Аннотация

В данном исследовании рассматривается система массового обслуживания типа G|M|n с накопителем бесконечной емкости и нетерпеливыми клиентами. Целью работы является нахождения оптимального управления входящим потоком заявок в систему с целью минимизации ее среднего срока окупаемости. Системы массового обслуживания широко используются для моделирования и анализа производственных, информационных или социальных систем. Одним из ключевых показателей эффективности системы является время ее окупаемости, так как этот показатель напрямую связан с тем, как быстро система начнет приносить доход. Результатами этого исследования являются построенная математическая модель системы в виде случайного процесса, построенная имитационная модель функционирования системы для экспоненциальной функции распределения интервалов между приходом требований в систему, а также найденная оптимальная стратегия управления входящим потоком заявок. Результаты, полученные в ходе выполнения данной работы, могут найти применение в других исследованиях или могут быть использованы для оптимизации реальных систем.

Abstract

This paper considers the G|M|n queuing system with infinite capacity storage and impatient customers. The purpose of this research is to determine optimal control strategy of incoming flow of requests to minimize the average payback period. Queuing systems are widely used for analysis and modeling of industrial, information, or social systems. One of the key performance indicators of any system is its payback time since this indicator is directly related to the moment when the system begins to generate profit. The results of this research are a constructed mathematical model in the form of the stochastic process, a constructed simulation model of the system functioning for exponential distribution function of intervals between requests arrivals, and the determined optimal control strategy of the incoming flow. The results obtained in this investigation can be applied in other studies or can be useful for optimization of real systems.

Содержание

Введение

Система массового обслуживания – объект, предназначенный для выполнения последовательности однотипных операций. Для того чтобы корректно описать систему массового обслуживания, необходимо правильно задать ее основные компоненты:

- Входящий поток заявок (требований);
- Процесс обслуживания;
- Обслуживающие приборы (каналы);
- Очередь (накопитель).

Для обозначения систем массового обслуживания принята символика Кендалла. В данной работе: G|M|n|∞, где G – распределение интервалов между приходами заявок в систему, M – распределение времени обслуживания заявок, n – число каналов обслуживания, ∞ – количество мест в очереди ожидания. Существует много разновидностей систем массового обслуживания, например системы с отказами, нетерпеливыми клиентами, повторными вызовами, приоритетными заявками, конечной или бесконечной очередью.

Целью данного исследования является нахождение оптимального управления входящим потоком заявок в немарковскую многоканальную систему массового обслуживания с накопителем бесконечной емкости с нетерпеливыми клиентами с целью минимизации срока окупаемости этой системы.

Для выполнения выше поставленной задачи необходимо понять, каким образом функционирует описанная система, формализовать задачу, построить математическую модель этой системы в виде случайного процесса, выбрать критерий эффективности, сформулировать задачу оптимизации, найти оптимальную стратегию управления входящим потоком (или показать, что ее нет) с помощью имитационного моделирования и проанализировать полученные результаты.

Теоретическая значимость и новизна этого исследования заключается в выборе критерия эффективности. В работе минимизируется среднее время функционирования системы до заработка определенного уровня дохода, в то время как обычно максимизируется средний удельный доход в единицу времени. Как правило, одним из главных показателей полезности той или иной системы является ее способность приносить прибыль на протяжении долгого времени функционирования. Однако срок окупаемости системы также является важным критерием, который и рассматривается в этой работе. Также исследуемая система является немарковской. В отличие от марковских систем массового обслуживания, такие системы мало изучены.

Практическая значимость этой работы определяется следующими факторами. Полученные в ходе исследования результаты в дальнейшем могут быть использованы для оптимизации работы, сокращения издержек и максимизации прибыли, например, каких-либо информационных (или любых других) систем, модель которых можно представить в виде заданной системы массового обслуживания. Так как сроки окупаемости системы являются важным фактором для получения дальнейшего дохода, результаты представленной работы могут найти применение в других исследованиях или реальных системах.

Список сокращений: СМО – система массового обслуживания; ПГР – процесс гибели и размножения; ЦМ – цепь Маркова; ФР – функция распределения; ДУ – дифференциальное уравнение; ПЛ – преобразование Лапласа; МПВ – матрица переходных вероятностей.

1 Обзор литературы

Изучение систем массового обслуживания является важным элементом оптимизации реальных систем, поэтому на эту тему проводится много исследований. Ниже представлен литературный обзор некоторых работ из этой области.

Так, например, в работе [1] автор рассматривает задачу оптимизации емкости основного накопителя в СМО типа G|M|1|K с дополнительным накопителем емкости M. Емкость основной очереди является одним из ключевых параметров, от объема которого зависит эффективность заданной системы.

Задана функция распределения интервалов времени между приходами требований (заявок) и экспоненциальная функция распределения с заданным параметром времени обслуживания заявок. Заявка поступает извне в главную очередь, если в ней есть место, и, если прибор свободен, заявка сразу же отправляется на обслуживание. После обслуживания требование либо покидает систему (успешное обслуживание), либо снова становится в очередь, освободив сам прибор. Если во время поступления заявки главная очередь заполнена, то требование либо теряется, либо встает в дополнительную очередь (при наличии свободных мест в ней, иначе теряется).

Пусть $D(K)$ есть предельный (стационарный) доход, получаемый системой в единицу времени, он рассматривается как функция переменной K, остальные параметры системы фиксированы. Тогда функция $D(K)$ является унимодальной для любого конечного значения $M \geq 0$. Также вычислительные эксперименты показали, что начиная с некоторого значения M емкость основного накопителя остается постоянной.

В статье [2] авторы провели исследование на тему оптимизации многоканальных систем массового обслуживания при больших загрузках. В работе рассматриваются модели марковских СМО с возможностью отказа заявке (требованию) в обслуживании. В системах массового обслуживания, как, например, технические или социальные приложения, зачастую встречаются экстремальные

ситуации – значительные загрузки (большие отношения интенсивности поступления требований к интенсивности обслуживания системой). По этой причине встает вопрос оптимизации модели – минимизация числа каналов обслуживания при условии гарантированной пропускной способности СМО в целом. Рассматривается система M|M|m, интенсивность входящего потока λ, интенсивность обслуживания μ.

Таким образом, была рассмотрена задача с предельно допустимыми нагрузками на систему массового обслуживания, в которой необходимо найти минимальное число каналов обслуживания, когда при высокой интенсивности прихода заявок обеспечивается почти безотказное функционирование системы массового обслуживания. Авторами был предложен эвристический алгоритм нахождения необходимого минимального числа каналов обслуживания, при котором вероятность отказов достаточно мала.

В исследовании [3] авторы изучили вопрос об оптимальном пороговом значении длины очереди с целью максимизации дохода СМО типа M|G|1 (один канал обслуживания и бесконечная очередь). На канал поступают требования с заданной интенсивностью, время обслуживания которых распределено по произвольному закону. Пришедшее требование встает в очередь (накопитель), если в момент ее прихода число занятых мест в очереди меньше заданного значения. Требование покидает систему только при завершении обслуживания.

В результате авторами были получены условия существования оптимального порогового значения длины очереди. Был найден алгоритм расчета оценок снизу для оптимальной длины очереди и соответствующего максимального дохода системы. Результаты исследования могут быть использованы при поиске оптимальных пороговых стратегий для систем, которые моделируются с помощью систем массового обслуживания типа M|G|1|r.

