МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГАОУ ВО «КФУ имени В.И. Вернадского»
Физико-технический институт
Кафедра компьютерной инженерии и моделирования

Руденко М.А.

КОМПЬЮТЕРНЫЕ СИСТЕМЫ

Методические рекомендации для выполнения лабораторных работ

по направлению подготовки (специальности)

09.03.01 Информатика и вычислительная техника

Симферополь 2019

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ

Цель лабораторного практикуму по дисциплине «Компьютерные системы» является глубокое изучение методов проектирования вычислительных систем различного назначения и приобретение практических навыков в их проектировании, определение основных параметров. В процессе лабораторных работ студент должен ознакомиться со средствами построения современных ЭВМ, микропроцессорных систем, вычислительных кластеров, решить задачу выбора оптимального по критерию стоимости или производительности комплекса устройств для заданных характеристик решаемых задач и условий эксплуатации системы. Следует выделить следующие основные этапы проектирования КС:

Функциональное (системное) - нахождение конфигурации технических средств и выбор состава структуры программного обеспечения при соблюдении заданных ограничений на стоимость, габариты и др.;

- структурное - разработка способов взаимодействия функциональных элементов друг с другом;

- логическое - распределение физических адресов в памяти системы и разработка прикладных программ, реализующих функции комплекса заданной конфигурации;

- техническое - конструктивная и электрическая компоновка комплекса, и разработка конструкторской и эксплуатационной документации.

Проектирование комплексов на модульной основе уменьшает трудоемкость технического проектирования, повышая при этом удельный вес функционального проектирования.

Необходимо знать наиболее распространенные разновидности информационно-вычислительных систем:

а) цифровые управляющие и контролирующие системы;

б) системы с оперативной обработкой;

в) мультипроцессорные вычислительные системы.

г) вычислительные кластеры.

При проектировании следует рассмотреть ряд вопросов: разработка технического задания на систему, оценка характеристик задач, решаемых системой, выбор оптимальной структуры системы, выбор подсистемы памяти, построение подсистем ввода-вывода, обслуживания пользователя, связи с объектом управления из заданной номенклатуры технических средств, оценка надежности, выбор программного обеспечения. В процессе выполнения проекта студенты приобретают навыки творческого решения инженерных задач, самостоятельного исследования проблемных вопросов.

3. ТЕМЫ ЛАБОРАТОРНЫХ РАБОТ

Расчет характеристик вычислительной системы

1. Лабораторная 1: Расчет трудоемкости алгоритма
2. Лабораторная 2: Расчет нагрузки компьютерной системы
3. Лабораторная 3: Расчет времени обслуживания заявок

Имитационное моделирование

4. Лабораторная 4: Имитационное моделирование различных режимов работы вычислительной системы.

Надежность компьютерных систем

5. Лабораторная 5: Оценка надежности системы и разработка мероприятий по ее повышению

Лабораторная работа №1

1. МЕТОДИКА РАСЧЕТА ТРУДОЕМКОСТИ ВЫПОЛНЕНИЯ АЛГОРИТМА

1.1 Теоретические основы

Трудоемкость алгоритма – количество вычислительной работы, требуемой для реализации алгоритма.

В задачах оценки трудоемкости операторы алгоритма разделяют на функциональные, перехода и ввода-вывода.

Функциональный оператор задает преобразование на множестве данных, т.е. задает некоторую совокупность вычислительных операций. *Оператор перехода* задает порядок вычисления функциональных операторов и правило выбора одного из возможных путей развития вычислительного процесса, соответствующего текущим значениям данных. *Оператор ввода-вывода* задает обращение к определенному файлу с целью передачи некоторого количества информации. Функциональные операторы и операторы перехода называют *основными операторами*.

Совокупность операторов и связей между ними наиболее наглядно представляется графом алгоритма, где вершины соответствуют основным операторам алгоритма, а дуги отображают связи между операторами.

Вершины графа бывают трех типов: начальная, конечная и операторная. *Начальная* вершина определяет начало алгоритма. Эта вершина не имеет ни одного входа и имеет только один выход. *Конечная* вершина определяет конец алгоритма и имеет не менее одного входа и ни одного выхода. *Операторная* вершина соответствует основному оператору или оператору ввода-вывода. Если она представляет функциональный оператор или ввода-вывода, то может иметь не меньше одного входа и только один выход. Если она представляет оператор перехода, то может иметь не меньше одного входа и не меньше двух выходов.

Граф алгоритма является корректным, если выполняются условия:

- имеется только одна начальная и только одна конечная вершина;

- для каждой вершины кроме начальной существует по крайней мере один путь, ведущий в эту вершину из начальной;

- из каждой вершины кроме конечной существует по крайней мере один путь, ведущий из этой вершины в конечную;

- выход из любой вершины должен вести только к одной вершине графа;

- при любых значениях логических условий существует путь из начальной вершины в конечную, причем любому фиксированному набору значений условий соответствует только один такой путь.

Пример графа алгоритма приведен на рис. 5.1.

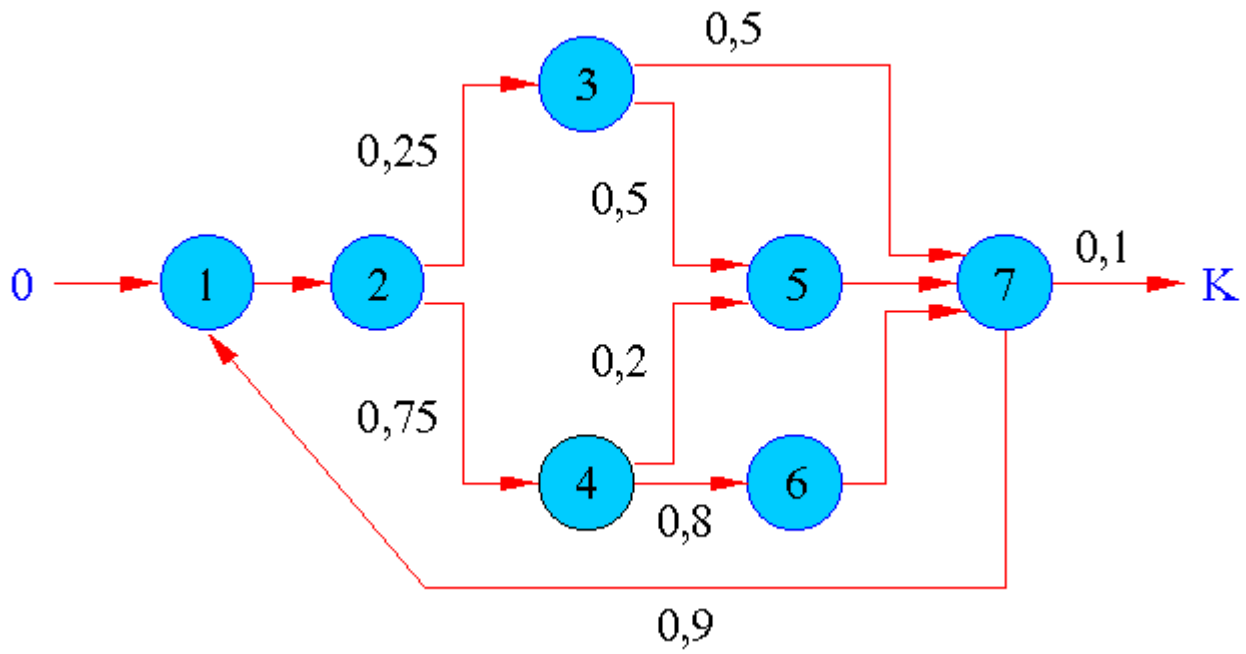


Рис. 5.1. Пример представления графа алгоритма

Вершины графа обозначены номерами $0,1,...,\mathrm{K}$; (0 - начальная вершина, K - конечная; 1,...,K-1 идентифицируют операторы алгоритма). Граф выявляет структуру алгоритма, определяя множество операторов $V=\{V_1,...,V_{K-1}\}$ и дуг $D=\{(i,j)\}$, $\mathrm{i}=0,...,\mathrm{K}-1$ и $\mathrm{j}=1,...,\mathrm{K}$, связывающих операторы.

