимое быстродействие вычислителя, если заданы объем буферной памяти m, вероятность потери требования $P_{\text{пот}}$ и значения λ_1 и λ_2.

4. ЗАКОН РАСПРЕДЕЛЕНИЯ ВРЕМЕНИ ОЖИДАНИЯ ОБСЛУЖИВАНИЯ АЛГОРИТМОВ В ПРИОРИТЕТНЫХ СИСТЕМАХ

Одной из важнейших характеристик УВС, которые ведут обработку сообщений нескольких видов, является закон распределения времени ожидания обслуживания. Действительно, некоторые сообщения представляют ценность только в течение определенных отрезков времени. Если в течение этого промежутка времени такое сообщение не было обработано, то его можно отбросить, так как его содержание обесценилось. Все поступившие в УВМ

сообщения, обработка которых задержалась больше, чем на определенный промежуток времени, теряют цену.

Определим скорость работы УВМ при обработке поступающей информации, чтобы сообщения не обесценивались за счет длительного времени ожидания. Для этого необходимо выявить соотношение между временем ожидания обслуживания и скоростью работы УВМ, т. е. закон распределения времени ожидания.

Зная вероятность состояния системы $p_k^{\lambda_i}$ по каждому потоку сообщений, найдем закон распределения времени ожидания обслуживания. Для упрощения выкладок, будем предполагать, что все потоки, поступающие в систему, имеют одинаковый экспоненциальный закон распределения времени обслуживания одного сообщения с параметром ν.

Определим закон распределения длительности ожидания окончания обслуживания требования, поступившего в какой-то момент времени и обладающего i-м приоритетом. Время ожидания обслуживания является в общем случае случайной величиной, которая зависит от того, какое количество требований потоков более высоких приоритетов и требований данного приоритета находилось в системе в момент поступления данного требования, сколько требований потоков более высоких приоритетов поступит в систему за время ожидания обслуживания рассматриваемого требования и как скоро закончится обслуживание всех этих требований.

Если время ожидания обслуживания (β) — случайная величина, то необходимо найти $P^{\lambda_i}\{\beta > t\}$, т. е. вероятность того, что время ожидания обслуживания для поступившего в произвольно выбранный момент времени требования потока L_i будет больше чем t. Обозначим через $P_k^{\lambda_i}\{\beta > t\}$ условную вероятность того же неравенства в предположении, что в момент поступления требования в системе находилось k требований, которые будут обслужены раньше поступившего.

Для потока L_1, обладающего наивысшим абсолютным приоритетом, при условии, что требования одного потока обслуживаются в порядке их поступления в систему, определим величину $P^{\lambda_1}\{\beta > t\}$ по формуле полной вероятности

$$P^{\lambda_1}\{\beta > t\} = \sum_{k=0}^{m} P_k^{\lambda_1}\{\beta > t\}\, p_k^{\lambda_1}. \qquad (7.16)$$

Суммирование производится до m, так как если в момент поступления очередного требования в системе находится $m+1$ требование потока L_1, то она отказывает в обслуживании всем последующим требованиям, т. е.

$$P_k^{\lambda_1}\{\beta > t\} = 0 \qquad (k = m+1,\ m+2,\ \ldots).$$

Величины $p_k^{\lambda_1}$ определяются по формулам (7.5), (7.6). Найдем значения $P_k^{\lambda_1}\{\beta > t\}$ при всех возможных k.

Пусть в момент поступления очередного требования потока L_1 в системе находится k требований этого же потока. При этом одно из них обслуживается, а $k-1$ ожидает своей очереди. Так как обслуживание требований одинакового приоритета производится в порядке поступления в систему, то вновь поступившее требование будет обслужено по окончании обслуживания $k+1$-го требования. Если этого не произойдет за время t, то время ожидания обслуживания (β) превысит величину t. Это равносильно тому, что за время t наступит одно из следующих событий: или не будет закончено обслуживание ни одного требования потока L_1, или будет закончено обслуживание только одного требования, или будет закончено обслуживание двух требований и т. д. или, наконец, будет обслужено k требований потока L_1. Во всех этих случаях время ожидания обслуживания того требования, которое поступило в систему, когда в ней уже находилось k требований потока L_1, превзойдет t. Вероятность этих событий обозначим соответственно:

$$q_0^{\lambda_1}(t),\ q_1^{\lambda_1}(t),\ q_2^{\lambda_1}(t),\ \ldots,\ q_k^{\lambda_1}(t).$$

Все эти события несовместные и в то же время исчерпывают все возможности, при которых время ожидания превысит t. Поэтому, применив теорему сложения вероятностей, найдем условную вероятность

$$P_k^{\lambda_1}\{\beta > t\} = \sum_{s=0}^{k} q_s^{\lambda_1}(t) \qquad (m \geqslant k \geqslant 0).$$

В связи с тем, что время обслуживания подчинено показательному закону и не зависит от того, сколько требований находится в данный момент в очереди, то вероятность того, что за время t не освободится обслуживающий аппарат,

$$q_0(t) = e^{-\nu t}.$$

Поток обслуживаемых требований является простейшим. Действительно, он обладает всеми тремя свойствами, характеризующими простейший поток: стационарностью, отсутствием последействия и ординарностью. Стационарность его вытекает из свойства показательного закона распределения времени обслуживания. Отсутствие последействия в потоке очевидно, так как число ранее обслуженных требований не влияет на последующий ход обслуживания. Что же касается ординарности потока, то она может быть легко установлена следующим образом. Вероятность того, что будет закончено обслуживание двух требований за время Δt,

$$(1 - e^{-\nu \Delta t})^2 = [\nu \Delta t + 0(\Delta t)]^2 = 0(\Delta t),$$

т. е. бесконечно малая величина по сравнению с Δt.

Параметр этого потока обслуженных требований равен ν, поэтому вероятность того, что за время t будет обслужено точно s требований потока L_1, можно вычислить по известной формуле, поставив вместо λ величину ν.

Вероятность появления точно s требований в выходящем потоке

$$q_s^{\lambda_1}(t) = \frac{(\nu t)^s}{s!} e^{-\nu t}.$$

Тогда условная вероятность

$$P_k^{\lambda_1}\{\beta > t\} = \sum_{s=0}^{k} q_s^{\lambda_1}(t) = \sum_{s=0}^{k} \frac{(\nu t)^s}{s!} e^{-\nu t}.$$

Подставляя значение $P_k^{\lambda_1}\{\beta > t\}$ в выражение (7.16), получаем

$$P^{\lambda_1}\{\beta > t\} = \sum_{k=0}^{m} p_k^{\lambda_1} \sum_{s=0}^{k} \frac{(\nu t)^s}{s!} e^{-\nu t}.$$

Заменим значение $p_k^{\lambda_1}$ из формулы (7.6)

$$P^{\lambda_1}\{\beta > t\} = \sum_{k=0}^{m} (\rho_1)^k p_0^{\lambda_1} \sum_{s=0}^{k} \frac{(\nu t)^s}{s!} e^{-\nu t} =$$

$$= p_0^{\lambda_1} e^{-\nu t} \sum_{k=0}^{m} (\rho_1)^k p_0^{\lambda_1} \sum_{s=0}^{k} \frac{(\nu t)^s}{s!}.$$

Определим вероятность $P^{\lambda_2}\{\beta > t\}$ для требований потока L_2. Время ожидания обслуживания требования потока L_2, поступившего в произвольный момент времени, зависит от того, какое количество требований потоков L_1 и L_2 находилось в системе в момент поступления очередного требования потока L_2, сколько требований потока L_1 поступит за время до окончания обслуживания поступившего требования потока L_2 и какое время будет затрачено на обслуживание этих требований. Исходя из этого, по формуле полной вероятности находим

$$P^{\lambda_2}\{\beta > t\} = \sum_{k=0}^{m-k_1'} p_k p_{k_1'}^{\lambda_1} P_{k}^{\lambda_2}\{\beta > t\},$$

где p_k — вероятность того, что в момент поступления очередного требования потока L_2 в системе находилось k требований потоков L_1 и L_2, причем из этих k требований было k_1 требований потока L_1 и k_2 требований потока L_2, т. е. $k = k_1 + k_2$.

Величину p_k определим как произведение вероятностей

$$p_k = p_{k_1}^{\lambda_1} p_{k_2}^{\lambda_2} \qquad (k_1 + k_2 = k),$$

где $p_{k_1}^{\lambda_1}$ — вероятность того, что в систему за время t поступит k_1' требований потока L_1. Так как поток L_1 простейший, то вероятность поступления точно k_1' требований за время t

$$P_{k_1}^{\lambda_1 t} = \frac{(\lambda_1 t)^{k_1'}}{k_1'!} e^{-\lambda_1 t}.$$

Величина k_1' является случайной величиной и ее математическое ожидание

$$M_t(k_1') = \lambda_1 t.$$

Время ожидания обслуживания β требования потока L_2 превзойдет величину t в том случае, если за время t будет обслужено меньше $k + k_1' + 1$ требований. Условную вероятность этого события определим как сумму вероятностей составляющих, т. е.

$$P_k^{\lambda_2}\{\beta > t\} = \sum_{s=0}^{k+k_1'} q_s(t) \qquad (k = k_1 + k_2).$$

Подставляя значения $q_s(t)$, получаем

$$P_k^{\lambda_3}\{\beta > t\} = \sum_{s=0}^{k+k_1'} \frac{(\nu t)^s}{s!} e^{-\nu t}.$$

Полученное значение $P_k^{\lambda_3}\{\beta > t\}$ подставим в выражение для определения $P_k^{\lambda_3}\{\beta > t\}$

$$P^{\lambda_2}\{\beta > t\} = p_{k_1'}^{\lambda_1} \sum_{k=0}^{m-k_1'} p_{k_1}^{\lambda_1} p_{k_2}^{\lambda_2} \sum_{s=0}^{k+k_1'} \frac{(\nu t)^s}{s!} e^{-\nu t}.$$

Подставим сюда значения $p_{k_1}^{\lambda_1}$ и $p_{k_2}^{\lambda_2}$:

$$P^{\lambda_2}\{\beta > t\} = p_{k_1'}^{\lambda_1} e^{-\nu t} \sum_{k=0}^{m-k_1'} \rho_1^{k_1} p_0^{\lambda_1} \left(\frac{\rho_2}{p_0^{\lambda_1}}\right)^{k_2} p_0^{\lambda_2} \sum_{s=0}^{k+k_1'} \frac{(\nu t)^s}{s!} =$$

$$= p_{k_1'}^{\lambda_1} p_0^{\lambda_1} p_0^{\lambda_2} e^{-\nu t} \sum_{k=0}^{m-k_1'} \rho_1^{k_1} \left(\frac{\rho_2}{p_0^{\lambda_1}}\right)^{k_2} \sum_{s=0}^{k+k_1'} \frac{(\nu t)^s}{s!}.$$

Аналогично определим вероятность $P^{\lambda_3}\{\beta > t\}$ для требований потока L_3. Очевидно, что время ожидания обслуживания требования потока L_3, поступившего в произвольно выбранный момент времени, зависит от количества требований потоков L_1, L_2 и L_3, находившихся в системе в момент поступления очередного требования потока L_3, от числа требований потоков L_1 и L_2, которые поступят за время до окончания обслуживания поступившего требования потока L_3, и от времени, которое будет затрачено на обслуживание этих требований. Поэтому

$$P^{\lambda_3}\{\beta > t\} = \sum_{k=0}^{m-k_1'-k_2'} p_k p_{k_1'}^{\lambda_1} p_{k_2'}^{\lambda_2} p_k^{\lambda_3}\{\beta > t\},$$

где $p_k = p_{k_1}^{\lambda_1} p_{k_2}^{\lambda_2} p_{k_3}^{\lambda_3}$ $\quad (k_1 + k_2 + k_3 = k)$;

вероятность того, что в систему за время t поступит k_1' требований потока L_1,

$$p_{k_1'}^{\lambda_1} = \frac{(\lambda_1 t)^{k_1'}}{k_1'!} e^{-\lambda_1 t};$$

вероятность того, что в систему за время t поступит k_2' требований потока L_2,

$$p_{k_2'}^{\lambda_2} = \frac{(\lambda_2 t)^{k_2'}}{k_2'!} e^{-\lambda_2 t};$$

k_1' и k_2' — число требований соответственно потоков L_1 и L_2, поступивших в систему за время t. Их математические ожидания

$$M_t(k_1') = \lambda_1 t; \quad M_t(k_2') = \lambda_2 t.$$

Условную вероятность $P_k^{\lambda_3}\{\beta > t\}$ найдем по формуле

$$P_k^{\lambda_3}\{\beta > t\} = \sum_{s=0}^{k+k_1'+k_2'} q_s(t) \qquad (k = k_1 + k_2 + k_3).$$

Подставляя значения $q_s(t)$, получаем

$$P_k^{\lambda_3}\{\beta > t\} = \sum_{s=0}^{k+k_1'+k_2'} \frac{(\nu t)^s}{s!} e^{-\nu t}.$$

Тогда

$$P^{\lambda_3}\{\beta > t\} = p_{k_1'}^{\lambda_1} p_{k_2'}^{\lambda_2} \sum_{k=0}^{m-k_1'-k_2'} p_{k_1}^{\lambda_1} p_{k_2}^{\lambda_2} p_{k_3}^{\lambda_3} \sum_{s=0}^{k+k_1'+k_2'} \frac{(\nu t)^s}{s!} e^{-\nu t} =$$

$$= p_{k_1'}^{\lambda_1} p_{k_2'}^{\lambda_2} e^{-\nu t} \sum_{k=0}^{m-k_1'-k_2'} \rho_1^{k} p_0^{\lambda_1} \left(\frac{\rho_2}{p_0^{\lambda_1}}\right)^{k_2} p_0^{\lambda_3} \times$$

$$\times \left(\frac{\rho_3}{p_0^{\lambda_1} p_0^{\lambda_2}}\right)^{k_3} p_0^{\lambda_3} \sum_{s=0}^{k+k_1'+k_2'} \frac{(\nu t)^s}{s!} =$$

$$= p_{k_1'}^{\lambda_1} p_{k_2'}^{\lambda_2} p_0^{\lambda_1} p_0^{\lambda_2} p_0^{\lambda_3} e^{-\nu t} \sum_{k=0}^{m-k_1'-k_2'} \rho_1^{k_1} \left(\frac{\rho_2}{p_0^{\lambda_1}}\right)^{k_2} \times$$

$$\times \left(\frac{\rho_3}{p_0^{\lambda_1} p_0^{\lambda_2}}\right)^{k_3} \sum_{s=0}^{k+k_1+k_2} \frac{(\nu t)^s}{s!} \qquad (k = k_1 + k_2 + k_3).$$

Аналогично можно определить значения $P^{\lambda_4}\{\beta > t\}$, $P^{\lambda_5}\{\beta > t\}$ и т. д. Тогда в общем виде для требований потока L_i вероятность того, что время ожидания обслуживания β превзойдет величину t, можно записать в виде:

$$P^{\lambda_i}\{\beta > t\} = e^{-\nu t} \prod_{n=1}^{i} p_{k'_{n-1}}^{\lambda_n - 1} p_0^{\lambda_n} \times$$

$$\times \sum_{k=0}^{m - \sum_{n=1}^{i-1} k'_n} \prod_{n=1}^{i} \left(\frac{\rho_n}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{k_n} \cdot \sum_{s=0}^{\sum_{n=1}^{i} k_n + \sum_{n=1}^{i-1} k'_n} \frac{(\nu t)^s}{s!},$$

или

$$P^{\lambda_i}\{\beta > t\} = e^{-\nu t} \prod_{n=1}^{i} p_{k'_{n-1}}^{\lambda_n - 1} p_0^{\lambda_n} \sum_{k=0}^{M} \prod_{n=1}^{i} \left(\frac{\rho_n}{\prod_{n=1}^{i-1} p_0^{\lambda_n}} \right)^{k_n} \times$$

$$\times \sum_{s=0}^{k+k'} \frac{(\nu t)^s}{s!}, \qquad (7.17)$$

где

$$M = m - \sum_{n=1}^{i-1} k'_n; \quad k = \sum_{n=1}^{i} k_n; \quad k' = \sum_{n=1}^{i-1} k'_n;$$

$$p_{k'_n}^{\lambda_n} = \frac{(\lambda_n t)^{k'_n}}{k'_n!} e^{-\lambda_n t};$$

$$M_t(k'_n) = \lambda_n t.$$

Таким образом, закон распределения времени ожидания обслуживания требованиями потока L_i определяется скоростью обслуживания и интенсивностями входящих потоков, обладающих более высокими приоритетами, чем поток L_i.