Таким образом, существует много разновидностей систем массового обслуживания, некоторые из которых были приведены выше. Однако реальные системы гораздо сложнее, чем рассматриваемые модели, поэтому возникает

необходимость изучать больше систем массового обслуживания с разными критериями и характеристиками. Каждый день запускаются новые проекты и в них функционируют новые и более сложные системы, работу которых также необходимо улучшать. Именно поэтому изучение методов теории массового обслуживания для анализа таких систем является важной задачей.

2 Основная часть

2.1 Функционирование системы с нетерпеливыми клиентами и отказами

В данной работе изучается система с нетерпеливыми клиентами и возможностью отказа в обслуживании. Это означает, что пришедшее в накопитель требование может с некоторой вероятностью покинуть систему из-за долгого ожидания (система в таком случае заплатит штраф). Рассматриваемая система имеет n каналов обслуживания, соответственно, в одно и то же время она может обрабатывать не более n требований. Таким образом, если в системе находится n требований, то пришедшее $n + 1$ требование становится в очередь на обслуживание (отправляется в накопитель бесконечной емкости), где и будет находиться до освобождения одного из приборов. Аналогично $n + 1$ заявке в очередь будут вставать все последующие требования и обслуживаться по мере освобождения приборов. Также система в праве отказать заявке в обслуживании (заявка не попадет в накопитель). Из накопителя заявки поступают в каналы обслуживания по принципу FIFO (First-In-First-Out). Моделью описанной системы может служить многоканальная СМО, схема которой представлена на Рисунке 1:

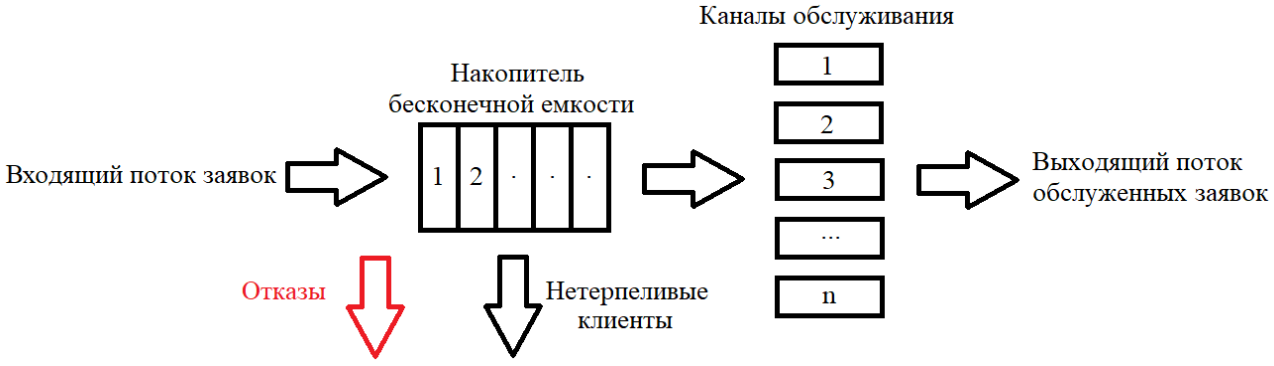


Рисунок. 1. Модель системы с отказами и нетерпеливыми клиентами.

2.2 Постановка задачи

Будем предполагать, что входящие в систему запросы образуют произвольный поток с заданной ФР. Время обслуживания заявок в каналах имеет экспоненциальное распределение с заданным параметром μ. Так как запросы, находящиеся в накопителе, являются нетерпеливыми, то каждый из них может покинуть систему, не дождавшись своего обслуживания. Время ожидания заявок в накопителе имеет экспоненциальное распределение с заданным параметром γ. Рассматриваемая система приносит определенный доход. Этот доход может быть как положительным, так и отрицательным (в зависимости от заданных параметров). Доход системы складывается из следующих компонент:

- C_0 – прибыль за обслуживание заявки;
- C_1 – штраф за уход требования из накопителя;
- C_2 – штраф в единицу времени за ожидание заявки в очереди;
- C_3 – затраты в единицу времени на обслуживание приборов;
- C_4 – штраф за недопуск заявки в систему.

В рамках поставленной задачи необходимо корректно управлять входящим потоком требований. Грамотное управления является неотъемлемой частью эффективного функционирования системы, поскольку иначе вполне возможно, что система будет нести убытки. Таким образом, во время прихода очередной заявки принимается решение о ее допуске или недопуске в систему. Это необходимо для минимизации ущерба. Например, если интенсивность поступления заявок высокая, то каналы обслуживания не сумеют обработать все приходящие запросы, значит, заявки будут попадать в очередь. Чем больше заявок в очереди, тем, во-первых, чаще будут уходить нетерпеливые клиенты (система будет нести убытки), и, во-вторых, штрафы за ожидание заявок в очереди будут выше. С другой стороны, если ограничить емкость накопителя слишком маленьким числом, то в этом случае заявки,

которые система потенциально могла обслужить, не будут попадать в систему (штраф за отказ в обслуживании), а недополученная прибыль будет расти.

Таким образом, задача заключается в том, чтобы оптимизировать данную систему, регулируя входящий поток заявок, с целью минимизации среднего времени до накопления заданного уровня дохода.

2.3 Построение математической модели

Итак, рассматриваемая система удобно представляется в виде СМО вида G|M|n|∞ с нетерпеливыми клиентами. Построим математическую модель этой системы.

- G – входящий поток заявок;
- M – процесс обслуживания заявок;
- n – количество обслуживающих приборов в системе;
- ∞ – количество мест в очереди;
- μ – интенсивность обслуживания требований;
- γ – среднее время ожидания нетерпеливых клиентов в очереди.

Регулировать очередь будем по следующему правилу: в момент прихода нового требования подсчитывается количество заявок в системе и на основании этого делается выбор: принять требование в систему или нет. С некоторой вероятностью p_v заявка допускается в систему, а с вероятностью $1 - p_v$ не допускается ($v -$ количество заявок в системе).

Введем случайный процесс $\xi(t)$ – количество заявок в системе в момент времени t, $E = \{0,1,2, \dots\}$ есть множество состояний процесса. По причине того, что из-за не экспоненциальной функции распределения времени интервалов между приходами заявок в систему, не выполняется свойство отсутствия последействия. Свойство отсутствия последействия гласит, что вероятность появления k событий на любом промежутке времени не зависит от того, появлялись или не появлялись

события в моменты времени, предшествующие началу рассматриваемого промежутка. То есть, выражаясь нестрого, при фиксированном настоящем будущее не зависит от прошлого. По этой причине процесс $\xi(t)$ не является марковским. В произвольный момент времени мы не можем говорить о марковском свойстве этого процесса: случайный процесс обладает марковским свойством, если условное распределение вероятностей будущих состояний процесса зависит только от текущего состояния, а не от предшествующих событий. В связи с этим, метод вложенных цепей Маркова будет использоваться в этой работе (подробно описан в [4]). Суть этого метода заключается в том, что необходимо определить специальные моменты времени, в которые рассматриваемый процесс будет обладать марковским свойством. Обозначим моменты прихода новых требований в систему за t_n. Тогда последовательность $\xi_n = \xi(t_n - 0)$ образует вложенную ЦМ. В рассматриваемой модели случайный процесс $\xi(t)$ будет обладать марковским свойством именно в моменты прихода новых заявок в систему [5].