Для оценки трудоемкости алгоритма необходимо:

1. разбить множество операторов на классы:
   - основных операторов $S_0=\{V_{\alpha_1},...,V_{\alpha_m}\}$, $V_k \in V$,
   - операторов ввода-вывода $S_h=\{V_{\beta_1},...,V_{\beta_l}\}$
2. для каждого основного оператора V_α определить количество операций k_α, составляющих оператор и для каждого оператора ввода-вывода V_β - среднее количество информации l_β, передаваемой при выполнении оператора;
3. переходы между операторами V_i и V_j следует рассматривать как случайные события и характеризовать их вероятностями p_{ij}, т.е. каждая дуга графа алгоритма отмечается вероятностью перехода p_{ij}, с которой переход из вершины V_i к V_j выполняется именно по этой дуге.

Так как вычислительный процесс не может приостановиться в вершине V_i, то с вероятностью 1 произойдет переход к какой-либо вершине графа алгоритма. Поэтому вероятности переходов должны отвечать условию:

$$\sum_{j=1}^{K} p_{ij} = 1, (i = 0,..., K-1); \tag{5.1}$$

Значения p зависят от вероятностей выполнения условия, проверяемого оператором i с целью выбора пути перехода.

Пусть $n_1,..., n_{K-1}$ - среднее число обращений к операторам $V_1,..., V_{K-1}$ за один прогон алгоритма. Тогда характеристика трудоемкости может быть вычислена следующим образом:

среднее число операций, выполняемых при одном прогоне алгоритма

$$\theta = \sum_{V_i \in S_0} n_i k_i \tag{5.2}$$

где S_0 - множество операторных вершин рассматриваемого графа алгоритма; k_i -количество операций, порождаемых оператором i.

В связи с тем, что вероятности переходов p_{ij} постоянны и после выполнения оператора V_i $(i = 0,..., K-1)$ переход к следующему оператору определяется только распределением вероятностей p_{ij}, т.е. не зависит от хода вычислительного процесса в прошлом, то процесс выполнения алгоритма является марковским процессом с K состояниями $S_1,..., S_K$, соответствующими пребыванию процесса в вершинах $V_1,..., V_K$ графа алгоритма.

Состояния $S_1,..., S_{K-1}$ - невозвратные (процесс после какого-то числа шагов покидает их), состояние S_K - поглощающее (в нем процесс прекращается). Начальное состояние S_0 определяется дугой $(0, i)$, выходящей из начальной вершины графа. Граф алгоритма, дуги которого отмечены вероятностями переходов p_{ij}, рассматривают как граф марковской цепи (марковская цепь – марковский процесс с дискретными состояниями).

Решение задачи определения трудоемкости алгоритма сводится к вычислению среднего числа $n_1,..., n_{K-1}$ пребываний марковского процесса в невозвратных состояниях $S_1,..., S_{K-1}$. Одним из способов расчета указанных величин является нахождение корней системы линейных алгебраических уравнений

$$n_i = \delta_{1i} + \sum_{j=1}^{K-1} p_{ji} n_j \ \ (i = 1,..., K-1), \tag{5.3}$$

где δ_{1i} - символ Кронекера (δ_{ij} = 1 при i = j и δ_{ij} = 0 при $i \neq j$).

Каноническая запись системы уравнений (5.3) имеет вид

$$\begin{cases} (p_{11}-1)n_1 + p_{21}n_2 + ... + p_{K-1,1}n_{K-1} = -1 \\ p_{12}n_1 + (p_{22}-1)n_2 + ... + p_{K-1,2}n_{K-1} = 0 \\ \vdots \\ p_{1,K-1}n_1 + p_{2,K-1}n_2 + ... + (p_{K-1,K-1}-1)n_{K-1} = 0 \end{cases}. \tag{5.4}$$

Решив систему (5.4), определяем среднее число попаданий $n_1,...,n_{K-1}$ вычислительного процесса в состояния $S_1,...,S_{K-1}$. При заданных значениях $k_1,...,k_{K-1}$ т.е. среднем количестве операций, порождаемых каждым функциональным оператором $S_1,...,S_{K-1}$, по формуле (5.2) можно вычислить трудоемкость алгоритма, характеризующую его как среднее число операций, выполняемых при одном прогоне.

Рассмотренный способ определения трудоемкости алгоритмов является ***универсальным***, позволяя получать оценки для алгоритмов с любой структурой, но требует решения системы линейных алгебраических уравнений высокого порядка. Поэтому для выполнения необходимых расчетов этот способ предполагает использование ЭВМ.

Вторым подходом к решению задачи расчета трудоемкости алгоритмов является сетевой метод, который позволяет найти среднюю, минимальную и максимальную величину трудоемкости. Суть метода состоит в выделении путей на графе алгоритма, соответствующих минимальной, максимальной и средней трудоемкости последовательности операторов. Эти пути могут быть выделены только на графах, не содержащих циклов. Поэтому сначала из графа алгоритма исключаются циклы путем их замены операторами с эквивалентной трудоемкостью.

Для расчета алгоритма, не содержащего циклы, необходимо:

1. пронумеровать вершины графа в порядке их следования следующим образом: начальной вершине присваивается номер 0; очередной номер $\mathrm{i}=1,2,...$ присваивается вершине, в которую входят дуги от уже пронумерованных вершин с номерами, меньшими i. Конечная вершина графа будет иметь максимальный номер K.
2. для каждой вершины можно вычислить среднее количество попаданий вычислительного процесса в эту вершину по формуле

$$n_i = \sum_{j=1}^{K} p_{ji} n_j \quad (i = 2,..., K-1) \tag{5.5}$$

где p_{ji} - вероятность перехода из вершины j в вершину i.

При установленном порядке нумерации вершин на момент вычисления n_i значения $n_1,...,n_{i-1}$ уже определены. Поэтому вычисление значения $\mathrm{n_i}$ сводится к суммированию произведений, причем поскольку $p_{ji} = 0$ для всех $j \geq i$, то суммирование следует проводить только для $j < i$.

Характеристика трудоемкости вычисляется по формуле (5.2).

Для расчета алгоритма, содержащего циклы, необходимо исключить циклы, заменяя их операторами с эквивалентной трудоемкостью. Для этого разделяют циклы по рангам. К рангу 1 относятся циклы, которые не содержат внутри себя ни одного цикла; к рангу 2 - циклы, внутри

которых есть циклы не выше ранга 1, и т.д. Совокупность операторов, входящих в цикл, и связывающих их дуг, за исключением дуги, замыкающей цикл, называют телом цикла. Тело цикла ранга 1 является графом без циклов. Применяя к этому графу вышеописанную методику, можно определить значения n_i для каждого из операторов, принадлежащих телу цикла; тогда трудоемкость тела цикла С равна

$$\sum_{V_j \in C} k_j n_j ,$$

где V_j - вершины, содержащиеся в цикле С.