Все результаты расчетов получены в предположении, что закон распределения времени обслуживания алгоритмов экспоненциальный. Однако такое предположение не всегда является справедливым. В частности, при обработке алгоритмов управления и контроля очень часто используется не экспоненциальный закон распределения времени обслуживания, а время обработки каждого алгоритма характеризуется некоторой постоянной величиной.

Тогда для исследования подобных систем необходимо иметь результаты, аналогичные результатам, полученным в гл. VII, заменив экспоненциальный закон распределения времени обслуживания каждого класса алгоритмов управления на некоторую постоянную величину. Однако получение таких результатов связано со значительными трудностями. Моделирование управляющих систем с приоритетами на ЭЦВМ, проведенное в гл. X, показало, что замена реального распределения времени обслуживания на экспоненциальный закон распределения вносит незначительные ошибки, и таким образом результаты гл. VII могут быть использованы для исследования систем управления и контроля.

Глава VIII
ПОСТРОЕНИЕ УВС НА ПОСТОЯННОЙ ПАМЯТИ

1. ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ НА УВМ С ПЗУ

Управляющие вычислительные системы, которые были рассмотрены в предыдущих главах, интерпретировались как системы массового обслуживания, у которых роль обслуживающего аппарата выполнял сумматор ЭЦВМ. С целью повышения эффективности обслуживания в системах массового обслуживания необходимо повышать скорость обработки заявок обслуживающим аппаратом. Этого можно достигнуть, если заранее иметь все возможные результаты обработки заявок, и тогда процесс обслуживания сведется только к выбору необходимого результата. Например, в кассе железнодорожного вокзала, продающей билеты до определенных станций, можно иметь набор билетов до всех станций. Тогда процесс обслуживания каждого пассажира сведется к выбору необходимого билета из этого набора, вместо выписывания каждого билета. В этом случае эффективность обслуживания, очевидно, увеличится.

Указанный принцип построения системы массового обслуживания может быть распространен на построение УВС. Большинство УВС содержит три основные устройства: запоминающее, арифметическое и управления. Все арифметические операции в таких машинах выполняются

в арифметическом устройстве по микропрограммам основных арифметических операций. Микропрограммы различных арифметических операций реализуются за различное время. Например, операции умножения и деления занимают значительно больше времени, чем операции сложения и вычитания.

Увеличение скорости выполнения арифметических операций достигается повышением быстродействия элементной базы, на которой строятся вычислительные устройства,

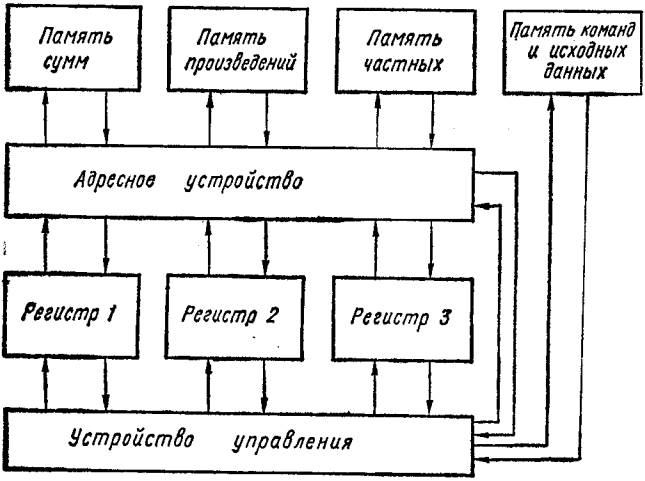


Рис. 20.

и различными методами построения структуры арифметического устройства, позволяющими ускорить выполнение операции сложения-вычитания.

Помимо повышения быстродействия элементной базы увеличение скорости выполнения арифметических операций достигают усовершенствованием схемных решений сумматоров и алгоритмов выполнения операций умножения и деления. Однако при таком подходе для выполнения операций умножения-деления всегда требуется значительно большее время, чем для операций сложения-вычитания. Таким образом, если свести время выполнения операций умножения-деления к времени выполнения операций сложения-вычитания, то можно значительно повысить быстродействие УВМ.

Приведение времени выполнения операции умножения-деления ко времени выполнения операции сложения-вычитания, т. е. времени реализации любой «длинной» операции ко времени реализации одного машинного такта, можно, если построить управляющую вычислительную машину по принципу системы массового обслуживания.

В такой УВМ результат получается не путем выполнения микропрограмм операций в арифметическом устройстве, а выборкой нужного числа из постоянного запоминающего устройства (ПЗУ). В этом случае роль обслуживающего аппарата играет устройство выборки чисел, и обслуживание заявки (выполнение операции) сводится к выбору результата выполнения операции из ПЗУ.

На рис. 20 изображена структурная схема вычислительного устройства (вычислителя), реализующего выборочный метод выполнения операций. При реализации таких вычислительных устройств трудно получить ПЗУ большой емкости, которая растет несоизмеримо быстрее роста разрядности чисел.

2. ЗАКОН РАСПРЕДЕЛЕНИЯ ВРЕМЕНИ ОЖИДАНИЯ АЛГОРИТМОВ В ВЫЧИСЛИТЕЛЯХ С ПЗУ

Для системы с одним обслуживающим аппаратом обработки заявок и m местами ожидания закон распределения времени ожидания начала обслуживания выражается формулой [48]

$$P\{\beta > t\} = \frac{P_1}{1-\rho} e^{-\nu t} \sum_{s=0}^{m-1} \frac{(\nu t)^s}{s!} (\rho^s - \rho^m), \qquad (8.1)$$

где $P\{\beta > t\}$ — вероятность того, что время ожидания обслуживания β превзойдет величину t; P_1 — вероятность того, что в системе имеется одна заявка; ρ — коэффициент загрузки системы.

Определим вероятность

$$P_1 = \rho p_0,$$

где вероятность отсутствия заявок в системе

$$p_0 = \frac{1-\rho}{1-\rho^{m+1}}.$$

Подставляя значения P_1 в формулу (8.1), получаем

$$P\{\beta > t\} = \frac{1}{1-\rho^{m+1}}\rho e^{-\nu t}\sum_{s=0}^{m-1}\frac{(\nu t)^s}{s!}(\rho^s - \rho^m).$$

Подставим вместо величины ν ее значение

$$\nu = \frac{\lambda}{\rho}.$$

В этом случае

$$P\{\beta > t\} = \frac{1}{1-\rho^{m+1}}\rho e^{-\frac{\lambda}{\rho}t}\sum_{s=0}^{m-1}\frac{\left(\frac{\lambda}{\rho}t\right)^s}{s!}(\rho^s - \rho^m). \tag{8.2}$$

После несложных вычислений получаем выражение для определения средней длительности ожидания

$$M_\beta = \int_0^\infty P\{\beta > t\}\,dt = \frac{\nu\rho_1}{(\nu-\lambda)^2}(1 - 2\rho^m + \rho^{m+1}),$$

или

$$M_\beta = \frac{\rho^2(1-2\rho^m+\rho^{m+1})}{\lambda(1-\rho^{m+1})(1-\rho)}. \tag{8.3}$$

При исследовании конкретной системы управления с несколькими потоками были получены значения $\rho = 0{,}38$; $\lambda = 533$ для машин с сумматором и $\rho = 0{,}2$; $\lambda = 533$ для машин с ПЗУ.

Подставив в выражение (8.2) значения $\rho = 0{,}38$; $\lambda = 633$, определим закон распределения времени ожидания $P\{\beta > t\} = f(t)$ при различных значениях $m = 1$; 3; 5; 7 и реализации заявок на машине с сумматором. По результатам вычислений построены графики рис. 21 сплошные линии). Аналогично находим функцию $P'\{\beta > t\} = f(t)$ при реализации задач выборочным методом, подставив в выражение (8.2) значения $\rho = 0{,}2$; $\lambda = 533$ (рис. 21 пунктирные линии).

На рис. 22 показан график зависимости $M_\beta = f(m)$ и $M'_\beta = f(m)$ при реализации задач на машине с сумматором и выборочным методом соответственно.

Как видно из рис. 21 и 22, при реализации задач выборочным методом значительно уменьшается вероятность потери

заявок за счет ожидания, а также сокращается средняя длительность ожидания по сравнению с реализацией этих же задач на УВС с сумматором.

3. МЕТОДЫ МИНИМИЗАЦИИ ПАМЯТИ В ВЫЧИСЛИТЕЛЯХ С ПЗУ

Метод разбиений. Разобьем разрядную сетку машины длиной n на m частей по p_i разрядов ($i = 1, 2, \ldots, m$) так, что

$$x = x_1 + x_2 + \cdots + x_m;$$
$$y = y_1 + y_2 + \cdots + y_m,$$

где x — аргумент функции; y — функция.

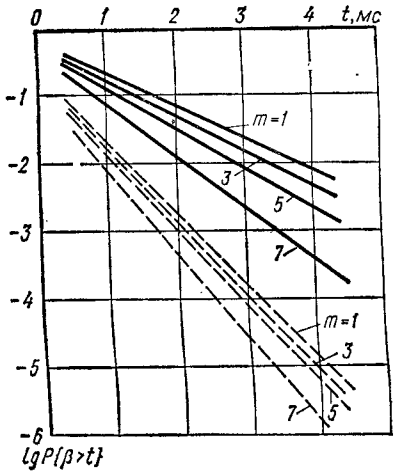


Рис. 21.

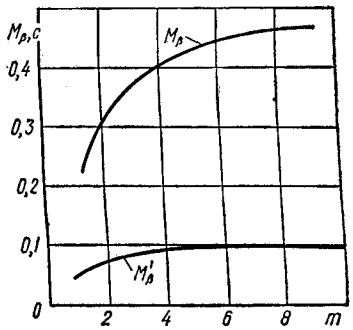


Рис. 22.

Будем считать, что y_1 — первое приближение $y^*(x)$, найденное по x_1 с учетом значений $x_2, x_3, \ldots, x_m$, а $y_1 + y_2$ — второе приближение $y^*(x)$, найденное по x_2 с учетом $x_1, x_3, \ldots, x_m$ и т. д.

Тогда для вычисления $y^*(x)$ с заданной точностью потребуется хранить m таблиц суммарным размером

$$\theta = \sum_{i=1}^{m} (b - a) q^{p_i},$$

где q — основание системы счисления.

Так как y_{i+1} сдвинуто относительно y_i на p_i разрядов, то нет необходимости производить суммирование составляющих $y^*(x)$, выбранных из соответствующих таблиц, а достаточно подать значения y_i в соответствующие разряды приемного регистра y_i (рис. 23).

Адресная система имеет, в общем случае, $(b-a)\,q^n$ выходов, в то время как каждое ПЗУ имеет $(b-a)\,q^{p_i}$ ячеек памяти. Это значит, что к некоторой ячейке памяти i-го ПЗУ может быть подключено

$$P = \frac{(b-a)\,q^n}{(b-a)\,q^{p_i}} = q^{n-p_i}$$

адресных шин. В соседних ПЗУ эти шины подключаются к различным ячейкам памяти, и поэтому, если шины не развязаны, возможен выбор более чем одной ячейки памяти i-м ПЗУ. Развязка (согласование) осуществляется схемой

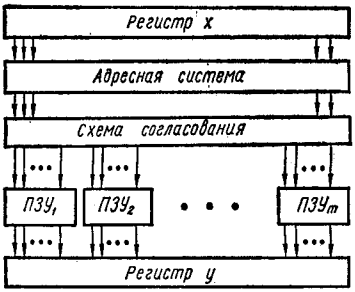


Рис. 23.

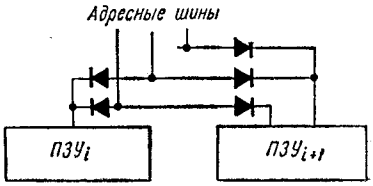


Рис. 24.

рис. 24 на диодных ячейках, общее количество которых определяется величиной

$$D = m\,(b-a)\,q^n\,.$$

При вычислении некоторых функций может оказаться, что

$$y_1 = \varphi_1(x_1);$$
$$y_2 = \varphi_2(x_2);$$
$$\cdots\cdots$$
$$y_m = \varphi_m(x_m).$$

Тогда каждое y_i зависит только от значения x_i, и каждая таблица имеет свою отдельную адресную систему. В этом случае схемы развязки исключаются, так как каждому выходу адресной системы соответствует только одна ячейка памяти соответствующего ПЗУ. Например, для функции

$$y_1 = kx_1;$$
$$y_2 = kx_2;$$
$$\cdots$$

$$y_m = kx_m,$$
$$y^*(x) = y_1 + y_2 + \cdots + y_m.$$

Алгоритм вычисления $y^*(x)$ состоит в выборке значений y_i из i-го ПЗУ по значениям x_i и в последовательном суммировании их.

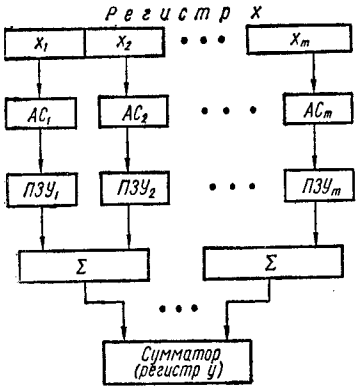


Рис. 25.

С целью минимизации времени суммирования можно использовать попарное сложение с помощью малоразрядных сумматоров (рис. 25).

Метод суперпозиции. В данном методе предлагается использование теории чисел для построения сокращенных таблиц значений $y = y(x)$. Действительно, если рассмотреть все значения мантиссы функции $y^* = y^*(x)$, которые она может принимать в разрядной сетке машины длиной n в десятичной системе счисления, то получим натуральный ряд чисел длиной q^n:

$$My^*_{10} \in \varepsilon \left\{ N \in M \,\middle|\, \begin{matrix} N \leqslant 2^n - 1 \\ N \geqslant 0 \end{matrix} \right\}, \qquad (8.4)$$

где My^*_{10} — мантисса $y^*(x)$ в десятичной системе счисления; N — целое число из натурального ряда.

Тогда задача построения сокращенной таблицы функции $y = y(x)$ сводится к задаче отыскания некоторого подмножества $M_1 \subset M$, из которого по простому алгоритму можно было бы получить все элементы множества M в формуле (8.4).

В натуральном ряде чисел выделим следующие основные подмножества.

1. Подмножество простых чисел

$$N \in \varepsilon \{a \in M_1 \mid a = p\},$$

где p — простое число.

Количество простых чисел (меньших или равных $N_{max} = q^n - 1$) в натуральном ряде определяется (из теории

чисел) величиной

$$\pi_1(a) = \frac{N_{\max}}{\ln N_{\max}} = \frac{q^n - 1}{\ln (q^n - 1)} .$$

2. Подмножество четных чисел (кратных 2)

$$N \in \varepsilon \{a \in M_2 \mid a = 2k\},$$

количество четных чисел

$$\pi_2(a) = \frac{1}{2} q^n = q^{n-1} .$$

3. Подмножество нечетных чисел

$$N \in \varepsilon \{a \in M_3 \mid a = 2k + 1\},$$

количество нечетных чисел

$$\pi_3(a) = \frac{1}{2} q^n = q^{n-1} .$$

4. Подмножество чисел, кратных числу A,

$$N \in \varepsilon \left\{ a \in M_4 \mid a = \frac{A}{a} \right\},$$

а количество таких чисел

$$\pi_4(a) = \frac{1}{A} q^n .$$

5. Подмножество чисел полных степеней m

$$N \in \varepsilon \{a \in M_5 \mid a = \alpha^m\}$$

с количеством чисел

$$\pi_5(a) = F\left[\sqrt[m]{q^n - 1}\right], \qquad (8.5)$$

где $F[t]$ — целая часть числа t.