2.3.1 Матрица переходных вероятностей

Теперь необходимо найти МПВ вложенной марковской цепи. Она выглядит следующим образом:

$$P = \begin{pmatrix} P_{00} & P_{01} & \dots \\ P_{10} & P_{11} & \dots \\ \dots & \dots & \dots \end{pmatrix},$$

где $P_{ij} = P(\xi_{n+1} = j \mid \xi_n = i)$.

ФР времени между поступлениями заявок в систему обозначим как $A(t)$. Рассмотрим несколько случаев.

I. Сначала рассмотрим случай, когда система переходит более чем на одно состояние вверх. То есть, если $j > i + 1$. В таком случае:

$$P_{ij} = 0.$$

Очевидно, что за время $(t_n; t_{n+1})$ система может перейти вверх из состояния i только в состояние $j = i + 1$, так как за это время придет только одна заявка.

II. Теперь рассмотрим случай, когда в начальном состоянии в системе находится заявок не больше, чем есть каналов обслуживания: $j \leq i + 1 \leq n$ (то есть очередь пуста). Пусть $i + 1 - j$ есть число требований, покинувших систему. Тогда вероятность того, что приборы обслужат ровно $i + 1 - j$ заявок находится из биномиального распределения:

$$P_{ij}(t) = C_{i+1}^{i+1-j}(1 - e^{-\mu t})^{i+1-j} e^{-j\mu t}.$$

Получаем следующее выражение для переходных вероятностей:

$$P_{ij} = \int_0^{\infty} C_{i+1}^{i+1-j}(1 - e^{-\mu t})^{i+1-j} e^{-j\mu t}\, dA(t)$$

III. Далее рассмотрим случай, когда число заявок в системе в начальном состоянии больше, чем каналов обслуживания. Здесь нетерпеливые клиенты уже оказывают влияние не переходные вероятности. Для изучения этого случая введем новый случайный процесс $\upsilon(t)$ – число заявок в системе в момент времени t в интервале между приходами соседних требований $(t_n; t_{n+1})$. Множество состояний процесса $E = \{0,1, \dots, i + 1\}, n < i + 1 < \infty$. Для удобства введем обозначение $k = i + 1$. Этот однородный марковский процесс является процессом чистой гибели (частный случай процесса гибели и размножения), потому что из начального состояния k процесс может перейти либо на одно состояние вниз в $k - 1$, либо остаться в состоянии k. Это обусловлено тем, что в интервалах между приходами двух соседних требований количество заявок в системе не может увеличиться. Такие процессы задаются инфинитезимальными характеристиками. Интенсивность перехода определяется по формуле:

$$a_{ij} = \lim_{t \to 0} \frac{P_{ij}(t)}{t}, \qquad i \neq j.$$

Найдем интенсивности перехода a_{ij} для процесса $\upsilon(t)$. Как уже было упомянуто, процесс не может перейти из состояния i в состояние $i+1$ и так далее, также процесс не может перейти более чем на одно состояние вниз за малый промежуток времени, поэтому интенсивности перехода для этих случаев будут равны нулю. Если процесс стартует из состояния $i > n$, то в данный момент времени на обслуживании находятся n заявок, а $i-n$ заявок ожидают в очереди, поэтому интенсивность перехода равна $n\mu + (i-n)\gamma$. Если же в начальный момент времени процесс находится в состоянии $i \leq n$, то на каналах обслуживания находятся ровно i требований, а очередь пуста, поэтому интенсивность перехода равна $i\mu$. Таким образом, мы получили интенсивности перехода a_{ij} для процесса $\upsilon(t)$:

$$a_{ij} = \begin{cases} 0, & |i-j| > 1 \\ 0, & j > i \\ n\mu + (i-n)\gamma, & j = i-1, \ i > n \\ i\mu, & j = i-1, \ i \leq n \end{cases}$$

Интенсивность выхода определяется по формуле:

$$a_i = \lim_{t \to 0} \frac{1 - P_{ii}(t)}{t}.$$

Для регулярного процесса (процесс $\upsilon(t)$ является регулярным, так как имеет конечное множество состояний) интенсивности выхода можно найти следующим образом:

$$a_i = \sum_{i \neq j} a_{ij}.$$

Так как в нашем случае интенсивность перехода a_{ij} для любого $j \neq i-1$ и $j \neq i$ равняется нулю, то справедливо следующее соотношение:

$$a_i = \sum_{i \neq j} a_{ij} = a_{i,i-1} \tag{1}$$

Тогда можем выписать выражение для интенсивности выхода a_i для процесса $\upsilon(t)$:

$$a_i = \begin{cases} n\mu + (i-n)\gamma, & i > n \\ i\mu, & i \leq n \end{cases}$$

Теперь выпишем систему дифференциальных уравнений Колмогорова для переходных вероятностей $P_j(t)$ рассматриваемого случайного процесса $\upsilon(t)$. Общий вид системы ДУ Колмогорова для ПГР записывается в следующем виде:

$$\begin{cases} P_j'(t) = -a_j P_j(t) + \sum_{i \neq j} a_{ij} P_i(t) \\ P_j(0) = p_j(0), \ \ j \in E \end{cases}$$

$$P_j(t) = P(\upsilon(t) = j)$$

Воспользовавшись соотношением (1) можно переписать систему:

$$\begin{cases} P_j'(t) = -a_j P_j(t) + a_{j+1} P_{j+1}(t) \\ P_j(0) = p_j(0), \ \ j \in E \end{cases}$$

$$P_j(t) = P(\upsilon(t) = j)$$

В начальный момент времени $t = 0$ случайный процесс стартует из состояния k. Вероятность того, что из состояния k система перейдет в состояние $k + 1$ равняется 0. Тогда система ДУ Колмогорова для процесса $\upsilon(t)$ может быть записана в следующем виде:

$$\begin{cases} P_j'(t) = -a_j P_j(t) + a_{j+1} P_{j+1}(t) \\ P_k'(t) = -a_k P_k(t) \\ P_k(0) = 1 \\ P_j(0) = 0, \qquad \forall j \in E: \ \ j \neq k \end{cases}$$

Подставим найденные значения в систему и перепишем, разбив на случаи:

$$\begin{cases} P_j'(t) = -j\mu P_j(t) + (j+1)\mu P_{j+1}(t), \qquad j < n \\ P_j'(t) = -(n\mu + (j-n)\gamma) P_j(t) + (n\mu + (j-n+1)\gamma) P_{j+1}(t), \qquad k > j \geq n \\ P_j'(t) = -[n\mu + (j-n)\gamma] P_j(t), \qquad j = k \\ P_k(0) = 1 \\ P_j(0) = 0, \qquad \forall j \in E: \ \ j \neq k \end{cases}$$

Решение приведенной выше системы дифференциальных уравнений можно найти с помощью преобразования Лапласа. Для этого необходимо сначала записать ее в терминах ПЛ, затем найти решение системы в терминах ПЛ, и с помощью

обратного преобразования Лапласа получить решение исходной системы. Прямое преобразование Лапласа:

$$F^*(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t) dt,$$

где $f(t)$ – оригинал, $F^*(s)$ – изображение функции $f(t)$. В нашем случае $F_j^*(s) = \mathcal{L}\{P_j(t)\}$. Обратное преобразование Лапласа:

$$f(t) = \mathcal{L}^{-1}\{F^*(s)\} = \frac{1}{2\pi i} \lim_{\omega \to \infty} \int_{\sigma - i\omega}^{\sigma + i\omega} e^{st} F^*(s) ds\,,$$

где σ – некоторое вещественное число. В нашем случае $P_j(t) = \mathcal{L}^{-1}\{F^*(s)\}$.