Определим среднее число повторений цикла n_C, равное числу выполнений тела цикла при одном прогоне алгоритма. Если вероятность перехода по дуге, замыкающей цикл, равна p_{kl}, то $n_C = 1 + p_{kl} n_C$, откуда

$$n_C = \frac{1}{1 - p_{kl}}. \tag{5.6}$$

Тогда средняя трудоемкость цикла

$$k_C = n_C \sum_{V_j \in C} k_j n_j \tag{5.7}$$

и цикл С можно заменить оператором с трудоемкостью k_C. Применяя указанную процедуру замены циклов операторами ко всем циклам ранга 1, затем к циклам ранга 2 и т. д., приходим к графу без циклов, трудоемкость которого находится вышеописанным способом.

Минимально и максимально возможное значения трудоемкости на момент окончания выполнения оператора V_i обозначим соответственно через A_i и B_i. Для начальной вершины имеем $A_0 = 0$ и $B_0 = 0$. Тогда для остальных вершин с номерами $i = 1,2,...k$

$$A_i = \min_{(j,i) \in D} (A_j) + k_{i\,\min}, \tag{5.8}$$

$$B_i = \max_{(j,i) \in D} (B_j) + k_{i\,\max}, \tag{5.9}$$

где (j,i) - дуга, выходящая из вершины j и входящая в вершину i; D - множество дуг графа программы; минимальное $\min(A_j)$ и максимальное $\max(B_j)$ значения определяются по отношению ко всем вершинам j, из которых выходят дуги, входящие в вершину i; значения $k_{i\,\min}$ и $k_{i\,\max}$ характеризуют минимальную и максимальную трудоемкость оператора V_i.

Для конечной вершины K графа вычисляются значения

$$A_K = \min_{(j,K) \in D} (A_j), \tag{5.10}$$

$$B_K = \max_{(j,K) \in D} (B_j), \tag{5.11}$$

характеризующие минимальную и максимальную трудоемкость алгоритма.

Лабораторная работа №2

2. ОПРЕДЕЛЕНИЕ РАБОЧЕЙ НАГРУЗКИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

Задание на курсовой проект содержит неоднородное описание рабочей нагрузки, представленной пятью классами задач (см. Приложение 2). На начальных стадиях проектирования системы необходимо знать **общий объем работ**, который она должна выполнять. При этом неоднородное описание заменяется однородным, представленным характеристиками одного, усредненного класса задач. Эта операция выполняется следующим образом.

Пусть рабочая нагрузка образована M классами задач, поступающими в систему с интенсивностями $\lambda_1, \lambda_2, ..., \lambda_M$. Задачи каждого класса требуют в среднем выполнения $\Theta_1, \Theta_2, ..., \Theta_M$ процессорных операций. Они работают с файлами $F_1, F_2, ..., F_K$, которые характеризуются длиной $G_1, G_2, ..., G_K$ и длиной блоков записей $l_1, l_2, ..., l_K$. Задача с номером m обращается к файлу с номером k d_{mk} раз. Если предполагается, что система будет иметь распределенную архитектуру (типа ВСТД или сети), то должна быть известна интенсивность обмена с удаленными источниками информации. Она может быть представлена числом обращений удаленных пользователей к задачам: $q_1, q_2, ..., q_M$. Кроме того, стандартами оговаривается длина пакета сообщений Z.

Для получения однородного описания рабочей нагрузки определяют следующие характеристики средней задачи:

1) интенсивность поступления:

$$\Lambda = \sum_{m=1}^{M} \lambda_m ,$$

2) доля задач класса m в общей смеси:

$$p_m = \lambda_m / \Lambda,$$

3) трудоемкость процессорных операций:

$$\Theta = \sum_{m=1}^{M} p_m * \theta_m ,$$

4) среднее число обращений к файлу F_k:

$$D_k = \sum_{m=1}^{M} p_m * d_{mk} ,$$

общее число обращений к файлам:

$$D = \sum_{k=1}^{K} Dk ,$$

6) средняя длина блока записей файлов:

$$l_{ср\ бл} = \left(\sum_{k=1}^{K} lk * Dk \right) / D,$$

7) при наличии удаленных источников информации (пользователей) – среднее число обращений этих источников к задаче:

$$Q = \sum_{m=1}^{M} p_m * q_m ,$$

очевидно, что для сосредоточенных ВС (одной ЭВМ или комплекса) $Q = 0$;

8) среднее количество прерываний центрального процессора определяется с учетом того, что любая операция обращения к файлу F_k или к удаленному пользователю вызывает прерывание центрального процессора:

$$H_{\text{ЦПр}} = D + Q + 1,$$

9) средняя трудоемкость (количество операций) непрерывного счета на процессоре:

$$\Theta_0 = \Theta / H_{\text{ЦПр}}.$$

Вычисленные характеристики средней задачи используются для выбора базового варианта структуры ВС и его оптимизации на первых стадиях проектирования.

Лабораторная работа №3

3. ВЫБОР БЫСТРОДЕЙСТВИЯ ПРОЦЕССОРА И ДИСЦИПЛИНЫ ОБСЛУЖИВАНИЯ ПРИ СИНТЕЗЕ КС

Компьютерные система функционируют в реальном масштабе времени (согласованно с темпом поступления заявок на решение определенных задач). Это означает, что заявки должны обслуживаться системой за какое-то время, определяемое назначением КС. Выполнение ограничений на время обслуживания заявок – основная задача, возникающая при проектировании КС.

В зависимости от требований к временным характеристикам различают системы:

1) с неограниченным временем пребывания заявок;

2) с относительными ограничениями на время пребывания заявок;

3) с абсолютными ограничениями на время пребывания заявок;

В качестве критерия эффективности системы выбирается коэффициент потери качества функционирования из-за задержек обслуживания заявок, который характеризуется функцией штрафа.

Для системы с неограниченным временем пребывания заявок функция штрафа имеет вид :

$$C_W = \sum_{i=1}^{M} \alpha_i \lambda_i w_i , \qquad (2.1)$$

где α_i - штраф за задержку одной заявки типа i на единицу времени; λ_i - интенсивность потока заявок типа i; w_i - среднее время ожидания заявок типа i.

Для системы с относительными ограничениями на время пребывания заявок налагаются ограничения на средние времена пребывания (ожидания) заявок всех типов и задаются в виде неравенства:

$$w_i \le w_i^* ,$$

где w_i^* - предельное ограничение на время ожидания заявки типа i, $i = 1,..., M$.

Для системы с абсолютными ограничениями на время пребывания заявок накладываются более жесткие требования. Показателем качества функционирования таких систем при одномерном потоке заявок может служить вероятность превышения допустимого времени ожидания $\Pr(W > w_i^*)$. Чем меньше эта вероятность, тем выше качество функционирования ВС.

В случае M потоков заявок потеря качества функционирования характеризуется функцией штрафа:

$$C_p = \sum_{i=1}^{M} \alpha_i' \lambda_i \Pr(W_i > w_i^*),$$

где α_i' - штраф за превышение одной заявкой типа i допустимого ограничения.

Произведение $\alpha_i'\lambda_i \Pr(W_i > w_i^*)$ определяет штраф за превышение допустимого ограничения на время ожидания заявок типа i, поступающих в систему за единицу времени. Использование указанного критерия вызывает трудности, связанные с необходимостью нахождения вероятностей $\Pr(W_i > w_i^*)$. Они определяются, если известны законы распределения времени ожидания для различных типов заявок. Аналитический вид законов распределения может быть получен лишь для простейших дисциплин обслуживания, а в случае более сложных дисциплин обслуживания ориентируются на использование только двух первых моментов распределения. Это обстоятельство ограничивает возможность применения указанного критерия.