Подмножества $M_1 - M_3$ не приводят к значительному сокращению таблицы. Подмножество M_4 может привести к значительному сокращению таблицы при больших A, однако, чем больше A, тем сложнее алгоритм определения остальных элементов множества (8.4).

Удобный алгоритм построения множества (8.4) из подмножества (8.5) вытекает из теоремы Гильберта, что для любого фиксированного числа m существует определенное число s, зависящее только от m, такое, что для каждого натурального числа N уравнение $N = a_1^m + a_2^m + \ldots + a_s^m$

имеет решение в целых неотрицательных числах a_1, a_2, ..., a_s. Это значит, что любое натуральное число N может быть получено суммированием (суперпозицией) конечного числа натуральных чисел a_i, являющихся полными m-ми степенями:

$$N = \sum_{i=1}^{s} a_i^m .$$

Для уменьшения времени вычислений рассмотрим случай $s = s_{\min}$, а $s_{\min} = \varphi(m)$.

Известно [35], что

$$\varphi(m) \geqslant 2^m + F\left[\frac{3^m}{2^m}\right] - 2.$$

Отсюда при $m = 2$ $s_{\min} = 4$, тогда

$$N = a_1^2 + a_2^2 + a_3^2 + a_4^2 .$$

Оказывается, что только два числа (23 и 239) требуют для своего представления $s_{\min} = 4$, а все остальные числа из ряда длиной q^n могут быть представлены суммой только трех квадратов целых положительных чисел.

Получаем следующий алгоритм определения значений мантисс $y^* = y^*(x)$, являющихся целыми положительными числами натурального ряда длиной q^n из чисел этого ряда, являющихся полными квадратами (1, 4, 9, ..., k^2):

$$N = a_1^2 + a_2^2 + a_3^2.$$

Для реализации этого алгоритма потребуется хранить три таблицы размерами

$$\theta_1 = F\left[\sqrt{(b-a)(q^n-1)}\right].$$

Суммарный размер таблицы определяется величиной

$$\theta = 3F\left[\sqrt{(b-a)(q^n-1)} + 2\right],$$

где две ячейки отведены для хранения мантисс, десятичный код которых соответствует числам 23 и 239.

Глава IX

ХАРАКТЕРИСТИКИ ДИНАМИЧЕСКИХ СВОЙСТВ МНОГОМОДУЛЬНЫХ УВС НА ПЗУ

1. АБСТРАКТНАЯ МОДЕЛЬ МНОГОМОДУЛЬНОЙ УВС НА ПЗУ

Одной из тенденций развития УВС на современном этапе является их постепенная «интегрализация», т. е. стремление к одновременному изготовлению большого количества элементов и узлов УВС, неразрывно связанных между собой. Значительные перспективы в этом направлении представляет табличный (выборочный) метод выполнения операций. Микроэлектронная и интегральная техника позволяет создавать оптимальные по габаритным размерам и весу блоки (модули) для вычисления значений любых или базовых функций, из которых может быть составлен вычислитель УВС, эффективно реализующий заданный класс алгоритмов.

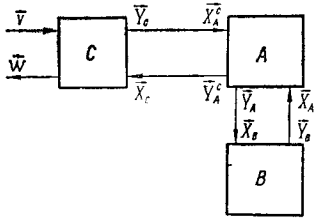


Рис. 26.

Рассмотрим абстрактную модель УВС, построенную на основе многомодульной памяти, интерпретированной конечным автоматом A (рис. 26). Автомат A имеет векторные входные и выходные сигналы. Назовем его операционным устройством, поскольку преобразование информации есть не что иное, как выполнение операции определенного вида. Для управления операциями необходимо иметь устройство для формирования управляющих сигналов. Представим это устройство в виде конечного автомата B, который назовем процессором. Для связи с объектом, а также для приема, хранения и выдачи управляющей информации имеется устройство C с соответствующими сигналами.

Рассмотрим основные связи и сигналы в абстрактной модели УВС. Процессор B имеет отдельные информационные шины для связи с каждым модулем памяти. Устройства ввода-вывода связаны со всеми модулями памяти, кроме памяти таблиц. Обозначим символом $\vec{v}$ — поток входной информации, а $\vec{w}$ — поток выходной информации. Вектор $\vec{X}_A$

множества входных сигналов автомата A совпадает с вектором $\vec{Y}_B$ всех выходных сигналов автомата B. Элементом этого множества является многокомпонентный вектор, называемый командой. Причем код операции этой команды определяет адрес (номер) требуемого модуля, в котором хранится таблица выполняемой операции, а адрес — номер ячейки таблицы, где хранится результат выполнения операции. Векторные входные сигналы $\vec{X}^{c}_{A}$ поступают в автомат A из устройства C. При этом $\vec{X}^{c}_{A}$ совпадает с управляющей информацией, участвующей в вычислениях. Длина слова $\vec{X}^{c}_{A}$, в общем случае, непостоянна и зависит от количества аргументов функции, для которой рассчитана таблица. На внешние устройства поступают также векторные выходные сигналы $\vec{Y}^{c}_{A}$ автомата A, причем номер устройства входит в состав этого сигнала. Множество $\vec{X}_B$ входных сигналов автомата B состоит из скалярных выходных сигналов Y_A автомата A и вектора $\vec{Y}_A$, являющегося результатом выполнения операций.

Рассмотрим функционирование абстрактной модели, используя для записи условий работы устройств язык регулярных событий [43], при следующих предположениях:

1) функционирование системы рассматривается относительно фиксированного момента времени t;

2) логическая структура функционирует в дискретном автоматном времени с интервалом дискретности T_0;

3) время выполнения алгоритма i определяется выражением

$$T_i = T_0 k_i \qquad (k_i \leqslant 1);$$

4) упорядоченность во времени потоков информации при выполнении i-й задачи описывается выражением вида $\nu_{k_i} < \nu_{l_i}$, означающим, что переработка элемента информации внутри потока ν_l начинается только после окончания последней команды внутри потока ν_k;

5) при обращении к оперативной памяти запросы от автомата C имеют более высокий приоритет, чем запросы от процессора B, чтобы избежать переполнения буферного накопителя;

6) запросы на доступ к оперативной и табличной памяти образуют очереди к соответствующим модулям и обслужива-

ются в соответствии с простейшей дисциплиной: «поступивший первым, обслуживается в первую очередь», т. е.

$$P_{\vec{X}_A(t)} > P_{\vec{X}_A(t+\Delta t)};$$

7) процессор имеет возможность «просматривать вперед» m команд, т. е. его цикл состоит из выборки m команд, и m обращений к памяти таблиц. В каждом цикле вырабатывается адрес к одному из модулей памяти и затем осуществляется переход к следующему циклу (только по выполнению условия 4). Во время выполнения этой последовательности модули памяти работают параллельно.

Условия функционирования системы запишем в следующем операторном виде:

$$И_0 \underset{1}{\square}\underset{1}{\square} B_t \underset{2}{\square}\underset{2}{\square} B_{tjl} \underset{3}{\square}\underset{3}{\square} \overset{ij}{M_{ij}} \underset{4}{\square}\underset{4}{\square} A_{lj} \underset{5}{\square}\underset{5}{\square} \Lambda_{ijl} \underset{6}{\square}$$

$$\underset{6}{\square} A_{lf} \underset{7}{\square}\underset{7}{\square} \Lambda_t \underset{8}{\square}\underset{8}{\square} P_{tjl} \overset{tjl}{\square}\underset{9}{\square}\underset{9}{\square} \overset{t}{P_t} \underset{10}{\square}\underset{10}{\square} Я_0. \qquad (9.1)$$

Автомат B_t реализует оператор диспетчеризации обслуживания задач в машине. В асинхронном режиме работы машины автомат B_t является сравнивающим устройством, определяющим наивысший приоритет поступивших на обслуживание задач. Входной алфавит автомата

$$X = (x_1, x_2, \ldots, x_k),$$

где x_l — приоритет i-й задачи; k — количество задач управления.

Множество выходных сигналов $Y = (y_1, y_2, y_3)$ ($y = 1$ при $x_i > x_j$; $y_2 = 1$ при $x_i < x_j$; $y_3 = 1$ при $x_i = x_j$).

Условия функционирования автомата B_t, записанные на языке регулярных событий, имеют вид:

$$S_3 = \{x_i \vee x_j\} S'|_{y_3} \qquad (i = j);$$

$$S_2 = \{x_1 \vee x_2 \vee \cdots \vee x_{i-1} \vee x_{i+1} \vee \cdots \vee x_{j-1} \vee$$
$$\vee x_{j+1} \vee \cdots \vee x_k\} x_{ij} \{x_i \vee x_j\} S'|_{y_2} \qquad (i < j);$$

$$S_1 = \{x_1 \vee x_2 \vee \cdots \vee x_{i-1} \vee x_{i+1} \vee \cdots$$
$$\cdots \vee x_{j-1} \vee x_{j+1} \vee \cdots \vee x_k\} x_{ij} \{x_i \vee x_j\} S'|_{y_1} \qquad (i > j);$$

$$S_4 = \overline{S_1 S_2 S_3 S' \mid e};$$

$$S' = (x_1 \vee x_2 \vee \cdots \vee x_k)(x_1 \vee x_2 \vee \cdots \vee x_k),$$

где x_{ij} (x_{ji}) — результат поразрядного сравнения слов x_i и x_j; e — пустое слово; S' — событие, определяющее сравнение входных слов попарно.

Приоритет выбран так, что старший номер задачи соответствует старшему приоритету.

В синхронном (детерминированном) режиме работы машины оператор B_i функционирует только один раз при составлении расписания. Входной алфавит автомата

$$X = (x_1,\ x_2,\ x_3,\ \ldots,\ x_k,\ x_{p_1},\ x_{p_2},\ \ldots,\ x_{p_k},\ x_{t_1},\ x_{t_2},\ \ldots,\ x_{t_k}),$$

где x_i — номер задачи; x_{p_i} — период обслуживания i-й задачи; x_{t_i} — время обслуживания i-й задачи.

Множество выходных сигналов $Y = (y_{\varphi S_1},\ y_{\varphi S_2},\ \ldots,\ \ldots,\ y_{\varphi S_k})$, где $y_{\varphi S_i}$ — фаза, т. е. номер такта S-го обслуживания i-й задачи.

Условия функционирования автомата B_i имеют вид:

$$S = \{S_1 S_2 S_3\};$$

$$S_1 = \{S_1' S_1'' \ \cdots \ S_1^{k-1}\};$$

$$S_2 = \{S_2' S_2'' \ \cdots \ S_2^{k+1}\};$$

$$S_4 = \overline{S_1 S_2 S_3}\,|_e,$$

где S_1 — условие того, что в каждом такте реализуется не более одной i-й задачи,

$$y_{\varphi S_i} = y_{\varphi S_j} \qquad (i = j).$$

При этом

$$S_1' = \{x_{p_1} \vee x_{p_1}x_{p_2} \vee x_{p_1}x_{p_3} \vee \cdots \vee x_{p_1}x_{p_k}\}\,|_{y_1'};$$

$$S_1'' = \{x_{p_2} \vee x_{p_2}x_{p_1} \vee x_{p_2}x_{p_3} \vee \cdots \vee x_{p_2}x_{p_k}\}\,|_{y_1''};$$

. . . . . .

$$S_1^k = \{x_{p_k} \vee x_{p_k}x_{p_1} \vee x_{p_k}x_{p_2} \vee \cdots \vee x_{p_k}x_{p_{k-1}}\}\,|_{y_1^k};$$

$$S_1^{k+1} = \overline{S_1' S_1'' \ \cdots \ S_1^k}\,|_e.$$

Событие S_2 показывает, что i-я задача должна быть реализована $\frac{1}{p_i}$ раз:

$$S_2' = \underbrace{(x_1 \vee x_2 \vee \cdots \vee x_k)(x_1 \vee x_2 \vee \cdots \cdots \vee x_k) \cdots (x_1 \vee x_2 \vee \cdots \vee x_k)}_{1/p_1}\,|_{y_2'};$$

$$S_2'' = \underbrace{(x_1 \vee x_2 \vee \cdots \vee x_k)(x_1 \vee x_2 \vee \cdots \cdots \vee x_k) \cdots (x_1 \vee x_2 \vee \cdots \vee x_k)}_{1/p_2}\,|_{y_2''};$$

. . . . . .

$$S_2^k = \underbrace{(x_1 \vee x_2 \vee \cdots \vee x_k)(x_1 \vee x_2 \vee \cdots \cdots \vee x_k) \cdots (x_1 \vee x_2 \vee \cdots \vee x_k)}_{1/p_k} |_{y_2^k} ;$$

$$S_2^{k+1} = \overline{S_2' S_2'' \cdots S_2^k}|_e .$$

Условие суммарного минимального времени обслуживания в каждом такте

$$S_3 = \{x_{t_1} \vee x_{t_2} \vee x_{t_k}\} ;$$

$$S_3' = \overline{S}_3 |_e .$$

B_{ijl} — оператор, определяющий очередь выполнения j-й операции i-й задачи на l-м модуле автомата A; зависит от расписания (программы) обслуживания команд и структуры автомата A. Входной алфавит автомата B_{ijl}

$$X = (x_1, x_2, \ldots, x_j, \ldots, x_n, x_{p_1}, x_{p_2}, \ldots, x_{p_j}, \ldots, x_{p_n},$$
$$x_{t_1}, x_{t_2}, \ldots, x_{t_j}, \ldots, x_{t_n}, x_{l_1}, x_{l_2}, \ldots, x_{l_j}, \ldots, x_{l_n},$$
$$x_{lk_1}, x_{lk_2}, \ldots, x_{lk_j}, \ldots, x_{lk_n}),$$

где x_j — номер операции j-го вида; x_{p_j} — вероятность появления операции j-го вида; x_{t_j} — время реализации операции j-го вида; x_{l_j} — номер модуля j-го типа; x_{lk_j} — количество модулей j-го типа.

Множество выходных сигналов $Y = (y_1, y_2, \ldots, y_\nu)$, где y_i — номер команды при выполнении i-й операции; ν — общее количество команд в программе. Автомат B_{ijl} функционирует один раз при составлении программы. Условий функционирования его в общем случае довольно много. Некоторые из них, записанные на языке регулярных событий, имеют вид:

условие выполнения всех операций, составляющих алгоритмы решения задач,

$$S' = \{x_1 \vee x_2 \vee \cdots \vee x_n\} |_{y'};$$

условие загрузки всех модулей блока A

$$S'' = \{x_{l_1} \vee x_{l_2} \vee \cdots \vee x_{l_n}\} |_{y''} ;$$

условие одновременной работы p модулей

$$S''' =$$
$$= \underbrace{(x_{l_1} \vee x_{l_2} \vee \cdots \vee x_{l_n}) \cdots (x_{l_1} \vee x_{l_2} \vee \cdots \vee x_{l_n})}_{r} |_{y'''} ;$$

условие того, что j-я операция реализуется $\frac{1}{p_j}$ раз,

$$S^{(4)} =$$

$$= \underbrace{(x_1 \vee x_2 \vee \cdots \vee x_n) \cdots (x_1 \vee x_2 \vee \cdots \vee x_n)}_{1/p_j} |_{y^{(4)}}.$$

Оператор M_{ij} проверяет необходимость масштабирования составляющих векторов $\vec{X}_A^c$ и $\vec{X}_A$ при выполнении j-й операции i-й задачи. Операнды, являющиеся адресами, при обращении к модулям табличной памяти, в общем случае, имеют величину, получающуюся либо в результате выполнения предыдущей операции (например, суммирования), либо в результате значительных отклонений от оптимума управляемого процесса, превышающую длину входного слова автомата A. Для обеспечения автоматической работы системы необходимо ввести масштабирование компонент векторов $\vec{X}_A^c$ и $\vec{X}_A^c$ при $|\vec{X}_A^c| > |\vec{X}_A^{c*}|$ или $|\vec{X}_A| > |\vec{X}_A^*|$, где $\vec{X}_A^*$, $\vec{X}_A^{c*}$ — значения операндов, при которых не нарушается автоматическая работа системы.