Некоторые свойства преобразования Лапласа:

1. Линейность:

$$\mathcal{L}\{af(t) + bg(t)\} = aF^*(s) + bG^*(s)$$

2. Умножение на число:

$$\mathcal{L}\{f(at)\} = \frac{1}{a} F^*\left(\frac{s}{a}\right)$$

3. Дифференцирование оригинала:

$$\mathcal{L}\{f'(t)\} = sF^*(s) - f(0^+)$$

4. Экспоненциальная функция:

$$\mathcal{L}\{e^{-at}\} = \frac{1}{s + a}$$

$$\mathcal{L}\{e^{-at} f(t)\} = F^*(s + a)$$

5. Степенная функция:

$$\mathcal{L}\{t^n\} = \frac{n!}{s^{n+1}}$$

6. Умножение изображения на число:

$$\mathcal{L}^{-1}\{aF^*(s)\} = a\mathcal{L}^{-1}\{F^*(s)\} = af(t)$$

Теперь, воспользовавшись прямым преобразованием Лапласа и его свойствами, можно переписать систему ДУ Колмогорова в терминах ПЛ:

$$\begin{cases} sF_j^*(s) - P_j(0) = -j\mu F_j^*(s) + (j+1)\mu F_{j+1}^*(s), & j < n \\ sF_j^*(s) - P_j(0) = -[n\mu + (j-n)\gamma]F_j^*(s) + [n\mu + (j-n+1)\gamma]F_{j+1}^*(s), & j \geq n \\ sF_j^*(s) - P_j(0) = -[n\mu + (j-n)\gamma]F_j^*(t), & j = k \\ P_k(0) = 1 \\ P_j(0) = 0, \quad \forall j \in E: \ j \neq k \end{cases}$$

Приводим подобные слагаемые, получаем:

$$\begin{cases} (s + j\mu)F_j^*(s) - P_j(0) = (j+1)\mu F_{j+1}^*(s), & j < n \\ [s + n\mu + (j-n)\gamma]F_j^*(s) - P_j(0) = [n\mu + (j-n+1)\gamma]F_{j+1}^*(s), & j \geq n \\ [s + n\mu + (j-n)\gamma]F_j^*(s) - P_j(0) = 0, & j = k \\ P_k(0) = 1 \\ P_j(0) = 0, \quad \forall j \in E: \ j \neq k \end{cases}$$

Теперь запишем систему в явном виде для каждого $j \in E$:

$$\begin{cases} sF_0^*(s) - P_0(0) = \mu F_1^*(s) \\ (s+\mu)F_1^*(s) - P_1(0) = 2\mu F_2^*(s) \\ (s+2\mu)F_2^*(s) - P_2(0) = 3\mu F_3^*(s) \\ \dots \\ (s + (n-1)\mu)F_{n-1}^*(s) - P_{n-1}(0) = n\mu F_n^*(s) \\ (s+n\mu)F_n^*(s) - P_n(0) = (n\mu + \gamma)F_{n+1}^*(s) \\ (s + n\mu + \gamma)F_{n+1}^*(s) - P_{n+1}(0) = (n\mu + 2\gamma)F_{n+2}^*(s) \\ (s + n\mu + 2\gamma)F_{n+2}^*(s) - P_{n+2}(0) = (n\mu + 3\gamma)F_{n+3}^*(s) \\ \dots \\ (s + n\mu + (k-1-n)\gamma)F_{k-1}^*(s) - P_{n+2}(0) = (n\mu + (k-n)\gamma)F_k^*(s) \\ (s + n\mu + (k-n)\gamma)F_k^*(s) - P_k(0) = 0 \\ P_k(0) = 1 \\ P_j(0) = 0, \quad \forall j \in E: \ j \neq k \end{cases}$$

Учтем начальное распределение и получим следующие выражения:

$$\begin{cases} sF_0^*(s) = \mu F_1^*(s) \\ (s+\mu)F_1^*(s) = 2\mu F_2^*(s) \\ (s+2\mu)F_2^*(s) = 3\mu F_3^*(s) \\ \dots \\ (s+(n-1)\mu)F_{n-1}^*(s) = n\mu F_n^*(s) \\ (s+n\mu)F_n^*(s) = (n\mu+\gamma)F_{n+1}^*(s) \\ (s+n\mu+\gamma)F_{n+1}^*(s) = (n\mu+2\gamma)F_{n+2}^*(s) \\ (s+n\mu+2\gamma)F_{n+2}^*(s) = (n\mu+3\gamma)F_{n+3}^*(s) \\ \dots \\ (s+n\mu+(k-1-n)\gamma)F_{k-1}^*(s) = (n\mu+(k-n)\gamma)F_k^*(s) \\ (s+n\mu+(k-n)\gamma)F_k^*(s) = 1 \end{cases}$$

Выразим каждую функцию $F_j^*(s)$ через $F_{j+1}^*(s), j \in E\backslash\{k\}$. Выражение для $F_k^*(s)$ получено в явном виде. Система выглядит следующим образом:

$$\begin{cases} F_0^*(s) = \dfrac{\mu}{s}F_1^*(s) \\ F_1^*(s) = \dfrac{2\mu}{s+\mu}F_2^*(s) \\ \dots \\ F_{n-1}^*(s) = \dfrac{n\mu}{s+(n-1)\mu}F_n^*(s) \\ F_n^*(s) = \dfrac{n\mu+\gamma}{s+n\mu}F_{n+1}^*(s) \\ F_{n+1}^*(s) = \dfrac{n\mu+2\gamma}{s+n\mu+\gamma}F_{n+2}^*(s) \\ \dots \\ F_{k-1}^*(s) = \dfrac{n\mu+(k-n)\gamma}{s+n\mu+(k-1-n)\gamma}F_k^*(s) \\ F_k^*(s) = \dfrac{1}{s+n\mu+(k-n)\gamma} \end{cases}$$

Можно выписать выражение для функции $F_n^*(s)$ в явном виде, подставив в него значения функций $F_{n+1}^*(s), \dots, F_k^*(s)$. В свою очередь выражение для функции $F_0^*(s)$ можно выписать как зависимость от $F_n^*(s)$, при этом подставить значения функций $F_1^*(s), \dots, F_{n-1}^*(s)$. Тога система будет записана в следующем виде:

$$\begin{cases} F_0^*(s) = \dfrac{\mu}{s} * \dfrac{2\mu}{s+\mu} * \dfrac{3\mu}{s+2\mu} * \dots * \dfrac{n\mu}{s+(n-1)\mu}F_n^*(s) \\ F_n^*(s) = \dfrac{n\mu+\gamma}{s+n\mu} * \dfrac{n\mu+2\gamma}{s+n\mu+\gamma} * \dots * \dfrac{n\mu+(k-n)\gamma}{s+n\mu+(k-1-n)\gamma} * \dfrac{1}{s+n\mu+(k-n)\gamma} \end{cases}$$

Далее рассмотрим два случая.

a) $j \geq n$.