В процессе исследования на статистических моделях установлено, что для широкого класса дисциплин обслуживания распределение времени ожидания можно аппроксимировать выражением вида:

$$\Pr(W > w^*) = R \cdot \exp(-Rw^* / w),$$

где w - среднее время ожидания заявок в очереди; w^* - предельно допустимое время ожидания; R - суммарная загрузка процессора.

Задача синтеза КС может быть разбита на 3 этапа.

На **первом этапе** синтеза определяется нижняя оценка быстродействия процессора, обеспечивающая как минимум существование стационарного режима, а при наличии ограничений на время ожидания заявок - существование некоторой дисциплины обслуживания, удовлетворяющей этим ограничениям. В стационарном режиме, когда времена ожидания и пребывания заявок в системе имеют конечные значения, суммарная загрузка системы от всех входящих потоков должна быть меньше единицы,

$$\sum_{i=1}^{M} \lambda_i \vartheta_i < 1. \qquad (2.2)$$

Среднее время обслуживания заявки ϑ_i определяется значением :

$$\vartheta_i = \frac{\theta_i}{B}, \qquad (2.3)$$

где θ_i - трудоемкость обслуживания заявок (программ); B - среднее быстродействие процессора.

С учетом (2.2) и (2.3) для систем с неограниченным временем пребывания минимально необходимое быстродействие процессора, при котором существует стационарный режим работы системы, определяется из условия :

$$B > \sum_{i=1}^{M} \lambda_i \theta_i . \qquad (2.4)$$

При наличии относительных ограничений на время пребывания заявок минимально необходимое быстродействие определяется из условия:

$$B > 0{,}5\sum_{i=1}^{M}\lambda_i\theta_i + \left[\frac{1}{4}\sum_{i=1}^{M}\lambda_i\theta_i\left(\sum_{i=1}^{M}\lambda_i\theta_i + 2\sum_{i=1}^{M}\lambda_i\theta_i^{(2)} \Big/ \sum_{i=1}^{M}\lambda_i\theta_i w_i^*\right)\right]^{0,5}, \qquad (2.5)$$

где $\theta^{(2)}$ - второй начальный момент трудоемкости $\theta^{(2)} = 2\theta^2$.

При $w^* \to \infty$ выражение (2.5) совпадает с выражением (2.4).

На **втором этапе** выбирается дисциплина обслуживания заявок, обеспечивающая минимум значения критерия эффективности:

$$C_W = \sum_{i=1}^{M}\alpha_i\lambda_i w_i ,$$

где произведение $\alpha_i\lambda_i w_i$ определяет штраф за задержку обслуживания заявок i - го типа, поступающих в систему за единицу времени; C_W - штраф за задержку в обслуживании всех заявок.

Если штрафы для всех заявок одинаковы, то

$$C_W = \sum_{i=1}^{M}\lambda_i w_i .$$

Задержки, т.е. времена ожидания $w_1,...,w_M$ зависят от двух факторов: быстродействия процессора B и дисциплины обслуживания заявок. Увеличение быстродействия приводит к уменьшению всех значений, $w_1,...,w_M$ а следовательно, к уменьшению величины C_W. Максимальная эффективность системы достигается при бесконечном быстродействии процессора, что свидетельствует о некорректности использования критерия 2.1 для выбора быстродействия процессора. Этот критерий может быть использован в задаче выбора дисциплины обслуживания. Дисциплина обслуживания считается оптимальной для определенной системы, если ей соответствует минимальное значение C_W по сравнению с другими дисциплинами обслуживания.

Задача оптимального назначения приоритетов сводится к расположению приоритетов в порядке убывания величины

$$\alpha_i \Big/ \vartheta_i > \alpha_{i+1} \Big/ \vartheta_{i+1}, (i = 1,...,M-1). \qquad (2.6)$$

Выражение (2.6) - условие выигрыша от введения относительных или абсолютных приоритетов в сравнении с бесприоритетными дисциплинами обслуживания.

В случае равных коэффициентов α_i в выражении (2.6) получим условие $J_i < J_{i+1}$ выигрыша при использовании дисциплин обслуживания с относительными приоритетами: заявкам с меньшей длительностью обслуживания должны присваиваться более высокие

относительные приоритеты $\vartheta_i < \vartheta_{i+1}$. Для дисциплин обслуживания с абсолютными приоритетами приоритеты должны назначаться в порядке, обратном номерам потоков заявок.

На **третьем этапе** определяется оптимальное быстродействие процессора. Для выбранной оптимальной дисциплины обслуживания необходимо уточнить быстродействие процессора, обеспечивающее заданное качество обслуживания заявок при минимально возможном простое процессора.

В качестве критерия эффективности при таком подходе может быть использован функционал вида

$$C_B = \beta_0 \eta(B) + \sum_{i=1}^{M} \beta_i \lambda_i w_i(B), \tag{2.7}$$

где η - коэффициент простоя; β_i - некоторые весовые коэффициенты, задаваемые при проектировании системы. Задание весовых коэффициентов должно осуществляться исходя из назначения системы и требований, предъявляемых к ней.

В системах с жесткими требованиями ко времени реакции должны присваиваться более высокие значения коэффициентов β_i, а системы, основное требование к которым - минимум материальных затрат, должны иметь наибольшее значение β_0.

Функция C_B имеет минимум, которому соответствует оптимальное значение быстродействия процессора B_{opt}. Для системы с неограниченным временем пребывания заявок оно равно

$$B_{opt} = L + 0{,}5L^{-1}\left\{k\Lambda L^{(2)} + \left[(2L^2 + k\Lambda L^{(2)})k\Lambda L^{(2)}\right]^{0,5}\right\}, \tag{2.8}$$

где k - некоторый коэффициент пропорциональности, равный

$$k = \frac{\beta_1}{\beta_0};$$

$$L = \sum_{i=1}^{M} \lambda_i \theta_i;$$

$$\Lambda = \sum_{i=1}^{M} \lambda_i;$$

$$L^{(2)} = \sum_{i=1}^{M} \lambda_i \theta_i^{(2)}.$$

Для системы, к которой не предъявляются никакие требования на время пребывания заявок, кроме требований, чтобы все заявки были обслужены за конечное время и для которых $k = 0$, оптимальное значение B_{opt} совпадает с нижней оценкой быстродействия процессора. Это означает, что быстродействие процессора для таких систем должно выбираться только из условия максимальной загрузки системы. Но чем выше требования, предъявляемые к качеству

обслуживания заявок, т.е. чем больше k, тем более высокое быстродействие процессора должно быть выбрано по сравнению с нижней оценкой.

Для систем с относительными ограничениями на время пребывания заявок, кроме ограничений на время ожидания заявок, задается ограничение на коэффициент простоя процессора η^* и задача определения оптимального быстродействия формулируется следующим образом: найти такое значение быстродействия B, которое обеспечивает минимум критерия эффективности C_B (2.7) при ограничениях:

$$\eta \le \eta^*;\ w_i \le w_i^*;\ (i = 1,...,M);\ B > 0.$$

При использовании системы с бесприоритетной дисциплиной обслуживания и относительными ограничениями на время пребывания заявок оптимальное быстродействие процессора определяется на основе решения неравенства:

$$\frac{L}{2} + \left[\frac{L^{(2)}}{4} + \frac{L^{(2)}}{2w^*}\right]^{0,5} \le B \le \frac{1}{1-\eta^*},$$

где

$$L = \sum_{i=1}^{M} \lambda_i \theta_i;\ L^{(2)} = \sum_{i=1}^{M} \lambda_i \theta_i^{(2)};$$

w^* - среднее предельно допустимое время ожидания заявок;

η^* - ограничение на коэффициент простоя процессора.