Если автомат M_{ij} функционирует как устройство сравнения двух чисел, то его условия работы имеют вид

$$S_1 = \{S_1' S_1'' \cdots S_1^n\}|_{y_1} \text{ при } |\vec{X}_A| = |\vec{X}_A^*|,$$

где $S_1^{(i)}$ — события равенства одноименных разрядов векторов $\vec{X}_A$ и $\vec{X}_A^*$:

$$S_1^i = \{x_{00}^i \vee x_{11}^i\} \qquad (i = 1, 2, \ldots, n);$$

$$S_2 = \{S_2' S_2'' \cdots S_2^n\}|_{y_2} \text{ при } |\vec{X}_A| > \vec{X}_A^*|.$$

S_2^i — условие того, что i-й разряд вектора $\vec{X}_A$ содержит 1, а вектора $\vec{X}_A^*$ содержит 0:

$$S_2^i = \{x_{00}^i \vee x_{01}^i \vee x_{10}^i \vee x_{11}^i\}\, x_{10}^i \{x_{00}^i \vee x_{11}^i\} \qquad (i = 1, 2, \ldots, n);$$

$$S_3 = \{S_3' S_3'' \cdots S_3^n\}|_{y_3} \text{ при } |\vec{X}_A| < |\vec{X}_A^*|,$$

где S_3^i — условие того, что i-й разряд $\vec{X}_A$ содержит 0,

а $\vec{X}_A^* - 1$:

$$S_3^l = \{x_{00}^l \vee x_{01}^l \vee x_{10}^l \vee x_{11}^l\}\, x_{01}^l \{x_{00}^l \vee x_{11}^l\} \qquad (l = 1, 2, \ldots, n);$$

$$S_A = \overline{S_1 S_2 S_3}\,|_e.$$

x_{kl}^l — буква входного алфавита $X = (x_{00}^l, x_{01}^l, x_{10}^l, x_{11}^l)$.

Оператор A_{ij} масштабирует операнд j-й операции i-й задачи. Так как масштабный коэффициент $k_м = |\vec{X}_A^*| / /|\vec{X}_A|$, то функционирование автомата A_{ij} сводится к определению $\vec{X}_A^*$.

Входной алфавит автомата

$$X = (x_{A_1}, x_{A_2}, \ldots, x_{A_n}, x_1, x_2, \ldots, x_{k_1} x_{p_1 i}, x_{p_2 i}, \ldots, x_{p_{ki}}),$$

где x_{A_i} — компоненты вектора $\vec{X}_A$; x_j — номер операции j-го типа; $x_{p_j i}$ — количество операций j-го типа в i-й задаче.

Множество выходных сигналов $Y = (y_1, y_2, \ldots, y_n)$, где $y_i = X_{A_i}^*$. Условия работы автомата запишем в виде

$$S = \{S_1 S_2 S_3\},$$

где условие того, что в расчете $\vec{X}^*$ участвуют все компоненты $\vec{X}$,

$$S_1 = \{x_{A_1} \vee x_{A_2} \vee \cdots \vee x_{A_n}\};$$

условие того, что $k_м$ рассчитывается по всем операциям, входящим в i-ю задачу,

$$S_2 = \{x_1 \vee x_2 \vee \cdots \vee x_k \vee x_{p_1 i} \vee x_{p_2 i} \vee \cdots \vee x_{p_{ki}}\};$$

условие того, что $k_м$ рассчитывается с учетом $\vec{Y}_A$ и $\vec{X}_A$,

$$S_3 = \{x_{A_1} \vee x_{A_2} \vee \cdots \vee x_{A_n}\} S'.$$

Событие S' показывает, что значение предыдущего сигнала было заполнено. Оператор Λ_{ijl} назначает l-й модуль автомата A для выполнения j-й операции i-й задачи в зависимости от числа операндов в команде. Этот оператор необходим, если автомат A содержит таблицы функции от двух и более переменных. Автомат Λ_{ijl} имеет множество входных сигналов X $(x_1, x_2, \ldots, x_{nk})$, k — число операндов, и множество выходных сигналов $Y = (y_1, y_2, \ldots, y_k)$. При

поступлении на вход автомата слова длиной n, т. е. $X = (x_1, x_2, \ldots, x_n)$, на выходе формируется сигнал y_1. При поступлении на вход слова длиной $2n$ формируется сигнал y_2 и т. д. Регулярные выражения функционирования автомата имеют вид:

$$S_0 = \{x_1 \vee x_2 \vee \cdots \vee x_n\} S' |_{y_1};$$
$$S_1 = \{x_1 \vee x_2 \vee \cdots \vee x_{2n}\} S' |_{y_2};$$
$$\cdots\cdots\cdots\cdots\cdots\cdots$$
$$S_{nk} = \{x_1 \vee x_2 \vee \cdots \vee x_{nk}\} S' |_{y_{nk}};$$
$$S_{nk+1} = \overline{S_0 S_1 \cdots S_{nk}} |_e;$$
$$S' = \underbrace{(x_1 \vee x_2 \vee \cdots \vee x_{nk}) \cdots (x_1 \vee x_2 \vee \cdots \vee x_{nk})}_{k}.$$

В событие S_{nk+1} входят все слова, не кратные k. Автомат A_{lf} есть оператор выработки адреса f ячейки модуля l, в котором хранится результат выполнения j-й операции i-й задачи. Этот автомат функционирует как обычная адресная система постоянных запоминающих устройств. Входной алфавит автомата $X = (x_0, x_1)$. Множество выходных сигналов $Y = (y_1, y_2, \ldots, y_m)$, $m = 2^n$. Регулярные выражения функционирования автомата имеют вид:

$$S_0 = S' x_0 x_0 \cdots x_0 |_{y_0};$$
$$S_1 = S' x_0 x_0 \cdots x_0 x_1 |_{y_1};$$
$$S_2 = S' x_0 x_0 \cdots x_1 x_0 |_{y_2};$$
$$\cdots\cdots\cdots\cdots$$
$$S_m = S' x_1 x_2 \cdots x_1 x_1 |_{y_m};$$
$$S_{m+1} = \overline{S_0 S_1 \cdots S_m} |_e;$$
$$S' = \underbrace{(x_0 \vee x_1)(x_0 \vee x_1) \cdots (x_0 \vee x_1)}_{n}.$$

Работа оператора Λ_i сводится к выдаче результата выполнения j-й операции в оперативную память автомата A, либо в буферный накопитель автомата C. Функционирование этого автомата аналогично функционированию оператора Λ_{ijl} с той лишь разницей, что записывается входное слово, а не выборка. По команде от P_{ijl} осуществляется переход на внутренний цикл, т. е. на оператор B_{ijl} при вы-

полнении условия $j_t \leqslant j_{\text{икон}}$, где j_t — текущее значение операций при решении i-й задачи; $j_{\text{икон}}$ — общее количество операций в i-й задаче. Поскольку сравниваются два числа, функционирование автомата P_{ijl} аналогично функционированию автомата M_{ij}. А по команде от P_i осуществляется переход на внешний цикл, т. е. на оператор B_i. При необходимости может быть введен приоритетный переход на более важную, чем $i+1$ задача. В остальном автомат P_i аналогичен автоматам P_{ijl} и M_{ij}.

Интервал дискретности функционирования УВС

$$T_0 = t_{B_i} + \sum_{l=1}^{m} t_{B_{ijl}} + \sum_{i=1}^{m}\sum_{j=1}^{kl} t_{M_{ij}} + \sum_{i=1}^{\omega}\sum_{j=1}^{\nu} t_{A_{ij}} + \\ + \sum_{i=1}^{m}\sum_{j=1}^{k_l} t_{\Lambda_{ijl}} + \sum_{i=1}^{m}\sum_{j=1}^{k_l} t_{A_{lf}} + \sum_{j=1}^{\varphi} t_{\Lambda_j}, \qquad (9.2)$$

где m — количество задач, реализуемых за время T_0; k_i — количество операций в i-й задаче; ω — количество задач, требующих масштабирования переменных; ν — количество операций в алгоритме масштабирования; φ — количество пересылок в оперативную память; $t_{M_{ij}}$, $t_{A_{ij}}$, $t_{\Lambda_{ijl}}$, $t_{A_{lf}}$, t_{λ_t} — времена функционирования соответствующих автоматов при реализации j-й операции i-й задачи.

Так как для реализации автоматов M_{ij}, A_{ij}, Λ_{ijl}, A_{lf}, Λ_j требуются различные устройства, то при

$$t_{M_{ij}} = t_{A_{ij}} = t_{A_{ijl}} = t_{A_{lf}} = t_{\Lambda_j} = T' \qquad (9.3)$$

функционирование их во времени можно совместить. Тогда

$$T_0 = t_{B_i} + t_{B_{ijl}} + T'. \qquad (9.4)$$

Время функционирования автомата B_i в асинхронном режиме работы системы, либо среднее время простоя системы, зависящее от расписания обслуживания алгоритмов управления и от времени перехода от одной задачи управления к другой при детерминированном режиме работы машины, определяется величиной

$$t_{B_i} = \frac{C - \sum_{n=1}^{N_0}\sum_{i=1}^{m} t_i}{N_0} + t_{\text{пер}}, \qquad (9.5)$$

где t_i — время решения i-й задачи; $C = N_0 T_0$ — цикл обслуживания всех алгоритмов управления с периодом P_i

каждый; N_0 — количество интервалов дискретности T_0 в цикле; $t_{\text{пер}}$ — время, необходимое на подготовку системы к решению новой задачи. Время, зависящее от качества программ реализации алгоритмов управления,

$$t_{B_{ijl}} = \frac{1}{L} \sum_{\nu=1}^{k_l} \eta\nu, \tag{9.6}$$

где ν — время, необходимое на выполнение операции передачи управления; k_l — общее количество передач управления в l-й задаче; η — коэффициент, определяющий «глубину» передачи управления (чем «глубже» передается управление, тем «дальше» в ЗУ находится команда, на которую передается управление, тем больше $t_{B_{ijl}}$); L — коэффициент параллельности (совмещения) выполнения операций.

Рассмотрим задачи, от которых зависит эффективность функционирования системы.

Во-первых, это задачи диспетчеризации (составление расписания обслуживания) алгоритмов управления для получения минимума выражения (9.5). Эта задача возникает ввиду того, что по условиям управления задачи могут решаться неодинаковое количество раз за весь период и за цикл управления. Циклом называется время, за которое все задачи управления реализуются хотя бы один раз. Так как УВС работает синхронно с другими устройствами системы управления, то цикл разбивают на такты длительностью T_0. Расписание составляется таким образом, чтобы выполнялось условие

$$T_0 = \left[\sum_{l=1}^{m} t_l\right]_{\max} \to \min. \tag{9.7}$$

Во-вторых, для выполнения условия (9.3) необходимо сократить время функционирования автомата A_{ij}, уменьшив количество вычислительных операций.

В-третьих, для выполнения условия (9.3) необходимо выбрать оптимальную структуру автомата A. Этот автомат состоит из оперативной памяти для хранения программ, констант и промежуточных результатов, от структуры которой зависят величины $t_{B_{ijl}}$ и t_{Λ_j}, и памяти таблиц, структура и объем которой определяют величины $t_{\Lambda_{ijl}}$ и $t_{A_{ij}}$ соответственно.

2. КРИТЕРИИ ДИНАМИЧЕСКИХ СВОЙСТВ УВМ НА ПЗУ И ВЫБОР ИХ ЧАСТОТНЫХ ХАРАКТЕРИСТИК

К параметрам, характеризующим динамические свойства УВМ при работе с реальными объектами [51], кроме того, можно отнести:

частотные характеристики численного метода, реализованного в ЦВМ;

частота выдачи решений (определяет разрешающую способность УВМ; при работе в замкнутом контуре бо́льшая частота выдачи решений обеспечивает устойчивость);

время отработки (время, требуемое для перехода УВМ из одного состояния в другое в соответствии с изменением величины на одном или более входов. Это время зависит от «скачка» на входе и масштаба переменных, выбранного в соответствии с требуемой точностью);

скорость обработки некоррелированных данных (при статистической обработке входных величин, сортировке, сравнении);

время запаздывания (оно не всегда равно периоду выдачи решений, так как может зависеть от скорости передачи данных между блоками);

точность вычислений, определяемая выбранным численным методом, реализованным в УВМ, и разрядностью чисел, с которыми оперируют машины;

разрешаемая способность по амплитуде входного сигнала, определяемая точностью работы входных и выходных преобразователей и разрядностью чисел, с которыми оперирует машина.

Перечисленные требования достаточно многообразны, а часто и противоречивы, поэтому не могут быть приняты в качестве частных показателей эффективности УВМ: во-первых, они не отражают в явном виде качества УВМ как функционального элемента системы, а во-вторых, их число настолько велико, что выбор оптимального решения затруднен. Поэтому возникает задача выявления такой группы показателей, которая возможно более полно характеризовала бы качество УВМ и была бы немногочисленной. С другой стороны, желательно, чтобы выбранные показатели качества имели достаточно простую аналитическую зависимость от параметров системы.

Если УВС работает в линейной области, то ее динамические свойства можно охарактеризовать тем, насколько

частотная характеристика реальной программы $W^*(j\bar{\omega})$ отличается от частотной характеристики той же задачи, реализованной в идеальной машине $W(j\bar{\omega})$.

Для линейного оператора с памятью частотная характеристика $W^*(j\bar{\omega})$ является дробнорациональной функцией частотной характеристики метода, реализованного в блоках ЦВМ [141]. С другой стороны, на динамику машины оказывает влияние способ передачи данных между блоками.

Частотная характеристика относительной ошибки метода

$$\Delta(j\bar{\omega}) = \frac{K(j\bar{\omega}) - K^*(j\bar{\omega})}{K(j\omega)} = 1 - \lambda(j\omega),$$

где $\lambda(j\omega) = \frac{K^*(j\omega)}{K(j\omega)}$ — относительная частотная характеристика метода; $K^*(j\omega)$ — частотная характеристика принятого метода; $K(j\omega)$ — частотная характеристика идеального метода.

Можно рассматривать УЦВМ как последовательное соединение дискретного фильтра, изменяющего закон модуляции входной последовательности импульсов, и элемента запаздывания. Передаточная функция дискретного фильтра

$$K(j\omega) = \frac{y(j\omega)}{x(j\omega)} = \frac{\sum_{k=0}^{m} a_k \exp\{-jkT\omega\}}{1 + \sum_{n=1}^{n} b_n \exp\{jnT\omega\}} =$$

$$= \frac{a_0 + a_1 \exp(-j\omega t) + \cdots + a_m \exp(-j\omega mT)}{1 + b_1 \exp(-j\omega t) + \cdots + b_n \exp(-j\omega nT)}$$

характеризует алгоритм (программу переработки входной информации), а передаточная функция элемента запаздывания

$$K(p) = e^{-pT}$$

характеризует промежуток времени между вводом данных в ЦВМ и выводом из нее результатов реализации алгоритма (программы). Однако такой подход к определению динамических свойств УВМ позволяет только качественно оценить несовершенство алгоритмов переработки информации, не учитывая при этом конкретную структуру УВМ и ошибки

(округления, сбойные), которые возникают из-за внутренних шумов машины.

Частотный анализ программ при работе ЦВМ в реальном масштабе времени проведен в работах [40, 131]. Однако при подобном анализе не учитывался полный объем задачи и, в частности, та часть программы вычислений, которая относится к нелинейным операциям. Метод оценки необходимого быстродействия ЦВМ в зависимости от допустимой погрешности ε, порядка применяемой квадратурной формулы k и от граничной частоты системы ω_c, основанный на согласовании допустимого шага интегрирования

$$T_{\max} = f(\varepsilon,\ k,\ \omega_c)$$

с временем одного шага решения, рассмотрен в работе [144].

В ряде работ предлагается оценка динамических свойств ЦВМ, учитывающая нелинейный характер машины, вводится критерий P, представляющий собой вероятность работы ЦВМ в нелинейной области при подаче на ее вход сигнала с нормальным распределением вероятностей,

$$P = 1 - \frac{1}{2\pi} \int_{-\frac{L}{\sigma_\Delta}}^{\frac{L}{\sigma_\Delta}} l^{-\frac{1}{2}\omega^2}\, d\omega,$$

где L — порог ограничения ЦВМ по первой разности на входе; σ_Δ — дисперсия первой разности входного сигнала; ω — условная переменная.