Выражение для функции $F_k^*(s)$ останется без изменений:

$$F_j^*(s) = \frac{1}{s + n\mu + (k-n)\gamma}, \qquad j = k$$

Нетрудно заметить, что выражения для функций $F_n^*(s), \dots, F_{k-1}^*(s)$ можно представить в следующем виде:

$$F_j^*(s) = \frac{1}{s + n\mu + (k-n)\gamma} \prod_{l=j-n}^{k-1-n} \frac{n\mu + (l+1)\gamma}{s + n\mu + l\gamma}, \qquad n \leq j < k$$

Для удобства можно сразу же выписать выражение для $F_n^*(s)$:

$$F_n^*(s) = \frac{1}{s + n\mu + (k-n)\gamma} \prod_{l=0}^{k-1-n} \frac{n\mu + (l+1)\gamma}{s + n\mu + l\gamma}$$

b) $j < n$:

Выражения для функций $F_0^*(s), \dots, F_{n-1}^*(s)$ записываются так:

$$F_j^*(s) = F_n^*(s) \prod_{l=j}^{n-1} \frac{(l+1)\mu}{s + l\mu}, \qquad j < n$$

Подставим в предыдущую формулу выражение для $F_n^*(s)$ и получим окончательные выражения:

$$F_j^*(s) = \frac{1}{s + n\mu + (k-n)\gamma} \prod_{l=0}^{k-1-n} \frac{n\mu + (l+1)\gamma}{s + n\mu + l\gamma} \prod_{l=j}^{n-1} \frac{(l+1)\mu}{s + l\mu}, \qquad j < n$$

Таким образом, решение системы ДУ Колмогорова в терминах ПЛ принимает следующий вид:

$$\begin{cases} F_j^*(s) = \dfrac{1}{s+n\mu+(k-n)\gamma}, & j = k \\ F_j^*(s) = \dfrac{1}{s+n\mu+(k-n)\gamma} \prod\limits_{l=j-n}^{k-1-n} \dfrac{n\mu+(l+1)\gamma}{s+n\mu+l\gamma}, & n \le j < k \\ F_j^*(s) = \dfrac{1}{s+n\mu+(k-n)\gamma} \prod\limits_{l=0}^{k-1-n} \dfrac{n\mu+(l+1)\gamma}{s+n\mu+l\gamma} \prod\limits_{l=j}^{n-1} \dfrac{(l+1)\mu}{s+l\mu}, & j < n \end{cases}$$

Теперь необходимо найти решение исходной системы дифференциальных уравнений, используя обратное ПЛ. Разберемся с 3 случаями по очереди.

1. $j = k$.

Для этого случая все тривиально:

$$P_k(t) = \mathcal{L}^{-1}\{F_k^*(s)\} = e^{-(n\mu+(k-n)\gamma)t}.$$

2. $n \le j < k$.

$$P_j(t) = \mathcal{L}^{-1}\{F_j^*(s)\} = \mathcal{L}^{-1}\left\{\frac{1}{s+n\mu+(k-n)\gamma} \prod_{l=j-n}^{k-1-n} \frac{n\mu+(l+1)\gamma}{s+n\mu+l\gamma}\right\}$$

Этот случай рассмотрим поэтапно. Сначала используем обратное ПЛ для $F_{k-1}^*(s)$:

$$P_{k-1}(t) = \mathcal{L}^{-1}\{F_{k-1}^*(s)\} = \mathcal{L}^{-1}\left\{\frac{1}{s+n\mu+(k-n)\gamma} \frac{n\mu+(k-n)\gamma}{s+n\mu+(k-1-n)\gamma}\right\}$$

Можно найти ответ с помощью свойств ПЛ, мы же воспользуемся сервисом Wolfram Alpha (всю информацию можно найти в приложении А). Таким образом:

$$P_{k-1}(t) = \frac{e^{\gamma t}-1}{\gamma} e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-n)\gamma)$$

Найдем выражение для $P_{k-2}(t)$:

$$P_{k-2}(t) = \mathcal{L}^{-1}\{F_{k-2}^*(s)\} =$$

$$= \mathcal{L}^{-1}\left\{\frac{1}{s+n\mu+(k-n)\gamma} \frac{n\mu+(k-n)\gamma}{s+n\mu+(k-1-n)\gamma} \frac{n\mu+(k-1-n)\gamma}{s+n\mu+(k-2-n)\gamma}\right\} =$$

$$= \frac{(e^{\gamma t}-1)^2}{2\gamma^2} e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-n)\gamma)(n\mu+(k-1-n)\gamma)$$

Аналогично получаем выражение для $P_{k-3}(t)$:

$$P_{k-3}(t) = \mathcal{L}^{-1}\{F^*_{k-3}(s)\} =$$

$$= \mathcal{L}^{-1}\left\{\begin{array}{c}\frac{1}{s+n\mu+(k-n)\gamma}\frac{n\mu+(k-n)\gamma}{s+n\mu+(k-1-n)\gamma}\frac{n\mu+(k-1-n)\gamma}{s+n\mu+(k-2-n)\gamma}* \\ *\frac{n\mu+(k-1-n)\gamma}{s+n\mu+(k-2-n)\gamma}\end{array}\right\} =$$

$$= \frac{(e^{\gamma t}-1)^3}{6\gamma^3} e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-n)\gamma)(n\mu+(k-1-n)\gamma) *$$

$$*(n\mu+(k-2-n)\gamma)$$

В приведенных выражениях можно увидеть естественную закономерность и вывести итоговую формулу для $P_j(t)$ при $n \le j < k$:

$$P_j(t) = \frac{(e^{\gamma t}-1)^{k-j}}{(k-j)!\,\gamma^{k-j}} e^{-(n\mu+(k-n)\gamma)t} \prod_{l=j-n+1}^{k-n}(n\mu+l\gamma), \qquad n \le j < k$$

Сейчас давайте проверим справедливость полученного выражения, а именно подставим найденные с помощью него $P_j(t)$ в систему ДУ Колмогорова. Напомним, что для случая, когда $n \le j < k$, справедливо следующее равенство:

$$P'_j(t) = -[n\mu+(j-n)\gamma]P_j(t) + [n\mu+(j-n+1)\gamma]P_{j+1}(t)$$

Рассмотрим случай, когда $j = k-2$ (тем более выражения для $P_{k-2}(t)$ и $P_{k-1}(t)$ уже были выписаны в явном виде). Тогда:

$$P'_{k-2}(t) = -[n\mu+(k-2-n)\gamma]P_{k-2}(t) + [n\mu+(k-1-n)\gamma]P_{k-1}(t)$$

$$\left(\frac{(e^{\gamma t}-1)^2}{2\gamma^2} e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-n)\gamma)(n\mu+(k-1-n)\gamma)\right)' = \frac{(e^{\gamma t}-1)^2}{-2\gamma^2} *$$

$$* e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-n)\gamma)(n\mu+(k-1-n)\gamma)(n\mu+(k-2-n)\gamma) +$$

$$+\frac{e^{\gamma t}-1}{\gamma}e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-n)\gamma)(n\mu+(k-1-n)\gamma)$$

Из-под производной можно вынести множители, которые не зависят от t. Затем, поделив обе части равенства на $(n\mu+(k-n)\gamma)(n\mu+(k-1-n)\gamma)$ и вычислив производную, получим:

$$\begin{gathered}-\frac{(e^{\gamma t}-1)^2}{2\gamma^2}e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-n)\gamma)+\frac{e^{\gamma t}-1}{\gamma}e^{-(n\mu+(k-n)\gamma)t}e^{\gamma t}= \\ =-\frac{(e^{\gamma t}-1)^2}{2\gamma^2}e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-2-n)\gamma)+\frac{e^{\gamma t}-1}{\gamma}e^{-(n\mu+(k-n)\gamma)t}\end{gathered} \quad (2)$$