Решением этой системы неравенств является область S допустимых значений быстродействия процессора B (рис. 2.1). Верхней границе B соответствует зависимость $B = B(\eta^*)$ нижней - зависимость $B = B(w^*)$.

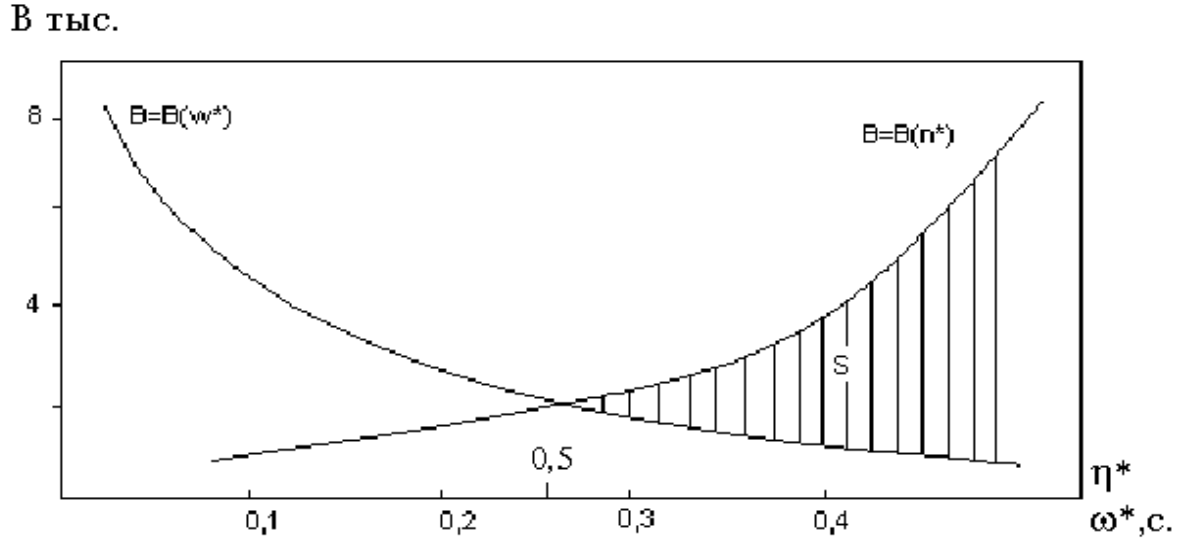


Рис. 2.1 - Область допустимых значений быстродействия процессора

В случае абсолютных ограничений на время пребывания заявок к настоящему времени аналитических зависимостей определения оптимального быстродействия не получено, поскольку задача сводится к решению трансцендентных уравнений.

После определения быстродействия процессора возникает необходимость уточнить время ожидания потоков заявок.

В некоторых системах необходимо выполнить жесткие ограничения на время ожидания отдельных заявок, поэтому им присваиваются абсолютные приоритеты. Чтобы выполнить ограничения по всем видам заявок, другим из оставшихся присваивают относительные приоритеты, а остальные заявки обслуживаются без приоритетов. Такая дисциплина обслуживания называется смешанной. При смешанной дисциплине обслуживания время ожидания заявок можно определить:

$$w_k = \begin{cases} \dfrac{R_{k-1}\vartheta_k}{1-R_{k-1}} + \dfrac{\sum_{i=1}^{k}\lambda_i\vartheta_i^{(2)}}{2(1-R_{k-1})(1-R_k)} & (k=1,...,M_1) \\ \dfrac{R_{M_1}\vartheta_k}{1-R_{M_1}} + \dfrac{\sum_{i=1}^{M}\lambda_i\vartheta_i^{(2)}}{2(1-R_{k-1})(1-R_k)} & (k=M_1+1,...,M_1+M_2), \\ \dfrac{R_{M_1}\vartheta_k}{1-R_{M_1}} + \dfrac{\sum_{i=1}^{M}\lambda_i\vartheta_i^{(2)}}{2(1-R_{M_1+M_2})(1-R)} & (k=M_1+M_2+1,...,M) \end{cases} \quad (2.9)$$

где R_k - загрузка со стороны заявок более высокого приоритета, включая рассматриваемую заявку типа k ($R_0=0$; $R_{M_1+M_2}=\sum_{i=1}^{M_1+M_2}\rho_i$; $R=\sum_{i=1}^{M}\rho_i$); заявкам типа $1,...,M_1$ присвоены абсолютные приоритеты; заявкам $M_1+1,...,M_1+M_2$ - относительные приоритеты; заявки $M_1+M_2+1,...,M$ обслуживаются по бесприоритетной дисциплине.

Задание

1. Найти минимальное значение быстродействия при котором существует стационарный режим обработки заданий.

2. Рассчитать оптимальное быстродействие процессора для КС. Режим обработки – отсутствие ограничений на время ожидания заявок. Коэффициент k выбирается произвольно ($k>0$).

3. Определить времена ожидания заявок в очереди для потоков с входными данными по вариантам из Приложение 2.

Лабораторная работа №4

4. ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СИСТЕМ С УСТРОЙСТВАМИ В РЕЖИМАХ ПРЕРЫВАНИЯ И НЕДОСТУПНОСТИ

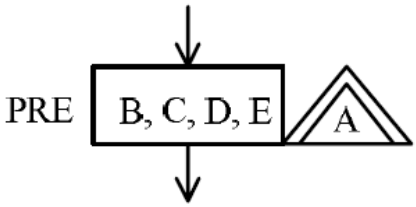
В этом разделе рассматриваются задачи, моделирующие ситуации, когда прерывается обслуживание заявок (транзактов) в устройствах, а также средства, позволяющие сэкономить машинное время при поиске заблокированных заявок.

4.1. Моделирование захвата устройств

Для моделирования захвата и освобождения устройств используются блоки PREEMPT (захватить) и RETURN (возвратить).

Формат и графическое изображение блока PREEMPT:

PREEMPT A, [B], [C], [D], [E]



В операнде A указывается имя устройства, подлежащего захвату.

Блок PREEMPT может работать или в приоритетном режиме, или в режиме прерывания. В первом случае в операнде B указывается PR, во втором этот операнд задается по умолчанию.

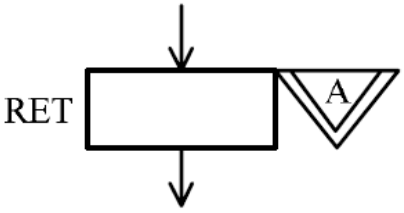
В операнде C указывается метка, куда направляется прерванный транзакт.

В операнде D указывается номер параметра прерванного транзакта, в котором записывается оставшееся время обслуживания.

Операнд E определяет право на дообслуживание: по умолчанию – сохраняется право, при указании RE – не сохраняется.

Блок RETURN фиксирует факт освобождения устройства от захвата. Его формат и изображение:

RETURN A



В операнде A указывается имя освобожденного устройства.

В модели устройство может быть захвачено любое количество раз различными транзактами, но не два раза подряд одним транзактом. Транзакт не может войти в блок, если в приоритетном режиме устройство захвачено транзактом с равным или более высоким приоритетом, чем активный транзакт.