По этому критерию невозможно оценить ошибку и качество системы. Так как УВС является мультипрограммной системой обработки управляющей информации, работающей в реальном масштабе времени, то частотную характеристику в этом случае можно записать в следующем виде:

$$W^*(j\omega) = f(x_1,\ x_2,\ \ldots,\ x_n),$$

где $x_1, x_2, \ldots, x_n$ — параметры мультипрограммной системы.

Количество параметров x_i в общем случае может быть довольно велико, что затрудняет исследование таких систем. Для удобства исследования ограничим величину n следующими основными параметрами [19]:

N — количество регулируемых процессов по данному объекту управления;

M_i — количество задач, которые необходимо решить для выработки одного сигнала управления по i-му процессу;

K_j — коэффициент сложности машинной реализации алгоритма j-й задачи, который определяется сумммарным количеством приведенных элементарных операций;

ε_j — коэффициент точности решения j-й задачи, определяемый отношением длины разрядной сетки α к длине разрядной сетки β, $\varepsilon_j = q^{-\alpha+\beta}$, если точность вычислений достигается конструктивным путем, или числом повторных вычислений, если точность вычислений достигается программным путем;

L_j — длина очереди по каждой задаче, определяемая временем задержки информации на выходе УВМ по NM — 1 задачам;

P_j — коэффициент параллельности обработки данных, определяемый отношением числа всех программ (или операций), которые необходимо обработать для выработки одного сигнала управления, к числу одновременно обрабатываемых программ (операций);

S — коэффициент структурной сложности машины, характеризующийся временем задержки обрабатываемых программ в системе прерывания программ (СПП). Это время затрачивается в СПП на распределение обрабатываемых программ по устройствам и блокам машины;

ω_0 — быстродействие выполнения элементарных операций средней длины.

Представим данную мультипрограммную систему как систему массового обслуживания с ожиданием без потерь, характеризующуюся функцией распределения времени обслуживания.

Рассмотрим частный случай. Пусть функция распределения времени обслуживания имеет вид

$$F_j(t) = 1 - \exp\{-\lambda_j t\},$$

где $F_j(t)$ — определяет вероятность того, что время обслуживания одной заявки (в нашем случае время решения одной задачи) меньше некоторого наперед заданного значения t; λ_j — интенсивность решения задачи $\left(\nu_j = \frac{1}{\lambda_j}\right.$ — среднее время обработки одной задачи$\left.\right)$.

Для выработки одного сигнала управления необходимо обработать M задач. Рассмотрим функцию распределения

времени обслуживания программ

$$F_j(t) = 1 - \exp\{-\lambda t\},$$

где $\lambda = \dfrac{1}{\sum\limits_{j=1}^{M} \nu_j}$.

Очевидно, что $F_j(t)$ тем больше, чем меньше λ при постоянном M. Но ν_j является функцией параметров, характеризующих алгоритмы обработки программ и вычислитель:

$$\nu_j = f(K_j,\ \varepsilon_j,\ L_j,\ P_j,\ S,\ \omega_0).$$

Для нахождения вида этой функции решим совместно уравнение, связывающее размерности параметров и ν_j

$$[\nu_j] = f([K_j],\ [\varepsilon_j],\ [l_j],\ [P_j],\ [S],\ [\omega]), \tag{9.8}$$

и систему

$$\nu_j = \begin{cases} a_1 K_j \\ a_2 \varepsilon_j \\ a_3 L_j \\ a_4 \dfrac{1}{P_j} \\ a_5 S \\ a_6 \dfrac{1}{\omega_0} \end{cases}, \tag{9.9}$$

где $a_1,\ a_2,\ \ldots,\ a_6$ — коэффициенты пропорциональности.

Решая совместно выражения (9.8) и (9.9), получаем

$$\nu_j = a_1 a_2 a_4 a_6 \frac{K_j \varepsilon_j}{P_j \omega_0} + a_3 L_j + a_5 S.$$

Для рассмотрения качественных характеристик $F_j(t)$ положим $a_1 = a_2 = \ldots = a_6$, тогда

$$F_j(t) = 1 - \exp\left\{-\frac{t}{\sum\limits_{j=1}^{M}\left(\frac{K_j \varepsilon_j}{P_j \omega_0} + L_j + S\right)}\right\}.$$

Если УВС управляет N независимыми процессами, то она характеризуется функцией

$$F(t) = 1 - \exp\left\{-\frac{t}{\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{M}\left(\frac{K_{ji} \varepsilon_{ji}}{P_{ji} \omega_0} + L_{ji} + S\right)}\right\}. \tag{9.10}$$

Для характеристики регулируемого процесса введем функцию $G_i(t)$ — априорную вероятность того, что за время t (считая с момента прихода последнего импульса управления) параметры i-го управляемого процесса изменились в наперед заданных пределах. Это значит, что при соответствующем выборе пределов изменения параметров процесса к приходу следующего импульса управления процесс является управляемым. Для нормального хода управляющего процесса необходимо, чтобы

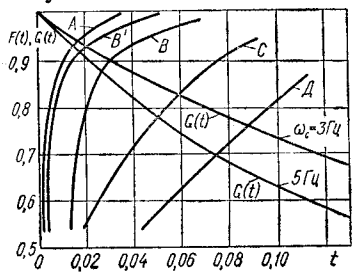


Рис. 27.

$$F(t) \geqslant G(t)_{\text{зад}}.$$

Функция $G_i(t)$ в общем случае может иметь произвольный вид при условии, что она определена на отрезке [0,1] и на этом отрезке монотонна. Для импульсной системы автоматического управления [141] можно положить, что

$$G_i(t) = \gamma_i \exp(-\omega_{ci} t),$$

где ω_{ci} — граничная частота системы i-го регулируемого процесса; γ_i — коэффициент состояния регулирующих и устанавливающих органов объекта управления к моменту прихода очередного управляющего импульса, который определяется отношением вероятности безошибочной отработки управляющего импульса к вероятности отработки с ошибкой δ, т. е.

$$\gamma_i = \frac{P(0)}{P(\delta)}.$$

Если объект управления имеет N независимых управляемых процессов, то он характеризуется функцией

$$G(t) = \prod_{i=1}^{N} \gamma_i \exp\left(-\sum_{i=1}^{N} \omega_{ci} t\right). \qquad (9.11)$$

На рис. 27 показаны графики функций $F(t)$ и $G(t)$, рассчитанные по формулам (9.10) и (9.11) для следующих параметров:

Кривая A: $\omega_c = 3 \div 5$ *Гц*; $N = \gamma_i = \varepsilon_j = P_j = 1$; $L_j = S = 0$; $\omega_0 = 10^6$ операций/с; $K_j = 1000$ операций; $M = 6$;

Кривая B: $\omega_c = 3 \div 5$ *Гц*; $N = \gamma_i = \varepsilon_j = 1$; $L_j = S = 0$; $\omega_0 = 10^{-5}$ операций/с; $K_j = 1000$ операций; $M = 6$; $P_j = 2$;

Кривая C: $\omega_c = 3 - 5$ *Гц*; $N = \gamma_i = \varepsilon_j = P_j = 1$; $L_j = S = 0$; $\omega_0 = 10^5$ операций/с; $K_j = 1000$ операций; $M = 6$;

Кривая D: $\omega_c = 3 - 5$ *Гц*; $N = \gamma_i = P_j = 1$; $L_j = S = 0$; $\omega_0 = 10^5$ операций/с; $K_j = 1000$ операций; $M = 6$; $\varepsilon_j = 2$.

Из графика видно, что, например, при $G(t) = 0{,}9$ в данной полосе частот только управляющие машины типа A и B имеют $F(t) \geqslant G(t) = 0{,}9$, и они могут нормально управлять данным процессом.

При выбранном $G(t) = 0{,}95$ ни одна из машин не пригодна для данного управляемого процесса. В этом случае необходимо либо увеличивать параметры ω_0 или P_j, либо уменьшать K_j или ε_j.

Приведенная методика оценки частотной характеристики позволяет выбрать из имеющихся УВМ такую, которая сможет осуществить оптимальное управление данным объектом управления. Кроме того, полученные соотношения могут быть использованы для оценки параметров проектируемой УВМ.

Оценим, например, частотные характеристики УВМ типа B, построенной по классическому принципу и использующей для выполнения длинных операций табличный метод. Пусть количество длинных операций составляет 30% от всего набора операций, а отношение времени выполнения короткой операции в классической ЦВМ к времени выполнения длинной операции равно 1 : 5. Тогда для классической ЦВМ при $t = 0{,}02$с $F(t) = 0{,}75$. Для ЦВМ с табличным методом выполнения длинных операций при $t = 0{,}02$с $F(t) \approx 0{,}985$.

Это значит, что вероятность точного управления процессом при использовании табличного метода увеличивается на 13%. Если же при тех же условиях увеличить P_j до 4, то частотная характеристика такой ЦВМ будет в 1,85 раз лучше частотной характеристики обычной ЦВМ. На рис. 27 для этих данных построена частотная характеристика B'. Она почти совпадает с кривой A, имеющей значение ω_0, на порядок выше. Использование в ЦВМ табличного метода может дать тот же эффект, что и увеличение быстродействия элементов машины на порядок, что связано с огромными материальными затратами.

3. АЛГОРИТМЫ ДИФФЕРЕНЦИАЛЬНЫХ ПОКАЗАТЕЛЕЙ КАЧЕСТВА ФУНКЦИОНИРОВАНИЯ МНОГОМОДУЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ПЗУ

Из всей совокупности параметров, наиболее полно и удобно характеризующих качество системы в целом, можно выделить три: показатель времени вычислительной работы, точность выполнения решения задачи [51, 82] и коэффициент параллелизма (совместимости) работы отдельных устройств во времени.

Вопросы точности обработки информации достаточно подробно рассмотрены в работах [19, 82, 168], анализ которых показывает, что использование табличного метода в УВМ не приводит к потере точности выполнения. Для учета величины задержки информации на выходе УВМ необходимо знать объем вычислительной работы, необходимой для реализации отдельных алгоритмов управления, и качественный и количественный состав элементарных операций алгоритмов, реализуемых на машине.

В качестве объекта исследования рассмотрим некоторую систему, уравнения движения и законы управления которой реализуются УВМ.

Оценим величину задержки информации на выходе УВМ при детерминированной последовательности реализации алгоритмов. Определим статистические характеристики алгоритмов, учитывая только чистое время выполнения арифметических операций.

Пусть общий период решения всех задач C разбит на N_0 тактов длительностью T_0, так что

$$C = N_0 T_0. \tag{9.12}$$

Каждая задача в общем периоде C реализуется k_i раз, т. е. с периодом дискретности

$$T_i = k_i T_0, \tag{9.13}$$

откуда

$$k_i = \frac{C}{T_i} \cdot \frac{T_i N_0}{C}. \tag{9.14}$$

В каждом T_0 реализуется или одна, или, в общем случае, m задач ($m \leqslant m_0$), причем

$$\sum_{i=1}^{m_0} t_i \leqslant T_0, \tag{9.15}$$

где t_i — время реализации i-й задачи.

Частота появления i-й задачи

$$p_i = \frac{k_i}{\sum_{i=1}^{m_0} k_i}. \tag{9.16}$$

Пусть известна алгоритмическая сложность S_i каждой задачи, которая представляется в виде суммы всех типов операций

$$S_i = \sum_{j=1}^{l} r_{ji},$$

где r_{ji} — количество операций j-го типа в i-й задаче.

Тогда информационная работа, выполняемая ЦВМ за период,

$$S_\Sigma = \sum_{i=1}^{m_0} k_i S_i,$$

общее количество j-х операций за период

$$R_j = \sum_{j=1}^{l} k_i r_{ji}. \tag{9.17}$$

Частота появления j-й операции во множестве задач

$$p_j = \frac{R_j}{S_\Sigma} = \frac{\sum_{j=1}^{l} k_i r_{ji}}{\sum_{i=1}^{m_0} k_i \sum_{j=1}^{l} r_{ji}}. \tag{9.18}$$

Используя соотношения (9.11) — (9.18), легко вычислить при известном времени выполнения каждой элементарной операции величину задержки информации на выходе УВМ по каждой задаче, по группе задач или по типам операций. Однако при детерминированном законе реализации задач управления эта величина зависит также и от порядка (расписания) обработки алгоритмов в УВМ. При синхронной работе устройств УВМ с другими блоками и устройствами системы управления величина такта T_0 не может быть выбрана только из условия непрерывной работы машины. Могут быть простои УВМ внутри такта, если группа алгоритмов, реализуемых в этом такте, требует для этого меньшего времени, чем величина T_0. В результате реальное время задержки информации на выходе УВМ за весь цикл

реализации алгоритмов будет несколько больше величины задержки, если бы задачи управления решались не в комплексе, а по отдельности. Если t_j — время выполнения операции j-го типа, то величина задержки информации по каждой задаче с учетом расписания обработки алгоритмов

$$t_i^* = t_i + t_S,$$

где время реализации i-й задачи

$$t_i = p_i \sum_{i=1}^{m_0} t_j \frac{\sum_{j=1}^{l} k_i r_{ji}}{\sum_{i=1}^{m_0} k_i \sum_{j=1}^{l} r_{ji}}. \qquad (9.19)$$

Величина t_S определяется средним временем простоя машины за такт

$$t_S = \frac{C - \sum_{i=1}^{m_0} k_i t_i}{N_0}. \qquad (9.20)$$

Глава X

МОДЕЛИРОВАНИЕ МОДЕЛЕЙ УВС С ПРИОРИТЕТАМИ НА ЭЦВМ

1. МОДЕЛЬ ПРИОРИТЕТНОЙ УВС

Для экспериментальной проверки результатов аналитических исследований, полученных в гл. VII, используются алгоритмы статистического моделирования (метод Монте-Карло), моделей УВС с реальными законами распределений времени обслуживания заявок.

Применение метода Монте-Карло при реализации на ЭЦВМ предполагает наличие либо автономного датчика случайных чисел, либо специальной программы для получения псевдослучайных чисел с помощью некоторого рекуррентного соотношения. При исследовании датчиков случайных величин на ЭЦВМ алгоритмы псевдослучайных чисел либо требуют очень большого числа команд для их реализации, либо получаемая последовательность чисел

не удовлетворяет критериям устойчивости случайных чисел [50, 122, 130].

Для получения последовательности псевдослучайных чисел используется следующее рекуррентное соотношение:

$$\xi_{i+1} = \{\xi_i + \xi_{i-1}\},$$

где ξ_0 и ξ_1 — начальные значения случайных чисел.

Рассмотрим алгоритмы моделирования системы, которая исследовалась аналитическими методами в гл. VII. Система состоит из одного обслуживающего аппарата (вычислителя) с ограниченным объемом буферного накопителя, в который поступают на обработку простейшие потоки сообщений, обладающие различными приоритетами на обслуживание. Время реализации заявки аппаратом фиксировано и равно θ. В зависимости от способа организации работы буферного накопителя (раздельные по каждому потоку или общий) различаются и алгоритмы, описывающие процесс обслуживания заявок.

Раздельные буферные накопители. Пусть в систему поступают два потока информации L_1 и L_2 соответственно высшего и низшего приоритетов. Буферный накопитель *БН1* для приема заявок потока с высшим приоритетом имеет m_1 ячеек, а буферный накопитель *БН2* для заявок потока низшего приоритета — m_2 ячеек. Условимся, если в ячейке *БН* записано какое-либо число, отличное от нуля, то ячейка занята, а если записан нуль, то ячейка свободна. Блок-схема алгоритма моделирования процесса обслуживания в такой системе показана на рис. 28

Опишем работу данного алгоритма. Датчики случайных чисел *ДСЧ1* и *ДСЧ2* формируют длины интервалов τ_1 и τ_2 между моментами поступления заявок потоков L_1 и L_2 соответственно как случайные величины, распределенные по закону Пуассона. Из двух текущих значений моментов поступления заявок T_{k_1} и T_{k_2} выбираем меньшее, которое определяет заявку потока, поступившую в систему раньше. Затем определяем незанятость аппарата к моменту поступления очередной заявки.