Рассмотрим более подробно левую часть равенства (2). Разобьем первое слагаемое на два, вынеся 2γ за скобки: $n\mu+(k-n)\gamma=n\mu+(k-2-n)\gamma+2\gamma$. Тогда получим:

$$-\frac{(e^{\gamma t}-1)^2}{2\gamma^2}e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-2-n)\gamma)-\frac{(e^{\gamma t}-1)^2}{\gamma}e^{-(n\mu+(k-n)\gamma)t}+$$

$$+\frac{e^{\gamma t}-1}{\gamma}e^{-(n\mu+(k-n)\gamma)t}e^{\gamma t}$$

Сгруппируем второе и третье слагаемые и вынесем общий множитель. Получим:

$$-\frac{(e^{\gamma t}-1)^2}{2\gamma^2}e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-2-n)\gamma)+$$

$$\frac{e^{\gamma t}-1}{\gamma}e^{-(n\mu+(k-n)\gamma)t}[-(e^{\gamma t}-1)+e^{\gamma t}]=$$

$$=-\frac{(e^{\gamma t}-1)^2}{2\gamma^2}e^{-(n\mu+(k-n)\gamma)t}(n\mu+(k-2-n)\gamma)+\frac{e^{\gamma t}-1}{\gamma}e^{-(n\mu+(k-n)\gamma)t}$$

Очевидно, мы получили то же самое выражение, которое стоит в правой части равенства (2). Таким образом, можно с уверенность сказать, что найденное выражение для $P_j(t)$ при $n\le j<k$ является верным.

3. $j<n$.

$$P_j(t) = \mathcal{L}^{-1}\{F_j^*(s)\} = \mathcal{L}^{-1}\left\{\frac{1}{s+n\mu+(k-n)\gamma}\prod_{l=0}^{k-1-n}\frac{n\mu+(l+1)\gamma}{s+n\mu+l\gamma}\prod_{l=j}^{n-1}\frac{(l+1)\mu}{s+l\mu}\right\}$$

Этот случай несколько сложнее, чем предыдущие два. Для начала сформулируем следующее утверждение [6] и докажем его.

Утверждение 1.

Если $F^*(s) = \frac{C}{B(s)}$ рациональная дробь, где C – константа, а многочлен $B(s)$ имеет только простые корни $s_0, s_1, \dots, s_a$, то функция $f(t) = \sum_{a=0}^{m}\frac{C}{B'(s_a)}e^{s_a t}$ является оригиналом, имеющим изображение $F^*(s)$.

Доказательство:

Разложим рациональную дробь $\frac{C}{B(s)}$ на простейшие:

$$\frac{C}{B(s)} = \frac{c_0}{s-s_0} + \frac{c_1}{s-s_1} + \dots + \frac{c_a}{s-s_a}, \tag{3}$$

c_m – неопределенный коэффициенты. Чтобы определить коэффициент c_0 для разложения сначала умножим обе части равенства (3) на $s - s_0$. Получим:

$$\frac{C}{B(s)}(s-s_0) = c_0 + \left[\frac{c_1}{s-s_1} + \dots + \frac{c_a}{s-s_a}\right](s-s_0)$$

Далее перейдем к пределу при $s \to s_0$:

$$c_0 = \lim_{s\to s_0}\frac{C}{B(s)}(s-s_0) = \lim_{s\to s_0}\frac{C}{\frac{B(s)-B(s_0)}{s-s_0}} = \frac{C}{B'(s_0)}.$$

Мы получили выражение для c_0. Аналогичным образом можно найти выражения для остальных коэффициентов c_m:

$$c_m = \frac{C}{B'(s_m)}.$$

Теперь подставим найденные коэффициенты c_m в равенство (3):

$$\frac{C}{B(s)} = \frac{C}{B'(s_0)}\frac{1}{s-s_0} + \frac{C}{B'(s_1)}\frac{1}{s-s_1} + \cdots + \frac{C}{B'(s_a)}\frac{1}{s-s_a}$$

Получили следующее выражение для $F^*(s)$:

$$F^*(s) = \frac{C}{B(s)} = \sum_{a=0}^{m} \frac{C}{B'(s_a)}\frac{1}{s-s_a}$$

В силу свойств ПЛ получаем конечное выражение для оригинала:

$$f(t) = \mathcal{L}^{-1}\{F^*(s)\} = \mathcal{L}^{-1}\left\{\sum_{a=0}^{m} \frac{C}{B'(s_a)}\frac{1}{s-s_a}\right\} = \sum_{a=0}^{m} \frac{C}{B'(s_a)} e^{s_a t}. \quad \blacksquare$$

Покажем, что функции $F_j^*(s)$ при $j < n$ можно представить в виде рациональных дробей $\frac{C}{B_j(s)}$:

$$F_j^*(s) = \frac{1}{s+n\mu+(k-n)\gamma} \prod_{l=0}^{k-1-n} \frac{n\mu+(l+1)\gamma}{s+n\mu+l\gamma} \prod_{l=j}^{n-1} \frac{(l+1)\mu}{s+l\mu} =$$

$$= \frac{\prod_{l=0}^{k-1-n}(n\mu+(l+1)\gamma) * \prod_{l=j}^{n-1}((l+1)\mu)}{(s+n\mu+(k-n)\gamma) * \prod_{l=0}^{k-1-n}(s+n\mu+l\gamma) * \prod_{l=j}^{n-1}(s+l\mu)}.$$

Выпишем выражение для $B_j(s)$:

$$B_j(s) = (s+n\mu+(k-n)\gamma) \prod_{l=0}^{k-1-n} (s+n\mu+l\gamma) \prod_{l=j}^{n-1}(s+l\mu) =$$

$$= (s+n\mu)(s+n\mu+\gamma)\ldots(s+n\mu+(k-1-n)\gamma)(s+n\mu+(k-n)\gamma) *$$

$$* (s+j\mu)(s+(j+1)\mu)\ldots(s+(n-1)\mu)$$

Видно, что многочлены $B_j(s)$ имеют только простые корни. Тогда можно выписать выражение для $P_j(t)$ при $j < n$ воспользовавшись утверждением 1:

$$P_j(t) = \sum_{a=0}^{k-j} \frac{\prod_{l=0}^{k-n-1}(n\mu+(l+1)\gamma)\ \prod_{l=j}^{n-1}\big((l+1)\mu\big)}{B_j'(s_a)} e^{s_a t}, \qquad j < n,$$

где

$$B_j(s) = (s + n\mu + (k-n)\gamma) \prod_{l=0}^{k-1-n} (s + n\mu + l\gamma) \prod_{l=j}^{n-1} (s + l\mu),$$

s_a – корни многочлена $B_j(s),\ a = 0,1, \dots, k-j$:

$$\begin{cases} s_0 = -n\mu \\ s_1 = -(n\mu + 1\gamma) \\ \dots \\ s_{k-1-n} = -(n\mu + (k-1-n)\gamma) \\ s_{k-n} = -(n\mu + (k-n)\gamma) \\ s_{k-n+1} - (n-1)\mu \\ s_{k-n+2} - (n-2)\mu \\ \dots \\ s_{k-j} = -j\mu \end{cases}$$

Сейчас давайте проверим справедливость полученного выражения, а именно подставим найденные с помощью него $P_j(t)$ в систему ДУ Колмогорова. Напомним, что для случая, когда $j < n$, справедливо следующее равенство:

$$P_j'(t) = -j\mu P_j(t) + (j+1)\mu P_{j+1}(t)$$

Рассмотрим случай, когда $j = n-2$. Тогда:

$$P_{n-2}'(t) = -(n-2)\mu P_{n-2}(t) + (n-1)\mu P_{n-1}(t)$$

Запишем выражение для $P_{n-1}(t)$:

$$P_{n-1}(t) = \sum_{a=0}^{k-n+1} \frac{n\mu \prod_{l=0}^{k-n-1}(n\mu + (l+1)\gamma)}{B_{n-1}'(s_a)} e^{s_a t}$$

где

$$B_{n-1}(s) = (s + (n-1)\mu)(s + n\mu + (k-n)\gamma) \prod_{l=0}^{k-1-n} (s + n\mu + l\gamma),$$

s_a – корни многочлена $B_{n-1}(s),\ a = 0,1, \dots, k-n+1$.