Рассмотрим примеры использования блока PREEMPT:

PREEMPT OTO

PREEMPT OTO, PR

PREEMPT OTO, PR, MET, , RE

В первом случае блок работает в режиме прерывания. Прерванный транзакт никуда не направляется, после обслуживания захватчика будет дообслужен.

В последнем случае прерванный транзакт теряет право на дообслуживание, поэтому по метке MET его можно направить, например, в блок TERMINATE, т.е. на удаление из модели. Этого нельзя было сделать, если в последнем случае операнд E задать по умолчанию. Более подробно различные варианты моделирования захвата изложены в [1].

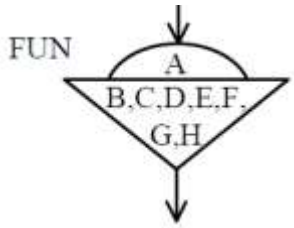
8.2. Моделирование недоступности устройств

При моделировании неисправностей устройств и реализации различных дисциплин обслуживания могут быть использованы блоки FUNAVAIL и FAVAIL.

Блок FUNAVAIL (F обозначает устройство, UNAVAIL – недоступно) моделирует недоступность одноканального устройства.

Формат и изображение блока:

FUNAVAIL A, [B], [C], [D], [E], [F], [G], [H]



В операнде A указывается имя устройства, которое становится недоступным.

Назначение остальных операндов зависит от характера транзактов, занимавших устройство до перевода его в недоступное состояние.

Операнды B, C, D соответствуют транзактам, занимавшим устройство после входа в него через SEIZE и PREEMPT .

Операнды E, F соответствуют прерванным транзактам.

Операнды G, H соответствуют транзактам, находящимся в списке задержки.

В операнде B задаются режимы работы с транзактами в период недоступности:

- CO (continue – продолжение) – обслуживание продолжается;
- RE (remove – удаление) – транзакт удаляется по адресу, указанному в операнде C;
- по умолчанию – обработка прерывается, будет продолжена, когда устройство станет доступным.

В операнде D задается номер параметра транзакта, занимавшего устройство, в который записывается оставшееся время обслуживания после того, как устройство стало недоступным.

В операнде E задаются режимы работы с ранее прерванными транзактами:

- CO – продолжение обслуживания;
- RE – удаление по адресу, указанному в операнде F;
- по умолчанию – прерванные ранее транзакты остаются в списке прерываний и не обслуживаются в устройстве в период его недоступности.

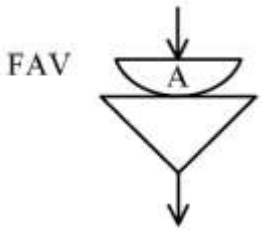
Операнд G определяет режимы работы с транзактами, находящимися в списке задержки в период недоступности устройства:

- CO – продолжение обслуживания;
- RE – удаление по адресу, указанному в операнде H;
- по умолчанию – оставление транзактов в списке задержки до момента, когда устройство станет доступным.

Блок FAVAIL (устройство доступно) моделирует доступность устройства.

Формат и изображение блока:

FAVAIL A



В операнде А указывается имя устройства, которое становится доступным.

Пусть, например, модуль моделирования аварийной ситуации имеет вид:

```
        GENERATE (NORMAL (1, 100, 10))
        FUNAVAIL OTO, CO , , , RE , MET1 , RE , MET1
        ADVANCE 5, 2
        FAVAIL OTO
MET1 TERMINATE
```

При единице модельного времени, равной 1 часу, распределение интервалов наступления аварийных ситуаций подчиняется нормальному закону при среднем значении 100 часов и стандартном отклонении 10 часов.

При переводе устройства ОТО в недоступное состояние продолжится обработка транзакта (если он находился в устройстве); транзакты, ранее прерванные, и находившиеся в списке задержки, будут удалены из модели (через MET1 TERMINATE). После восстановления устройства в течение (5 ± 2) часов оно становится доступным.

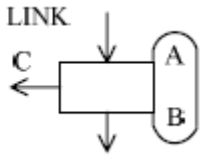
8.3. Применение списков пользователя

Списки пользователя (СП) организуются как с целью экономии машинного времени (из–за устранения потерь на просмотр СТС для поиска заблокированных транзактов), так и для реализации различных дисциплин обслуживания.

Ввод транзактов в список пользователя осуществляется блоком LINK (внести в список), вывод – блоком UNLINK (вывести из списка).

Формат и изображение блока LINK:

<имя> LINK A, B, [C]



В операнде А указывается символическое или цифровое имя списка пользователя.

Операнд В задает место списка, куда направляется транзакт.

Допустимые значения операнда В:

FIFO – в конец СП;

LIFO – в начало СП;

PR – в порядке убывания приоритета;

P – в порядке возрастания значения параметра;

M1 – в порядке возрастания относительного времени.

Операнд С указывает альтернативный выход. Если операнд С не задан (безусловный режим), то индикатор СП устанавливается в «1». Все транзакты заносятся в СП в порядке, определенном операндом В. Если операнд С задан, то проверяется индикатор СП. Если при этом индикатор в положении «1», то вошедший транзакт заносится в СП, в порядке определенном операндом В.

Ели же индикатор окажется в положении «0», то он переводится в «1», а транзакт направляется по адресу, указанному в операнде С.

Например:

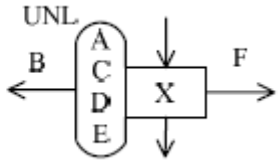
LINK FAC, FIFO

LINK DDD, M1, MET1

В первом случае транзакты присоединяются к СП по правилу FIFO. Во втором случае, если индикатор СП будет в положении «0», транзакт пойдет на метку MET1, а не в СП.

Рассмотрим блок UNILINK. Его формат и изображение:

<имя> UNLINK X A, B, C, **[D]**, **[E]**, **[F]**



В операнде A указывается символическое или цифровое имя СП.

В операнде B записывается метка, по которой направляются выведенные из СП транзакты.

В операнде C записывается число выводимых из СП транзактов или слово ALL (все), по умолчанию – ALL.

Операнды D и E вместе с операндом X задают порядок вывода транзактов из СП.

В дополнительном операнде X указываются операторы отношения (G, GE, L, LE, NE), по умолчанию E (равно). При этом сравнивается значение операнда D со значением операнда E.

В операнде D могут быть указаны булева переменная, номер параметра, слово BACK (обратно).

Если в D указана булева переменная, то операнды X и E пусты. Булева переменная вычисляется относительно транзакта из СП. Если BVj = 1, транзакт удаляется из СП, если BVj = 0 для всех транзактов СП, то вошедший транзакт пытается пройти по адресу, указанному в операнде F, если последний опущен, то в следующий оператор.

Если в D указан параметр, а операнд E отсутствует, то значение параметра вошедшего транзакта сравнивается со значением такого же параметра транзактов из СП, если E не пропущен, то со значением, указанным в операнде E.

Удаляемые транзакты направляются по адресу, указанному в операнде B.

При использовании в операнде D ключевого слова BACK (операнд E отсутствует) транзакты выводятся из конца СП.

Операнд F указывает метку, по которой направляется транзакт, вошедший в блок UNLINK, если ни один транзакт не выводится из СП.

Например, блок UNLINK ABC, MET, ALL выводит все транзакты из списка пользователя с именем ABC по правилу LIFO и направляет их по адресу MET. Блок UNLINK BBB, ALFA, 1, BACK выводит из конца списка BBB один транзакт и направляет его по адресу ALFA.

Всего из–за сочетания значений операндов X, D и E может быть восемь вариантов вывода транзактов. Они рассмотрены в [3].