Если окажется, что $T_{k\min} \geqslant t$ (t — момент окончания обслуживания заявки), т. е. обслуживающий аппарат свободен, то он обращается к буферному накопителю *БН1*; если *БН1* свободен от заявок, то аппарат обращается к *БН2*, и если он также свободен, поступившая заявка

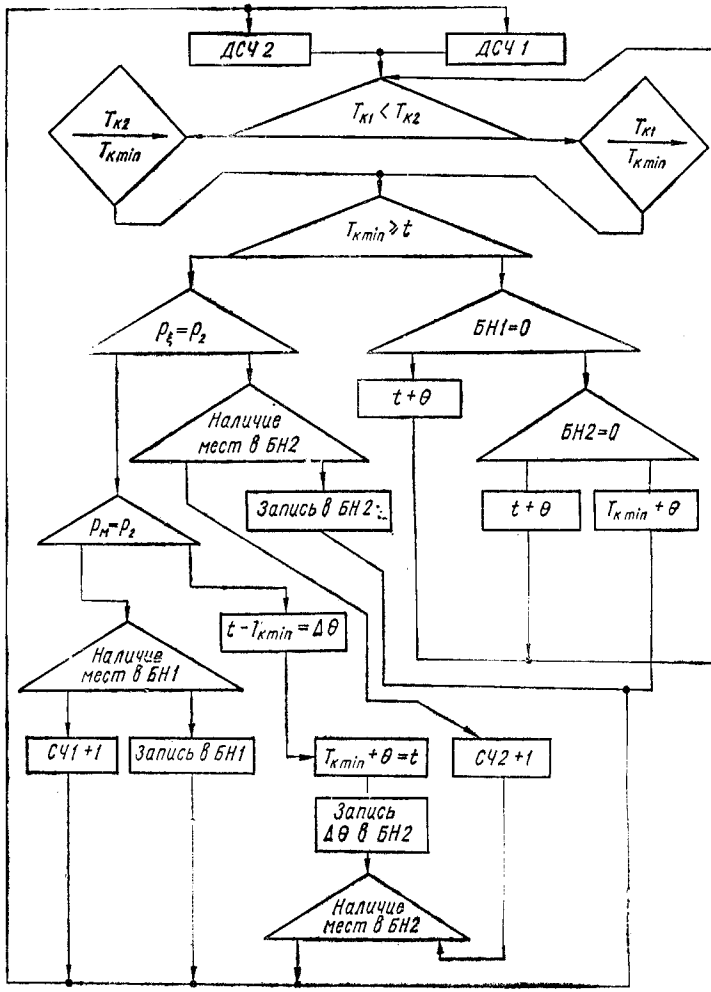


Рис. 28.

реализуется и изменяется текущее значение

$$t_{тек} = T_{k\min} + \theta,$$

после чего датчики вырабатывают очередные значения случайных величин τ_1 и τ_2.

Если $T_{k\min} \geqslant t$ (*БН1* свободен) но в *БН2* имеются заявки, то из *БН2* на обслуживающий аппарат поступает

заявка, стоящая на первом месте, и меняется величина $t_{\text{тек}}$. При этом

$$t_{\text{тек}} = t + \theta \text{ или } t_{\text{тек}} = t + \Delta\theta,$$

в зависимости от того, успеет или не успеет обслужиться заявка до прихода заявки более высокого приоритета, где $\Delta\theta$ — время, необходимое для дообслуживания прерванной заявки,

$$\Delta\theta = t - T_{k\min}.$$

Если $T_{k\min} \geqslant t$, но в *БН1* имеются заявки, то при освобождении обслуживающего аппарата выбирается первая заявка из *БН1* и обслуживается до конца, при этом текущее значение

$$t_{\text{тек}} = t + \theta.$$

Предположим теперь, что $T_{k\min} < t$, т. е. следующая заявка поступит раньше, чем освободится обслуживающий аппарат. В этом случае необходимо выяснить, заявка какого потока находится на обслуживании и заявка какого потока поступила в момент $T_{k\min}$.

Если на обслуживании находится заявка потока L_2 низшего приоритета, и поступила заявка из этого же потока, то проверяется наличие свободных мест в накопителе *БН2*, и если таковые имеются, поступившая заявка записывается на первое свободное место. В противном случае поступившая заявка теряется, и к счетчику потерянных заявок *СЧ2* потока L_2 добавляется единица.

Если на обслуживании находится заявка потока L_2, а поступила заявка потока L_1, то обслуживание прерывается и прерванная заявка возвращается в *БН2*, где ожидает дообслуживания, а обслуживающий аппарат приступает к обслуживанию поступившей заявки и обслуживает ее до конца, т. е.

$$t_{\text{тек}} = T_{k\min} + \theta.$$

Если в *БН2* находилось m_2 заявок, то заявка, стоящая на последнем месте, теряется, и к *СЧ2* добавляется единица.

Если на обслуживании находится заявка потока L_1, а поступила заявка потока L_2, то проверяется наличие свободных мест в *БН2*, и если они есть, поступившая заявка записывается в первую свободную ячейку, если *БН2* полностью занят, то эта заявка теряется, и к счетчику *СЧ2* добавляется единица.

Если на обслуживании находится заявка потока L_1 и поступила заявка из этого же потока, то проверяется наличие свободных мест в *БН1*, и поступившая заявка записывается на первое свободное место, если такие имеются, и теряется, если *БН1* полностью занят. При потере заявки потока L_1 к счетчику *СЧ1* (счетчик потерянных заявок потока L_1) добавляется единица.

Общий буферный накопитель. Пусть в систему поступают два потока информации L_1 и L_2, и *БН* для приема заявок обоих потоков имеет m ячеек. Поток L_1 обладает абсолютным приоритетом перед потоком L_2. Блок-схема алгоритма моделирования процесса обслуживания в такой системе показана на рис. 29.

Так же как и в случае раздельных накопителей, *ДСЧ1* и *ДСЧ2* формируют длины интервалов между моментами поступления заявок потоков L_1 и L_2 соответственно. Из двух текущих значений моментов поступления заявок T_{k_1} и T_{k_2} выбирается меньшее и определяется, поступит ли она после момента окончания обслуживания заявки (t), находящейся на обслуживании, или до окончания обслуживания.

Если $T_{k\min} \geqslant t$, т. е. обслуживающий аппарат свободен, тогда проверяется наличие в *БН* заявок потоков L_1; если таковые имеются, то первая из них направляется на обработку, и текущее значение

$$t_{\text{тек}} = t + \theta.$$

Если в *БН* отсутствуют заявки потока L_1, но имеется недообслуженная заявка потока L_2, то она дообслуживается, и текущее значение $t_{\text{тек}}$ принимает значение

$$t_{\text{тек}} = t + \Delta\theta;$$

если в *БН* нет заявок потока L_1 и отсутствуют недообслуженные заявки потока L_2, но имеются заявки потока L_2, то первая из них поступает на обработку, и текущее значение

$$t_{\text{тек}} = t + \theta;$$

если в *БН* отсутствуют заявки потоков L_1 и L_2, то обрабатывается поступившая заявка, т. е.

$$t_{\text{тек}} = T_{k\min} + \theta,$$

после чего датчики *ДСЧ1* и *ДСЧ2* вырабатывают очередные значения случайных величин.

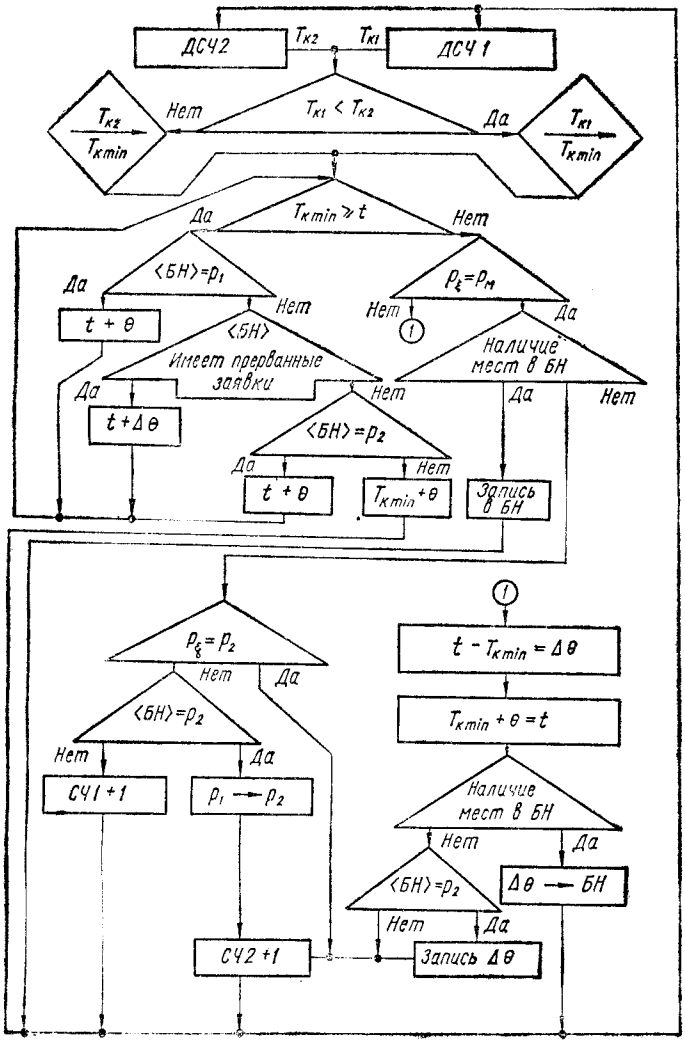

ДСЧ2
$T_{к2}$
$T_{к1}$
ДСЧ1
$T_{к2}$ / $T_{к min}$
Нет
$T_{к1} < T_{к2}$
Да
$T_{к1}$ / $T_{к min}$
Да
$T_{к min} \geqslant t$
Нет
Да
⟨БН⟩=p_1
Нет
$p_ξ = p_н$
Нет
Да
1
$t + \theta$
⟨БН⟩
Имеет прерванные заявки
Наличие мест в БН
Да
Нет
Да
Нет
$t + \Delta\theta$
⟨БН⟩=p_2
Да
Нет
$t + \theta$
$T_{к min} + \theta$
Запись в БН
1
$p_ξ = p_2$
Нет
Да
$t - T_{к min} = \Delta\theta$
⟨БН⟩=p_2
$T_{к min} + \theta = t$
Нет
Да
СЧ1 + 1
$p_1 \rightarrow p_2$
Наличие мест в БН
Нет
Да
⟨БН⟩=p_2
$\Delta\theta \rightarrow$ БН
Нет
Да
СЧ2 + 1
Запись $\Delta\theta$

Рис. 29.

Предположим, что $T_{k\min} < t$, т. е. очередная заявка поступит раньше, чем освободится обслуживающий аппарат. В этом случае необходимо выяснить, заявка какого приоритета находится на обслуживании и заявка какого приоритета поступила в момент $T_{k\min}$.

Если на обслуживании находится заявка потока L_1, и в систему поступила заявка этого же потока или потока L_2, то при наличии свободных мест поступившая заявка записывается в *БН*. При отсутствии свободных мест в *БН* поступившая заявка потока L_2 теряется, и к счетчику (*СЧ2*) потерянных заявок потока L_2 добавляется единица. При отсутствии свободных мест в *БН* и поступлении заявки потока L_1 проверяется наличие в *БН* заявок потока L_2, и если такие имеются, то поступившая заявка записывается вместо одной из заявок потока L_2, которая теряется. В этом случае к счетчику *СЧ2* добавляется единица. Если же в *БН* отсутствуют заявки потока L_2, то поступившая заявка потока L_1 теряется, и к счетчику *СЧ1* добавляется единица.

Если $T_{k\min} < t$, на обслуживании находится заявка потока L_2, а поступила в момент $T_{k\min}$ заявка потока L_1, то обслуживание заявки потока L_2 прерывается, вычисляется значение времени, необходимого для дообслуживания,

$$\Delta\theta = t - T_{k\min},$$

обслуживается поступившая заявка потока L_1

$$T_{k\min} + \theta = t_m$$

и при наличии свободных мест в *БН* величина $\Delta\theta$ записывается на свободное место. Если свободных мест в *БН* нет, то при наличии в *БН* заявок потока L_2 вместо одной из них записывается недообслуженная заявка, вытесненная заявка теряется, и к *СЧ2* добавляется единица. Если *БН* полностью занят заявками потока L_1, то теряется прерванная заявка, и к счетчику *СЧ2* добавляется единица, после чего датчики *ДСЧ1* и *ДСЧ2* вырабатывают очередные значения случайных величин.

Сопоставление результатов аналитических исследований с экспериментальными, полученными с помощью программ статистического моделирования, показало применимость разработанного метода для расчета некоторых параметров рассматриваемых структур.

Замена реального закона распределения времени обслуживания заявок экспоненциальным при определении ем-

кости буферного накопителя дает погрешность, не превышающую 0,5 единицы младшего разряда ячейки, что вполне удовлетворительно, так как полученные расчетные значения округляются до целых чисел.

2. ОПТИМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ АЛГОРИТМОВ, ПОДЛЕЖАЩИХ РЕАЛИЗАЦИИ НА УВС

Пусть в УВС на реализацию поступают k типов алгоритмов с определенными периодами поступления $n_i T_0$ ($i = 1, 2, \ldots, k$), и все время работы УЦВМ разбито на N_0 тактов длительностью T_0 так, что в каждом T_0 реализуется один или, в общем случае, m алгоритмов ($m \leqslant k$), причем

$$\sum_{i=1}^{m} t_i \leqslant T_0,$$

где t_i — время реализации алгоритмов i-го типа.

Необходимо задать такое расписание реализации алгоритмов, чтобы максимальная сумма $\sum_{i=1}^{m} t_i$ имела минимальное значение, при котором сохраняется периодичность реализации алгоритмов.

Система алгоритм-заявка, УВМ-обслуживающий аппарат представляют собой систему массового обслуживания, которая может быть исследована с помощью теории массового обслуживания. Обычно предполагается, что заявки образуют поток — последовательность заявок с определенным чередованием моментов их поступления. Если известны математические описания потоков заявок и процессов обслуживания, то методами теории массового обслуживания решают задачи оценки качества обслуживания, выбора оптимальных параметров обслуживающих систем, структуры этих систем и т. д.

Очевидно, при необходимости, можно поставить и другую задачу — обратную задачу теории массового обслуживания — нахождение математического описания потоков заявок (закона распределения времени поступления заявок), которое удовлетворяло бы данной системе массового обслуживания с известными процессами обслуживания, структурой и параметрами. В такой постановке задача нахождения

оптимального расписания поступления заявок в литературе не встречалась, и аналитические методы решения ее не разработаны.

Задача является обратной задачей теории массового обслуживания при следующих ограничениях.

1. Моменты времени φ_i прихода на обслуживающий аппарат только первых заявок по каждому потоку случайны. Моменты поступления остальных заявок детерминированные и определяются периодом следования заявок по этому потоку и моментом прихода первой заявки.

2. Случайные моменты времени φ_i прихода первых заявок по каждому потоку равновероятны и имеют целочисленные значения, кратные T_0.

Задачи решались методом статистических испытаний (метод Монте-Карло). Решение прямой задачи состоит в разработке алгоритма, при помощи которого можно вырабатывать случайные реализации заданных потоков однородных событий, а также моделировании процесса функционирования обслуживающих систем. Этот алгоритм используется для многократного воспроизведения реализаций случайного процесса обслуживания. Полученная информация о состояниях процесса подвергается статистической обработке с целью оценки величин, являющихся показателями качества обслуживания.

Поскольку в нашем случае поток требований не задан, необходимо разработать алгоритм получения k-мерного случайного вектора φ с целочисленными параметрами. Алгоритм состоит в следующем. С помощью датчика случайных чисел получаем случайные значения ξ, равномерно распределенные в интервале $[0; 1]$. Эти значения ξ преобразовались в интервале $[1; n_i]$.