Теперь выпишем выражение для $P_{n-2}(t)$:

$$P_{n-2}(t) = \sum_{a=0}^{k-n+2} \frac{n\mu(n-1)\mu\prod_{l=0}^{k-n-1}(n\mu+(l+1)\gamma)}{B'_{n-2}(s_a)} e^{s_a t}$$

где

$$B_{n-2}(s) = (s+(n-2)\mu)(s+(n-1)\mu)(s+n\mu+(k-n)\gamma)\prod_{l=0}^{k-1-n}(s+n\mu+l\gamma),$$

s_a – корни многочлена $B_{n-2}(s),\ a = 0,1,\dots,k-n+2.$

Выражение для $P'_{n-2}(t)$ будет выглядеть следующим образом:

$$P'_{n-2}(t) = \frac{d}{dt}\left(\sum_{a=0}^{k-n+2} \frac{n\mu(n-1)\mu\prod_{l=0}^{k-n-1}(n\mu+(l+1)\gamma)}{B'_{n-2}(s_a)} e^{s_a t}\right)$$

Из-под производной можно вынести множители, которые не зависят от t. Затем поделим обе части равенства на $n\mu(n-1)\mu\prod_{l=0}^{k-n-1}(n\mu+(l+1)\gamma)$, вычислим производную и подставим в ДУ Колмогорова. Получим следующее выражение:

$$\sum_{a=0}^{k-n+2} \frac{s_a}{B'_{n-2}(s_a)} e^{s_a t} = -\sum_{a=0}^{k-n+2} \frac{(n-2)\mu}{B'_{n-2}(s_a)} e^{s_a t} + \sum_{a=0}^{k-n+1} \frac{1}{B'_{n-1}(s_a)} e^{s_a t}$$

Переносим слагаемое с минусом в левую часть равенства и группируем в одну сумму:

$$\sum_{a=0}^{k-n+2} \frac{s_a+(n-2)\mu}{B'_{n-2}(s_a)} e^{s_a t} = \sum_{a=0}^{k-n+1} \frac{1}{B'_{n-1}(s_a)} e^{s_a t}$$

Теперь рассмотрим знаменатель суммы из левой части $B'_{n-2}(s)$. Несложно заметить, что это выражение не что иное, как:

$$B'_{n-2}(s) = [B_{n-1}(s)(s+(n-2)\mu)]'$$

Воспользуемся формулой Лейбница для дифференцирования произведения:

$$B'_{n-2}(s) = B'_{n-1}(s)(s+(n-2)\mu) + B_{n-1}(s)(s+(n-2)\mu)'$$

Очевидно, что при подстановке корней s_a многочлена $B_{n-2}(s)$ в выражение для его производной правое слагаемое обращается в ноль. Тогда сумма принимает следующий вид:

$$\sum_{a=0}^{k-n+2} \frac{s_a + (n-2)\mu}{B'_{n-1}(s_a)(s_a + (n-2)\mu)} e^{s_a t} = \sum_{a=0}^{k-n+1} \frac{1}{B'_{n-1}(s_a)} e^{s_a t}$$

Сокращаем числитель и знаменатель:

$$\sum_{a=0}^{k-n+2} \frac{1}{B'_{n-1}(s_a)} e^{s_a t} = \sum_{a=0}^{k-n+1} \frac{1}{B'_{n-1}(s_a)} e^{s_a t}$$

Мы получили верное равенство. Таким образом, можно с уверенность сказать, что найденное выражение для $P_j(t)$ при $j < n$ является верным.

Объединяем все случаи и получаем окончательное решение системы ДУ Колмогорова для переходных вероятностей процесса $\upsilon(t)$:

$$\begin{cases} P_j(t) = e^{-(n\mu+(k-n)\gamma)t}, & j = k \\ P_j(t) = \dfrac{(e^{\gamma t}-1)^{k-j}}{(k-j)!\,\gamma^{k-j}} e^{-(n\mu+(k-n)\gamma)t} \prod\limits_{l=j-n+1}^{k-n} (n\mu + l\gamma), & n \le j < k \\ P_j(t) = \sum\limits_{a=0}^{k-j} \dfrac{\prod_{l=0}^{k-n-1}(n\mu + (l+1)\gamma)\ \prod_{l=j}^{n-1}\big((l+1)\mu\big)}{B'_j(s_a)} e^{s_a t}, & j < n \end{cases}$$

где

$$B_j(s) = (s + n\mu + (k-n)\gamma) \prod_{l=0}^{k-1-n} (s + n\mu + l\gamma) \prod_{l=j}^{n-1} (s + l\mu),$$

s_a – корни многочлена $B_j(s),\ a = 0,1,\dots,k-j$.

Теперь, когда были найдены переходные вероятности случайного процесса $\upsilon(t)$ можно выписать переходные вероятности для вложенной марковской цепи ξ_n. Эти вероятности связаны следующим соотношением:

$$P_{ij} = \int_0^\infty P_{ij}(t)\, dA(t)$$

Помним, что в нашем случае $k = i + 1$. Подставляем и получаем следующие выражения для переходных вероятностей вложенной ЦМ:

$$\begin{cases} P_{ij} = 0, \qquad j > i + 1 \\ P_{ij} = \int_0^\infty C_{i+1}^{i+1-j} \left(1 - e^{-\mu t}\right)^{i+1-j} e^{-j\mu t}\, dA(t), \qquad j \le i + 1 \le n \\ P_{ij} = \int_0^\infty e^{-(n\mu + (i+1-n)\gamma)t}\, dA(t), \qquad n < j = i + 1 \\ P_{ij} = \int_0^\infty \frac{(e^{\gamma t} - 1)^{i+1-j}}{(i+1-j)!\,\gamma^{i+1-j}} e^{-(n\mu + (i+1-n)\gamma)t} \prod_{l=j-n+1}^{i+1-n} (n\mu + l\gamma)\, dA(t), \qquad n \le j < i + 1 \\ P_{ij} = \int_0^\infty \sum_{a=0}^{i+1-j} \frac{\prod_{l=0}^{i-n}(n\mu + (l+1)\gamma)\ \prod_{l=j}^{n-1}\left((l+1)\mu\right)}{B'_{ij}(s_a)} e^{s_a t}\, dA(t), \qquad j < n < i + 1 \end{cases}$$

где

$$B_{ij}(s) = (s + n\mu + (i + 1 - n)\gamma) \prod_{l=0}^{i-n}(s + n\mu + l\gamma) \prod_{l=j}^{n-1}(s + l\mu),$$

s_a – корни многочлена $B_j(s),\ a = 0,1, \dots, i + 1 - j.$

Переходные вероятности вложенной марковской цепи ξ_n можно считать найденными.