3.4. Примеры

Пример 3.1. Одноканальный вычислительный стенд предназначен для обработки одной заявки(транзакта). Режим работы системы – трехсменный по 24 минуты в сутки. Безприоритетные заявки поступают на стенд по пуассоновскому закону со средним временем 8 секунд (с интенсивностью $\alpha_{бп}$), на их обслуживание уходит (3 ± 0,5) с. Заявки с относительным приоритетом, необходимые для дальнейшего использования, поступают закону со средним временем 24 секунд (с интенсивностью $\alpha_{оп}$), их обслуживание – экспоненциальное со средним временем 2 с. Заявки с относительным приоритетом могут захватить стенд, после их обслуживания фоновые изделия дообслуживаются. Через 8 с после начала работы на стенд поступают более заявки с абсолютным приоритетом и занимают стенд на 2 с.

Составить модель работы участка в течение 25 часов.

Особенности модели следующие. В модуле безприоритетные заявки транзакты занимают и освобождают стенд с помощью блоков SEIZE STEND и RELEASE STEND; в модуле заявки с относительным приоритетом транзакты захватывают и освобождают стенд с помощью блоков PREEMPT STEND (режим прерывания) и RETURN STEND; в модуле заявки с абсолютным приоритетом захват и освобождение стенда происходят с использованием блока PREEMPT STEND, PR (режим приоритетный) и RETURN STEND.

При единице модельного времени в одну секунду, модельное время составляет 25 × 3 × 8 × 60 = 36000 единиц.

Программа модели имеет вид:

```
GENERATE        (EXPONENTIAL (1, 0, 2880))
SEIZE       STEND
ADVANCE         180, 30
RELEASE         STEND
TERMINATE
GENERATE        (EXPONENTIAL (2, 0, 1440))
PREEMPT         STEND
ADVANCE         (EXPONENTIAL (3, 0, 120))
RETURN          STEND
TERMINATE
GENERATE        (EXPONENTIAL (4, 0, 1440)) , , 480, , 1
PREEMPT         STEND, PR
ADVANCE         120
RETURN          STEND
TERMINATE
GENERATE        36000
TERMINATE 1
START           1
```

Результаты моделирования показывают, что за 25 часов прошло обслуживание 13 безприоритетных, 25 с относительным приоритетом и 25 с абсолютным приоритетом заявок (по числу входов в первый, второй и третий блоки TERMINATE).

Пример 3.2. Составить модель работы системы с реализацией дисциплины обслуживания FIFO, если транзакты генерируются с интервалами от 17,5 до 22,5 единиц, распределенными по равномерному закону, а задерживаются при обработке в одноканальном устройстве по экспоненциальному закону со средним значением 10 единиц.

Программа обработки 10000 транзактов с использованием блоков LINK и UNLINK при выводе из СП по одному транзакту имеет вид:

```
        GENERATE        20, 2.5
        LINK            ALFA, FIFO, MET1
MET1 SEIZE              ROBOT
        ADVANCE         (EXPONENTIAL (1, 0, 10))
        RELEASE         ROBOT
        UNLINK          ALFA, MET1, 1
        TERMINATE 1
        START           10000
```

Лабораторная работа №5

5. РАСЧЕТ НАДЕЖНОСТИ КС

Работоспособность системы или отдельных ее частей в процессе эксплуатации может быть нарушена в результате отказа аппаратуры – выхода из строя элементов или соединений между ними. Перед разработчиком технических средств стоит задача повышения надежности создаваемой аппаратуры.

Надежность системы определяется вероятностью безотказной работы, т.е. вероятностью того, что при определенных условиях эксплуатации в заданный интервал времени не произойдет одиночного отказа.

Выражение для вычисления безотказной работы:

$$P(t) = e^{-t\sum_{i=1}^{m}\lambda_i}$$

Где t – интервал времени;

λ_i – интенсивность отказов i-го блока;

m – число блоков ВС;

Для повышения надежности ВС можно использовать резервирование ее элементов. Однако этот прием приводит к существенному увеличению стоимости системы.

В нашем случае рассчитывается вероятность безотказной работы системы с частичным контролем оборудования и профилактическими испытаниями.

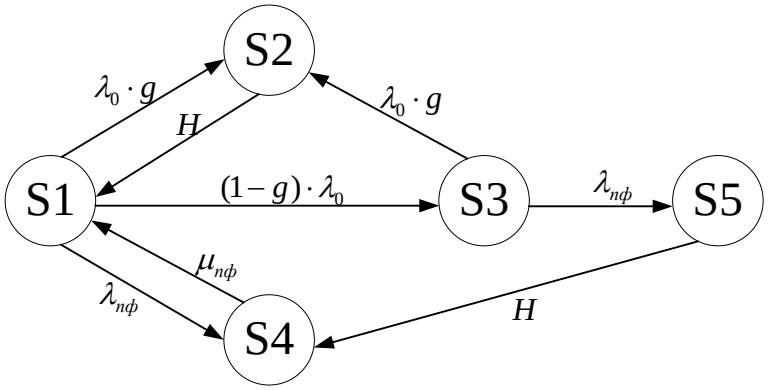


Рисунок 7.1 – Граф надежности устройства

Где S_i – состояния системы;

S_1 – система работоспособна;

S_2 – в системе обнаружен отказ;

S_3 – состояние необнаруженного отказа;

S_4 – состояние выполнения профилактических испытаний;

S_5 – в системе установлен скрытый отказ в результате профилактических испытаний.

λ_0 – интенсивность потока отказов;

$\lambda_{пф}$ – интенсивность профилактических испытаний;

$\mu_{пф}$ - интенсивность профилактики;

H – интенсивность восстановления;

g – доля контролируемого оборудования;

$\lambda_0 = 0,5$ $\lambda_{пф} = 0,65$ $\mu_{пф} = 0,5$ $H = 0,7$ $g = 0,73$.

Составим систему уравнение Колмогорова:

$$\begin{cases} -P_1\cdot(g\cdot\lambda_0 + (1-g)\cdot\lambda_0 + \lambda_{пф}) + P_2\cdot H + \mu_{пф}\cdot P_4 = 0 \\ -P_2\cdot H + g\cdot\lambda_0\cdot P_1 + g\cdot\lambda_0\cdot P_3 = 0 \\ -P_3\cdot(g\cdot\lambda_0 + \lambda_{пф}) + (1-g)\cdot\lambda_0 P_3 = 0 \\ -P_4\cdot\mu_{пф} + P_1\cdot\lambda_{пф} + P_5\cdot H = 0 \\ -P_5\cdot H + \lambda_{пф}\cdot P_3 = 0 \end{cases}$$

Найдем вероятности:

$$P_2 = \frac{P_1\cdot g\cdot\lambda_0}{H}\cdot\frac{(\lambda_0+\lambda_{пф})}{(g\cdot\lambda_0+\lambda_{пф})}$$

$$P_4 = \frac{P_1}{\mu_{пф}}\cdot\frac{(\lambda_0\cdot\lambda_{пф}+\lambda_{пф})}{g\cdot\lambda_0+\lambda_{пф}}$$

$$P_5 = \frac{P_1\cdot\lambda_0\cdot\lambda_{пф}\cdot(1-g)}{H\cdot(g\cdot\lambda_0+\lambda_{пф})}$$

$$P_3 = \frac{P_1\cdot(1-g)\cdot\lambda_0}{g\cdot\lambda_0+\lambda_{пф}}$$

P_1 определяет стационарную вероятность нахождения ВС в состоянии S_1.

P1 = 1 –P2-P3-P4-P5 =

Для расчета надежности может быть использована программа DIFUR1.