Затем, после получения случайного вектора φ потока заявок, проводится моделирование процесса обслуживания заявок при заданном значении $C = N_0 T_0$. Величина N_0 выбирается как наибольшее общее кратное всех n_i, чтобы соблюдалась периодичность обслуживания. Время T_0 на первом шаге задается равным $t_{i\max}$. Если в процессе моделирования обслуживания заявок получается положительный ответ, т. е. все заявки обслуживаются с заданными периодами за время C, то вектор φ принимается как оптимальный, а ступенчатая функция $\varphi_i = f(i, t)$ принимается как оптимальное расписание поступления заявок.

Если в процессе моделирования обслуживания получается отрицательный ответ, то генерируется новый случайный вектор φ и т. д.

Если после проведения l подобных испытаний окажется, что при данном C нет распределения, удовлетворяющего данной системе, т. е. получены все отрицательные ответы, то выбирают новое значение $T_0 = t_{i\max} + a$, где a — наименьшее общее кратное значение t_i, и процесс моделирования повторяется. И так до тех пор, пока не получат оптимальный вектор φ.

Алгоритм моделирования системы на ЦВМ, записанный в операторной форме, имеет вид

$$A_0 A_1 P_2 A_3 A_4 P_5 P_6 A_7 A_8 P_9 Я_{10},$$

где A_0 — оператор подготовки (запись нулей и констант в рабочие ячейки); A_1, A_3 — операторы формирования случайного вектора φ.

Оператор A_1 формирует k случайных значений номеров потока, из которого в момент φ_i, формируемый оператором A_3, приходит заявка. Эти операторы разделены оператором P_2 для увеличения быстродействия.

P_2 — оператор проверки значения k. Так как значения k формируются по случайному закону, то могут появиться значения k, по которым заявки уже обслужены. В этом случае переходят к оператору A_1.

A_4 — оператор моделирования процесса обслуживания производит выборку значения t_i и получение значений

$$S_k = \sum_{i=1}^{m} t_i.$$

P_5 — оператор проверки условия

$$S_k \leqslant T_0.$$

Если это условие выполняется, то переходят к оператору A_7; если не выполняется, то возвращаются к оператору A_3.

A_i — оператор обслуживания заявок. По этому оператору в ячейки с номерами φ_i записывается значение S_j.

P_6 — оператор проверки условия

$$l \leqslant l_{зад},$$

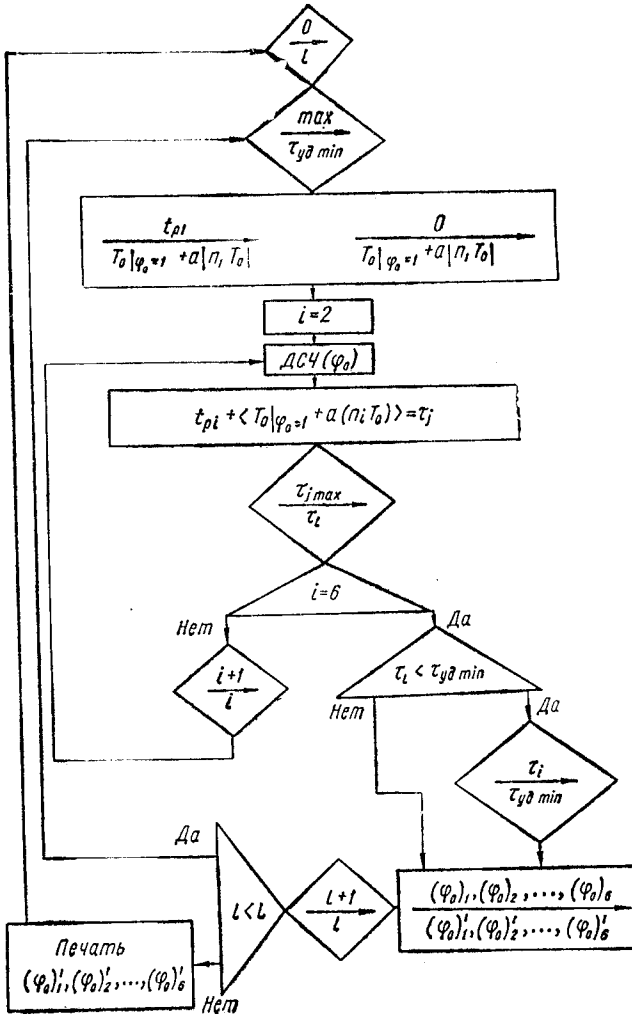

0
l
max
τуд min
tpl
T0|φ0=1 + a|n, T0|
0
T0|φ0=1 + a|n, T0|
i=2
ДСЧ (φ0)
tpi + < T0|φ0=1 + a (ni T0) > = τj
τj max
τi
i=6
Нет
Да
i+1
i
τi < τуд min
Нет
Да
τi
τуд min
Да
i<l
i+1
i
(φ0)1, (φ0)2, ..., (φ0)6
(φ0)'1, (φ0)'2, ..., (φ0)'6
Печать
(φ0)'1, (φ0)'2, ..., (φ0)'6
Нет

Рис. 30.

где l — число переходов от оператора P_5 к оператору A_3. По этому оператору проверяется вероятность того, что при выбранном T_0 имеется случайный вектор φ, удовлетворяющий данной системе обслуживания. Если условие $l \leqslant l_{зад}$ не выполняется, то с вероятностью p можно сказать, что при выбранном T_0 не существует вектора φ, удовлетворяющего данной системе. Значение T_0 увеличивается на величину a и переходят на оператор A_0.

A_8 — оператор обработки. По этому оператору запоминается значение φ.

P_9 — оператор проверки условия — все ли заявки обслужены. Если это условие выполняется, то переходят к оператору $Я_{10}$.

$Я_{10}$ — оператор выдачи результатов, печать полученного случайного вектора φ и значения T_0, при котором вектор φ удовлетворяет данной системе.

Работа ЦВМ начинается с занесения в ячейки x констант: значений n_i, нулей в ячейки β, соответствующие интервалам T_0, и нулей в счетчики l и k. Счетчик l — это счетчик числа полученных отрицательных ответов на вопрос: удовлетворяет ли вектор φ данной системе. Счетчик k — счетчик числа потоков, заявки которых обслужены системой.

С помощью датчика случайных чисел вырабатывается случайное число ξ и формируется первый параметр вектора φ. Проверяется условие $\alpha_k = 0$. Это условие определяет, обслужены ли заявки по потоку k или нет. Если $\alpha_k = 0$, то заявки обслужены, и программа возвращается на получение нового k. С помощью датчика случайных чисел формируется случайное число, являющееся моментом прихода заявки по k-му потоку.

Как только будут обслужены все заявки по всем каналам, т. е. все $x_k = 0$, машина переходит к печати значений φ в виде моментов времени прихода первых заявок.

По описанному алгоритму, блок-схема которого показана на рис. 30, составлена программа для оптимального распределения задач управления, подлежащих реализации на УЦВМ, и получения вектора φ при минимально возможном значении T_0.

ЛИТЕРАТУРА

1. Анисимов Б. В., Четвериков В. Н. Основы теории и проектирования цифровых вычислительных машин. М., Машгиз, 1962.

2. Артамонов Г. Т., Будагян Э. А. Два метода анализа моделей ЦВМ как многофазных систем массового обслуживания с конечными очередями.— «Известия АН СССР. Техническая кибернетика», 1968, № 3.

3. Бабенко К. И. К теории второй вариации функционалов на классе однолистных функций. М., ДАН СССР. Т. 5, 1970, № 5.

4. Бабенко К. И. О построении области коэффициентов однолистных функций класса *S*. М., ДАН СССР. Т. 194, 1970, № 2.

5. Бахвалов Н. С. Об устойчивом вычислении значений многочленов.— «Журнал вычислительной математики и математической физики», 1971, № 6.

6. Бахвалов Н. С. О вычислении кратных интегралов с автоматическим выбором шага.— В сб.: «Вычислительные методы и программирование». Под ред. В. В. Воеводина, М., Изд-во МГУ, 1965.

7. Бахвалов Н. С. О параболических системах с малыми параметрами при старших производных. М., ДАН СССР. Т. 174, 1967, № 2.

8. Бахвалов Н. С. О свойствах оптимальных методов решения задач математической физики.— «Журнал вычислительной математики и математической физики», 1970, № 3.

9. Башарин Г. П. Некоторые математические методы исследования систем связи и управления сложной структуры. Автореферат докторской диссертации. М., Изд. МЭИС, 1967.

10. Башарин Г. П. Обслуживание двух потоков на однолинейной системе с ограниченным числом мест ожидания и абсолютным приоритетом.— «Известия АН СССР. Техническая кибернетика», 1967, № 5.

11. Башарин Г. П. Об обслуживании двух потоков с относительным приоритетом на полнодоступной системе с ограниченным числом мест ожидания.— «Известия АН СССР. Техническая кибернетика», 1967, № 2.

12. Башарин Г. П. О пуассоновских обслуживающих системах с абсолютным приоритетом и обратной связью.— В сб.: «Массовое обслуживание в системах передачи информации». М., «Наука», 1969.

13. Башарин Г. П. О расчете буферной памяти в вычислительных системах с несколькими входящими информационными потоками.— В сб.: «Системы управления и коммутации». Вып. 18. М., «Наука», 1965.

14. Бекишев Г. А. Об алгоритмах, эффективно реализуемых на вычислительных машинах.— В сб.: «Вычислительные системы». Вып. 7. Новосибирск, Изд-во СО АН СССР, 1968.

15. Бекишев Г. А. О распараллеливании вычислительных алгоритмов.— В сб.: «Вычислительные системы». Вып. 5. Новосибирск, Изд-во СО АН СССР, 1963.

16. Березин И. С., Жидков Н. П. Методы вычислений. Т. 1, 2. М., Физматгиз, 1959, 1962.

17. Березюк Н. Т. Выбор оптимального пути реализации алгоритмов управляющими машинами летательных аппаратов.— В сб.: «Самолетостроение и техника воздушного флота». Вып. 11. Харьков, Изд-во ХГУ, 1967.

18. Березюк Н. Т. Один вариант алгоритмов дифференциальных арифметических операций для УВМ летательных аппаратов.— В сб.: «Радиоэлектроника летательных аппаратов». Вып. 3. Харьков, Изд. ХАИ, 1971.

19. Березюк Н. Т., Близнюк В. Д. О способе выбора частотных характеристик управляющих вычислительных машин.— В сб.: «Радиоэлектроника летательных аппаратов». Вып. 2. Харьков, Изд. ХАИ, 1971.

20. Березюк Н. Т. и др. Алгоритмы и алгоритмические системы. Харьков, Изд. ХВКИУ, 1966.

21. Березюк Н. Т., Фурманов К. К. Исследование некоторых оценок и алгоритмов повышенной точности вычислений для управляющих машин.— В сб.: «Кибернетическая техника». Вып. 4. Киев, изд-во АН УССР, 1970.

22. Березюк Н. Т., Шаронов В. Б. Исследование построения системы управляющих вычислительных машин методами теории массового обслуживания.— В сб.: «Управляющие машины и системы». Вып. 3. Киев, Изд-во АН УССР, 1970.

23. Березюк Н. Т., Шаронов В. Б. Исследование цифровых вычислительных систем с приоритетами методами теории массового обслуживания.— В сб.: «Техническая кибернетика». Вып. 2. Киев, Изд-во АН УССР, 1970.

24. Березюк Н. Т., Шилов В. И. Элементы алгоритмической теории вычислительных систем. Харьков, Изд. ХВКИУ, 1968.

25. Берж К. Теория графов и ее применения. М., Изд. иностр. лит., 1962.

26. Брик В. А. Выбор оптимальных параметров вычислительного устройства с помощью метода линейного программирования.— «Известия АН СССР. Энергетика и автоматика», 1962, № 5.

27. Бронштейн О. И. и др. Об однолинейной системе массового обслуживания с потерями.— «Известия АН СССР. Техническая кибернетика», 1965, № 4.

28. Бронштейн О. И., Рыков В. В. Об оптимальных дисциплинах обслуживания в управляющих системах.— В сб.: «Труды III Всесоюзного совещания по автоматическому управлению». Одесса, 1965.

29. Бронштейн О.И., Рыков В. В. Об оптимальных приоритетах в системах массового обслуживания.— «Известия АН СССР. Техническая кибернетика», 1965, № 6.

30. Будагян Э. А. Анализ производительности структур ЦВМ методом вложенных цепей Маркова. Автореферат кандидатской диссертации. М., 1968.

31. Будагян Э. А. Исследование одной модели ЦВМ как системы массового обслуживания.— В сб.: «Вычислительная техника в системах управления летательными аппаратами». Вып. 182. М., Изд. МАИ, 1968.
32. Бусленко Н. П. Математическое моделирование производственных процессов. М., «Наука», 1964.
33. Бусленко Н. П. Моделирование сложных систем. М., «Наука», 1968.
34. Бусленко Н. П., Шрейдер Ю. А. Метод статистических испытаний. М., Физматгиз, 1961.
35. Бухштаб А. А. Теория чисел. М., «Просвещение», 1966.
36. Витушкин А. Г. Оценка сложности задач табулирования. М., Физматгиз, 1959.
37. Воеводин В. В. Ошибки округления и устойчивость в прямых методах линейной алгебры. М., Изд-во МГУ, 1969.
38. Воеводин В. В. Численные методы алгебры. М., «Наука», 1966.
39. Воеводин В. В. и др. Вычислительные методы и программирование. М., Изд-во МГУ, 1965.
40. Гетманов А. Г. Динамические особенности реализации некоторых линейных операций на ЦВМ.— В сб.: «Автоматическое управление и вычислительная техника». Вып. 3. М., Машгиз, 1960.
41. Глушков В. М. Два универсальных критерия эффективности ЦВМ. М., ДАН СССР. 1960, № 5.
42. Глушков В. М. О применении абстрактной теории автоматов для минимизации микропрограмм.— «Известия АН СССР. Техническая кибернетика», 1964, № 1.
43. Глушков В. М. Синтез цифровых автоматов. М., Физматгиз, 1962.
44. Глушков В. М. Теория алгоритмов. Киев, Изд. КВИРТУ, 1961.
45. Глушков В. М. и др. Вычислительные машины с развитыми системами интерпретации. Киев, «Наукова думка», 1970.
46. Гнеденко Б. В. Основные направления исследований в теории массового обслуживания.— В сб.: «Труды VI Всесоюзного совещания по теории вероятностей и математической статистике». Вильнюс, 1962.
47. Гнеденко Б. В. Про одне узагальнення формул Ерланга.— «Доповіді АН УРСР», 1959, № 4.
48. Гнеденко Б. В., Коваленко И. Н. Введение в теорию массового обслуживания. М., «Наука», 1966.
49. Гнеденко Б. В., Коваленко И. Н. Лекции по теории массового обслуживания. Киев, Изд. КВИРТУ, 1963.
50. Голенко Д. Н. Образование случайных величин с произвольным законом распределения.— «Вычислительная математика», 1959, № 5.
51. Голубев-Новожилов Ю. С. Многомашинные комплексы вычислительных средств. М., «Советское радио», 1967.
52. Голубков Ю. А. К правильному выбору алгоритмов аппроксимации функции для ЭЦВМ, работающих в реальном масштабе времени.— В сб.: «Труды семинара отделения структурных и логических схем». Вып. 3. М., Изд-во АН СССР, 1965.
53. Данилов В. Л. и др. Математический анализ. Функции, пределы, ряды, цепные дроби. М., Физматгиз, 1961.

54. Дегтярев Е. К. Об одном классе многофазных систем массового обслуживания для структурно-временного анализа ЦВМ.— «Известия АН СССР. Техническая кибернетика», 1969, № 1.

55. Дегтярев Е. К. Стохастические модели функционирования асинхронных ЦВМ.— «Известия АН СССР. Техническая кибернетика», 1967, № 2.

56. Демидович Б. П. и др. Численные методы анализа. М., Физматгиз, 1967.

57. Демидович Б. П., Марон И. А. Основы вычислительной математики. М., Физматгиз, 1963.

58. Дроздов Е. А. и др. Основы вычислительной техники. М., Оборонгиз, 1964.

59. Духовный И. М. Некоторые задачи обслуживания двух потоков с приоритетами.— В сб.: «Массовое обслуживание в системах передачи информации». М., «Наука», 1969.

60. Евреинов Э. В., Косарев Ю. Г. О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во СО АН СССР, 1962.

61. Евреинов Э. В., Косарев Б. Г. Однородные универсальные вычислительные системы высокой производительности. М., «Наука», 1966.

62. Ершов А. П. Операторные алгоритмы.— В сб.: «Проблемы кибернетики». Вып. 3. М., Физматгиз, 1960.

63. Ершов А. П. Сведение задачи экономии памяти при составлении программы к раскраске вершин графов. М., ДАН СССР, Т. 15, 1962, № 4.

64. Иванов В. В. Вопросы точности и эффективности вычислительных алгоритмов.— В сб.: «Обзор достижений в области кибернетики и вычислительной техники». Вып. 2. Киев, Изд-во ИК АН УССР, 1969.

65. Иванов В. В. Об оптимальных алгоритмах минимизации в классах дифференцируемых функций. М., ДАН СССР. Т. 201, 1971, № 3.

66. Иванов В. В. Об оптимальных алгоритмах минимизации функций некоторых классов.— «Кибернетика», 1972, № 4.

67. Иванов В. В. Теория приближенных методов и ее применение к численному решению сингулярных интегральных уравнений. Киев, «Наукова думка», 1968.

68. Иванов В. В., Задиpака В. К. Некоторые новые результаты о теории аппроксимации.— «Вычислительная техника», 1966, № 2.

69. Калужнин Л. А. Об алгоритмизации математических задач.— В сб.: «Проблемы кибернетики». Вып. 2. М., Физматгиз, 1959.

70. Каляев А. В. Теория цифровых интегрирующих машин и структур. М., «Советское радио», 1970.

71. Камынин С. С. и др. Об автоматизации программирования при помощи программирующей программы.— В сб.: «Проблемы кибернетики». Вып. 1. М., Физматгиз, 1958.

72. Карп Р. М. Заметка о приложении теории графов к программированию для цифровых вычислительных машин.— «Кибернетический сборник», 1962, № 4.

73. Карцев М. А. Арифметика цифровых машин. М., Физматгиз, 1969.

74. Касаткин А. С., Кузьмин И. В. Оценка эффективности автоматизированных систем контроля. М., «Энергия», 1967.

75. Китов А. И., Криницкий Н. А. Электронные цифровые машины и программирование. М., Физматгиз, 1961.
76. Клини С. Введение в математику. М., Изд. иностр. лит., 1957.
77. Коваленко И. Н. Об условии независимости вероятностей состояний системы обслуживания от вида распределения времени обслуживания.— В сб.: «Проблемы передачи информации». Вып. II. 1962.
78. Колин К. К., Липаев В. В. Проектирование алгоритмов управляющих ЦВМ. М., «Советское радио», 1970.
79. Колмогоров А. Н., Успенский В. А. К определению алгоритма.— «Успехи математических наук», 1958, № 4 (82).
80. Корбут А. А., Финкельштейн Ю. Ю. Дискретное программирование. М., «Наука», 1969.
81. Кофман А., Крюон Р. Массовое обслуживание. Теория и приложения. М., «Мир», 1965.
82. Красовский А. А., Поспелов Г. С. Основы автоматики и технической кибернетики. М., Госэнергоиздат, 1962.
83. Криницкий Н. А. Язык логических схем.— В сб.: «Цифровая вычислительная техника и программирование». Вып. 1. М., «Советское радио», 1966.
84. Кузьмин И. В. Оценка эффективности и оптимизация АСКУ. М., «Советское радио», 1971.
85. Кузьмин И. В. Теоретические основы информационной техники. Харьков, Изд. ХВКИУ, 1969.
86. Кулик В. Т. Алгоритмизация объектов управления. Киев, «Наукова думка», 1968.
87. Кулик В. Т. Принципы алгоритмизации и построения управляющих машин. Киев, Гостехиздат УССР, 1963.
88. Кулик В. Т. Цифровое моделирование сложных систем. Киев, Изд. КГУ, 1964.
89. Кухтенко А. И. Основные направления в развитии систем автоматического управления.— В сб.: «Сложные системы». Вып. 4. Киев, 1968.
90. Кухтенко А. И. Проблемы инвариантности в автоматике. Киев, Гостехиздат УССР, 1963.
91. Лебедев В. И. Об итерационных методах решения операторных уравнений со спектром, лежащим на нескольких отрезках.— «Журнал вычислительной математики и математической физики», 1969, № 6.
92. Лебедев В. И., Финогенов С. А. Об одном алгоритме выбора параметров в чебышевских циклических методах.— В сб.: «Вычислительные методы линейной алгебры». Новосибирск, Изд-во СО АН СССР, 1972.
93. Лебедев В. И., Финогенов С. А. О порядке выбора итерационных параметров в чебышевском методе.— «Журнал вычислительной математики и математической физики», 1971, № 2.
94. Липаев В. В. и др. Математическое обеспечение управляющих ЦВМ. М., «Советское радио», 1972.
95. Люстерник Л. А. и др. Математический анализ. Вычисление элементарных функций. М., Физматгиз, 1963.
96. Ляпунов А. А. О логических схемах программ.— В сб.: «Проблемы кибернетики». Вып. 1. М., Физматгиз, 1958.

97. Майоров С. А., Новиков Г. И. Малогабаритные вычислительные машины. Л., «Машиностроение», 1967.
98. Мак-Кракен Д. Д. Программирование для ЦВМ. М., Изд. иностр. лит., 1960.
99. Малиновский Б. Н. и др. Основы проектирования управляющих машин промышленного назначения. М., «Машиностроение», 1969.
100. Марков А. А. Теория алгоритмов. Труды математического института им. В. А. Стеклова. Т. 42. Изд-во АН СССР, 1954.
101. Мартынюк В. В. Выделение цепей в схеме алгоритма.— «Журнал вычислительной математики и математической физики», 1961, № 5.
102. Мартынюк В. В. Об экономном распределении памяти.— «Журнал вычислительной математики и математической физики», 1962, № 3.
103. Мартынюк В. В. О некоторых методах анализа операторных схем. Автореферат кандидатской диссертации. М., Изд-во МГУ, 1963.
104. Марчук Г. И. Методы расчета ядерных реакторов. М., Атомиздат, 1961.
105. Марчук Г. И., Атанбаев С. А. Некоторые вопросы «глобальной» регуляризации. М., ДАН СССР. Т. 190, 1970, № 3.
106. Марчук Г. И., Кузнецов Ю. А. К вопросу об оптимальных итерационных процессах. М., ДАН СССР. Т. 181, 1968, № 6.
107. Марчук Г. И., Лебедев В. И. Численные методы в теории переноса нейтронов. М., Атомиздат, 1971.
108. Марчук Г. И., Сарбасов К. Е. Об одном методе решения стационарной задачи. М., ДАН СССР. Т. 182, 1968, № 1.
109. Мебуке Б. К. Задача оптимального распределения приоритетов в одноканальной системе массового обслуживания с потерями. Тбилиси, Изд-во АН ГССР, 1967.
110. Михалевич В. С. Последовательные алгоритмы оптимизации и их применение.— «Кибернетика», 1965, № 1, 2.
111. Михалевич В. С., Шор Н. З. Численное решение многовариантных задач по методу последовательного анализа вариантов.— В сб.: «Научно-методические материалы семинара АН СССР». Т. 1. М., 1962.
112. Мунтян Э. Анализ графсхемных алгоритмов. Автореферат кандидатской диссертации. М., Изд-во МГУ, 1964.
113. Новиков О. А., Петухов С. И. Прикладные вопросы теории массового обслуживания. М., «Советское радио», 1969.
114. Петер Р. Рекурсивные функции. М., Изд. иностр. лит., 1954.
115. Подловченко Р. И. Об основных понятиях программирования.— В сб.: «Проблемы кибернетики». Вып. 1. М., Физматгиз, 1958.
116. Подловченко Р. И. Об основных понятиях программирования.— В сб.: «Проблемы кибернетики». Вып. 3. М., Физматгиз, 1960.
117. Поспелов Д. А. Введение в теорию вычислительных систем. М., «Советское радио», 1971.
118. Поспелов Д. А. Классификация структур алгоритмов, реализуемых на вычислительных системах.— «Известия АН СССР. Техническая кибернетика», 1967, № 5.

119. Поспелов Д. А. Оценка производительности вычислительной системы, состоящей из обычных вычислительных машин.— «Известия АН СССР. Техническая кибернетика», 1966, № 5.
120. Прангишвили И. В. и др. Микроэлектроника и однородные структуры для построения логических и вычислительных устройств. М., «Наука», 1967.
121. Пятибратов А. П., Мачулин В. В. Эффективность систем обработки информации. М., «Советское радио», 1972.
122. Раков Г. К. Выработка случайных величин на быстродействующих счетных машинах.— В сб.: «Автоматическое управление и вычислительная техника». Киев, Гостехиздат УССР, 1958.
123. Растригин Л. А. Случайный поиск в задачах оптимизации многопараметрических систем. Рига, «Зинатне», 1965.
124. Риордан Д. Ж. Вероятностные системы обслуживания. М., Связьиздат, 1966.
125. Рухман Е. А. Сравнение отдельных методов приема цифровой информации в спорадической системе.— «Известия ЛЭТИ». Вып. 68. Ленинград, Изд. ЛЭТИ, 1968.
126. Саати Т. Л. Элементы теории массового обслуживания и ее приложения. М., «Советское радио», 1965.
127. Самарский А. А. Лекции по теории разностных схем. М., Изд-во АН СССР, 1969.
128. Севастьянов Б. А. Эргодическая теорема для марковских процессов и ее приложение к телефонным системам с отказами.— В сб.: «Теория вероятностей и ее применение». Т. 2. Вып. 1. М., 1957.
129. Слободянюк Т. Ф. и др. Алгоритмы обработки чисел, превышающих разрядность вычислительной машины.— «Автоматика», 1967, № 3.
130. Соболь И. М. Псевдослучайные числа для машины «Стрела».— В сб.: «Теория вероятностей и ее применение». Т. 7. Вып. 3. М., 1962.
131. Солзер Дж. Частотный анализ ЦВМ, работающих в реальном времени.— В сб.: «Частотные методы в автоматике». М., Изд. иностр. лит., 1957.
132. Тимофеев Б. Б. и др. Алгоритмизация в автоматизированных системах управления. Киев, «Техніка», 1972.
133. Тихонов А. Н. Об устойчивости алгоритмов для решения вырожденных систем линейных алгебраических уравнений.— «Журнал вычислительной математики и математической физики», 1965, № 4.
134. Тихонов А. Н. Об устойчивости задачи оптимизации функционалов.— «Журнал вычислительной математики и математической физики», 1966, № 4.
135. Тихонов А. Н. О решении некорректно поставленных задач и методе регуляризации. М., ДАН СССР. Т. 151, 1963, № 3.
136. Тихонов А. Н. и др. Сборник задач по математической физике. М., Физматгиз, 1972.
137. Успенский В. А. Лекции о вычислимых функциях. М., Физматгиз, 1960.
138. Фрай Т. Теория вероятностей для инженеров. М., Гостехиздат, 1934.
139. Хетагуров Я. А. и др. Основы инженерного проектирования УЦВМ. М., «Советское радио», 1972.
140. Хинчин А. Я. Работы по математической теории массового обслуживания. М., Физматгиз, 1963.

141. Цыпкин Я. З. Теория линейных импульсных систем. М., Физматгиз, 1963.
142. Чумаченко В. Ф., Попов В. А. Эффективность логических структур специализированных вычислительных машин. Харьков, Изд. ХВКИУ, 1970.
143. Шилейко А. В. Цифровые модели.— «Автоматика и телемеханика», 1959, № 12.
144. Шилейко А. В., Козырева Л. М. О структурах специализированных ЦВМ.— В сб.: «Комбинированные вычислительные машины». М., Изд-во АН СССР, 1962.
145. Шилов В. И. и др. Массовое обслуживание и синтез вычислительных систем. Харьков, Изд. ХВКИУ, 1969.
146. Шилова Г. А. Анализ производительности системы массового обслуживания, являющейся моделью ЦВМ с виртуальной памятью.— В сб.: «Вычислительная техника в системах управления летательными аппаратами». Вып. 182. М., Изд. МАИ, 1968.
147. Шкурба В. В. и др. Задачи календарного планирования и методы их решения. Киев, «Наукова думка», 1966.
148. Шура-Бура М. Р. Решение математических задач на автоматических цифровых машинах.— В сб.: «Программирование для быстродействующих счетных машин». М., Изд-во АН СССР, 1952.
149. Шура-Бура М. Р. Система стандартных подпрограмм. М., Физматгиз, 1958.
150. Юдин Д. Б., Гольштейн Е. Г. Задачи и методы линейного программирования. М., «Советское радио», 1964.
151. Юрченко В. Е. Одна задача определения объема оперативной памяти УВС.— В сб.: «Массовое обслуживание в системах передачи информации». М., «Наука», 1969.
152. Ющенко Е. Л. Адресное программирование. Киев, Гостехиздат УССР, 1963.
153. Янов Ю. И. О логических схемах алгоритмов.— В сб.: «Проблемы кибернетики». Вып. 1. М., Физматгиз, 1958.
154. Янов Ю. И. О преобразовании логических схем программ.— «Известия вузов СССР. Радиофизика», 1968, № 1.
155. Church A. An unsolvable problem of elementary number theory. Amer. J. Math. 58, 1936.
156. Church A. A note on the Entscheidungsproblem. J. Symbolie 1, 1936.
157. Church A. The Calculi of Lambdaconversion. Princeton, N. J., 1941.
158. Gödel K. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. Math und Phys., 38, 1931.
159. Post E. Finite combinatory processesformulation 1. J. Symbolie Lodic, 1, 1936.
160. Turing A. Computable numbers with, an application to Entscheidungsproblem. Proc. Lond. Math. Soc (2), T. 42, 1936.
161. Prosser R. T. Application of Boolean matrices fo the analysis of flow diagrams, Prol. Eastern. Joint Computer conf., 1959.
162. Keith M. Howell. Multiple Precision Arithmetic Techniques, The Computer Journale, 1967, N 4.
163. Takacs L. Priority Quenes Operat. Res., 1964, 12, N 1.
164. Schrage L. E., Miller L. W. The Quene, 1967 with Shortest Remaining Processing Time Dissipline. J. Oper. Res., 1966, 14, N 4.

165. Oliver R. M., Pestablozzi G. On a Problem of Optimum Priority Classification. J. Sos. Industr. and Appl. Math., 1965, 13, № 3.

166. Basic A. Dimitation on the specd of Digital Computers, IRE. Trans on Electron. Comput, 10, № 3, 1961.

167. Lamb D. Design ana applications of a digital differential analyezer Pros. IRE. Australia, № 4, 1961.

168. Steward M. H. Rote Lemiting in ineremental computers. Proc. of the Computers in Control Systems Conference, Oktober, 1957.

ОГЛАВЛЕНИЕ

Иван Васильевич Кузьмин, докт. техн. наук,
Березюк Николай Тимофеевич, докт. техн. наук,
Фурманов Клайд Константинович, канд. техн. наук,
Шаронов Валерий Борисович, канд. техн. наук

Синтез вычислительных алгоритмов управления и контроля

Редактор издательства инж. *Н. М. Корнильева*
Обложка художника *Г. Т. Конева*
Художественный редактор *В. С. Шапошников*
Технический редактор *Н. И. Старченкова*
Корректор *Т. Е. Царинская*

Сдано в набор 19. II. 1974 г. Подписано к печати 9. XII. 1974 г. Формат бумаги 84x108$^1/_{32}$. Бумага типографская № 1. Объем: 13 усл. печ. л.; 11,91 уч.-изд. л. Тираж 8800. Зак. № 5-7. БФ 05394. Цена 1 руб. 5 коп.

Издательство «Техніка», 252601, Киев, 1, ГСП, Пушкинская, 28.

Отпечатано с матриц Головного предприятия республиканского производственного объединения «Полиграфкнига» Госкомиздата УССР, г. Киев, ул. Довженко, 3 на Харьковской книжной фабрике «Коммунист» республиканского производственного объединения «Полиграфкнига» Госкомиздата УССР, Харьков, ул. Энгельса, 11.