Машинный расчет

	S1	S2	S3	S4	S5
S1 0,0		0,36	0,14	0,65	
S2 1,0	0,70				
S3 0,0		0,36			0,65
S4 0,0	0,50				
S5 0,0				0,70	

Рисунок В.1 – Матрица переходов

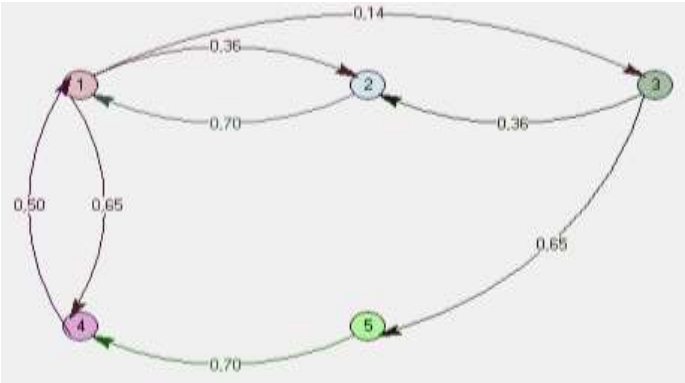


Рисунок В.2 – Граф переходов

Шаг	T	P1	P2	P3	P4	P5
82	8,20	0,3000	0,1762	0,0416	0,4434	0,0388
83	8,30	0,3000	0,1761	0,0416	0,4434	0,0388
84	8,40	0,3000	0,1761	0,0416	0,4435	0,0388
85	8,50	0,3000	0,1761	0,0416	0,4435	0,0388
86	8,60	0,3000	0,1760	0,0416	0,4436	0,0388
87	8,70	0,3000	0,1760	0,0416	0,4436	0,0388
88	8,80	0,3000	0,1760	0,0416	0,4436	0,0388
89	8,90	0,3000	0,1760	0,0416	0,4437	0,0388
90	9,00	0,3000	0,1760	0,0416	0,4437	0,0388
91	9,10	0,3000	0,1759	0,0416	0,4437	0,0387
92	9,20	0,3000	0,1759	0,0416	0,4438	0,0387
93	9,30	0,3000	0,1759	0,0416	0,4438	0,0387
94	9,40	0,3000	0,1759	0,0416	0,4438	0,0387
95	9,50	0,3000	0,1759	0,0416	0,4438	0,0387
96	9,60	0,3000	0,1759	0,0416	0,4438	0,0387
97	9,70	0,3000	0,1758	0,0416	0,4439	0,0387
98	9,80	0,3000	0,1758	0,0416	0,4439	0,0387
99	9,90	0,3000	0,1758	0,0416	0,4439	0,0387
100	10,00	0,3000	0,1758	0,0416	0,4439	0,0387

Рисунок В.3 – Вероятность переходов

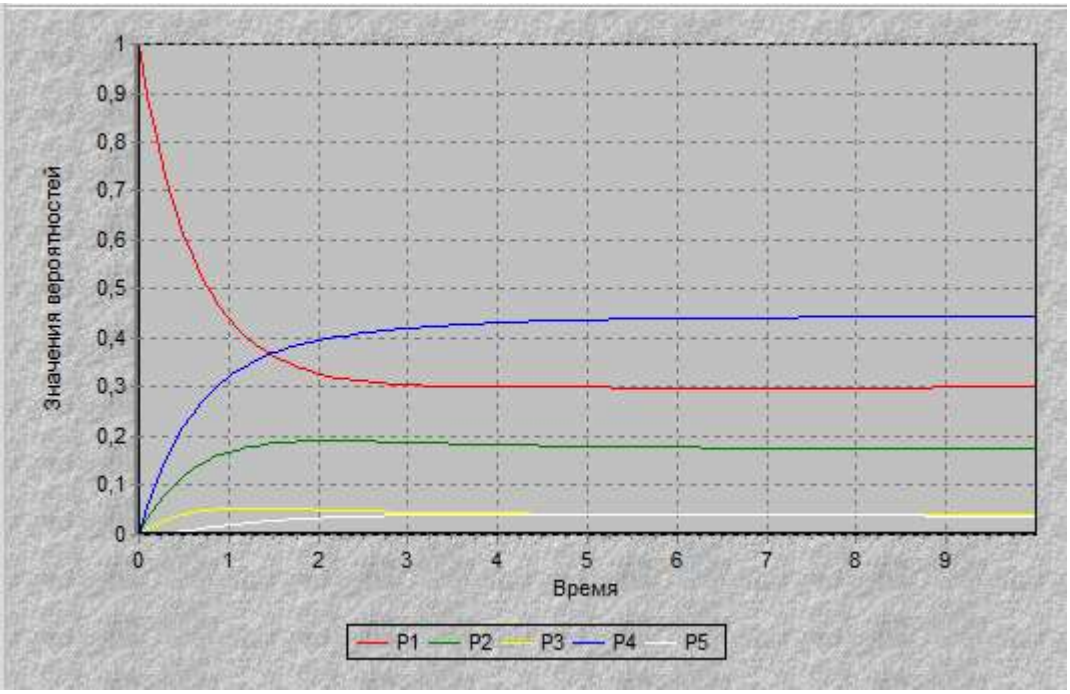


Рисунок В.4 – График показывающий, что наступил установившийся режим

СПИСОК ЛИТЕРАТУРЫ

1. Майоров С.А.Основы теории вычислительных систем.-М.:Высш.шк.,1977.-408 с.
2. Ларионов А.М.,Майоров С.А.Вычислительные машины,комплексы и сети.-Л.:Энергоатомиздат,1987.-215 с.
3. Корнеев В.В. Параллельные вічислительные системы..-М.:Нолидж,1999-320с.
4. Гук М. Аппаратные средства IBM PC. Энциклопедия.-СпБ.:ПИТЕР,1998.-816с.
5. Рудометов Е. и др. Архитектура ПК,комплектующие,мультимедиа..-СпБ.:ПИТЕР,2000.-416с.
6. Заморин А.П. Вычислительные машины, системы, комплексы: справочник. М.: Энергоатомиздат, 1985.-264 с.
7. Технические средства АСУ'.Справочник: В 2 Т.-Т. I. Технические средства ЕС ЭВМ/Под общ.ред.Г.Б.Кезлинга.-Л.Машиностроение Ленингр.отд-ние,1986.-544 с.
8. Денников Ю.Ф.Проектирование управляющих вычислительных комплексов для АСУ ТП.-М.:Энергоатомиздат,1986.-184 с.
9. Шенброт И.М.Распределение АСУ технологическими процессами-М,:Энергоатомиздат,1985.-238 с.
10. Хокни Р.,Джессхоуп К.Параллельные ЭВМ.-М.:Радио к связь, 1986.-390 с.
11. Вейцман К.Распределенные системы мини-и микроЭВМ.-М.:Финансы и статистика,1983.-382 с.
12. Феррари Д.Оценка производительности вычислительных систем.-М.:Мир,1981.-576 с.
13. Балашов Г.П.Эволюционный синтез систем.-М.:Радио и связь,1985.-328 с.
14. Мячев А.А. Интерфейсы ВС на базе мини- и микроЭвм – М. : Радио и связь, 1986г.. – 248 с.
15. Методические указания к практическим занятиям по курсу «Вычислительные комплексы и системы ». – Харьков, 1986. 24 с.
16. Каган Б. М.; Мкртумян И. Б. Основы эксплуатации ЭВМ. – М.: Энергоатомиздат, 1989. – 376 с.