2.3.2 Функции распределения срока окупаемости

Вспомним, что случайный процесс чистой гибели $\upsilon(t)$ показывает число заявок в системе в момент времени t в интервале времени $(t_n; t_{n+1})$ между приходом двух соседних требований и имеет множество состояний $E = \{0,1,2, \dots, k\},\ k < \infty.$

Пусть $W_i(t)$ – доход от работы системы, полученный за время t при старте из состояния i, который рассчитывается на основе вышеперечисленных компонент. Выпишем условные функции распределения срока окупаемости для случайного процесса $\xi(t)$. Пусть η – время до накопления дохода S. Вспомним, что $A(t)$ есть ФР времени между приходом соседних требований в систему. Обозначим за $F_i(t,x)$ вероятность того, что $\eta < t$, учитывая, что система начинает работу из состояния i и для того, чтобы заработать доход S осталось получить прибыль x.

Для вероятностей $F_i(t,x), i = 0,1,2,\ldots$ при помощи формулы полной вероятности можно выписать систему интегральных уравнений. Здесь имеет место три случая: система не вышла из состояния i и получила необходимый доход x, система вышла из состояния i и получила необходимый доход x, система вышла из состояния i и не получила необходимый доход x. Можно выписать выражение для $F_i(t,x)$:

$$F_i(t,x) = P(W_i(t) \geq x)\bar{A}(t) + \int_0^t P(W_i(u) \geq x)\, A(u) +$$

$$+ \int_0^t \int_{-\infty}^{x} F_j(t-u, x-y)\, dP(W_i(u) < y)\, P_{ij} dA(u),$$

где P_{ij} – переходные вероятности вложенной марковской цепи ξ_n.

Теперь рассмотрим вероятность $P(W_i(t) < x)$. Напомним, что $W_i(t)$ описывает доход на траекториях процесса чистой гибели. Пусть u – момент прихода новой заявки в систему $(0 < u < t)$. Рассмотрим несколько случаев.

1. Для $0 < i \leq n$.

Доход, который система получает в состоянии i, складывается либо из платы за обслуживание приборов, если заявка не обслужилась, либо из платы за обслуживание приборов и прибыли за обслуживание заявки, если заявка обслужилась. В первом случае:

$$W_i(t) = -C_3 t.$$

Этот доход должен быть меньше, чем x: $-C_3 t < x$. Вероятность того, что система не перейдет в другое состояние и получит доход меньше, чем x:

$$P_1 = e^{-i\mu t} * \begin{cases} 0, & x \le -C_3 t \\ 1, & x > -C_3 t \end{cases}$$

Во втором случае:

$$W_i(t) = C_0 - C_3 t.$$

Этот доход должен быть меньше, чем x: $C_0 - C_3 t < x$ Вероятность того, что система перейдет в другое состояние и получит доход меньше, чем x:

$$P_2 = \begin{cases} 0, & x \le C_0 - C_3 t \\ \int\limits_0^t P(W_{i-1}(t-u) < x - C_0 + C_3 u) i\mu e^{-i\mu u}\, du, & x \ge C_0 \\ \int\limits_{\frac{C_0 - x}{C_3}}^t P(W_{i-1}(t-u) < x - C_0 + C_3 u) i\mu e^{-i\mu u}\, du, & C_0 - C_3 t < x < C_0 \end{cases}$$

Тогда итоговое выражение находится по формуле полной вероятности:

$$P(W_i(t) < x) = e^{-i\mu t} * \begin{cases} 0, & x \le -C_3 t \\ 1, & x > -C_3 t \end{cases} +$$

$$+ \begin{cases} 0, & x \le C_0 - C_3 t \\ \int\limits_0^t P(W_{i-1}(t-u) < x - C_0 + C_3 u) i\mu e^{-i\mu u}\, du, & x \ge C_0 \\ \int\limits_{\frac{C_0 - x}{C_3}}^t P(W_{i-1}(t-u) < x - C_0 + C_3 u) i\mu e^{-i\mu u}\, du, & C_0 - C_3 t < x < C_0 \end{cases}$$

2. Для $n < i \le k$.

Доход, который система получает в состоянии i, складывается либо из платы за обслуживание приборов и штрафов за ожидание заявок в очереди, если заявка не

обслужилась, и нетерпеливый клиент не ушел, либо из платы за обслуживание приборов, штрафов за ожидание заявок в очереди и прибыли за обслуживание заявки, если заявка обслужилась, а нетерпеливый клиент не ушел, либо из платы за обслуживание приборов, штрафов за ожидание заявок в очереди и штрафа за уход нетерпеливого клиента, если заявка не обслужилась, а нетерпеливый клиент ушел. В первом случае:

$$W_i(t) = -C_2(i-n)t - C_3 t.$$

Этот доход должен быть меньше, чем x: $= -C_2(i-n)t - C_3 t < x$. Вероятность того, что система не перейдет в другое состояние и получит доход меньше, чем x:

$$P_1 = e^{-(n\mu+(i-n)\gamma)t} * \begin{cases} 0, & x \le -C_2(i-n)t - C_3 t \\ 1, & x > -C_2(i-n)t - C_3 t \end{cases}$$

Во втором случае:

$$W_i(t) = C_0 - C_2(i-n)t - C_3 t.$$

Этот доход должен быть меньше, чем x: $C_0 - C_2(i-n)t - C_3 t < x$. Вероятность того, что система перейдет в другое состояние и получит доход меньше, чем x:

$$P_2 = \begin{cases} 0, & x \le C_0 - C_2(i-n)t - C_3 t \\ \int_0^t P(W_{i-1}(t-u) < x - C_0 + C_2(i-n) + C_3 u) n\mu e^{-(n\mu+(i-n)\gamma)u}\, du, & x \ge C_0 \\ \int_{\frac{C_0-x}{C_2(i-n)+C_3}}^t P(W_{i-1}(t-u) < x - C_0 + C_2(i-n) + C_3 u) n\mu e^{-(n\mu+(i-n)\gamma)u}\, du, & C_0 - C_2(i-n)t - C_3 t < x < C_0 \end{cases}$$

В третьем случае:

$$W_i(t) = -C_1 - C_2(i-n)t - C_3 t.$$

Этот доход должен быть меньше, чем x: $-C_1 - C_2(i-n)t - C_3 t < x$. Вероятность того, что система перейдет в другое состояние и получит доход меньше, чем x:

$$P_3 =$$

$$= \begin{cases} 0, & x \le -C_1 - C_2(i-n)t - C_3 t \\ \int_0^t P(W_{i-1}(t-u) < x + C_1 + C_2(i-n) + C_3 u)(i-n)\gamma e^{-(n\mu+(i-n)\gamma)u}\, du, & x \ge -C_1 \\ \int_{\frac{-C_1-x}{C_2(i-n)+C_3}}^t P(W_{i-1}(t-u) < x + C_1 + C_2(i-n) + C_3 u)(i-n)\gamma e^{-(n\mu+(i-n)\gamma)u}\, du, & -C_1 - C_2(i-n)t - C_3 t < x < -C_1 \end{cases}$$

Тогда итоговое выражение находится по формуле полной вероятности:

$$P(W_i(t) < x) = P_1 + P_2 + P_3.$$

Таким образом, была выписана система интегральных уравнений для $F_i(t,x), i = 0,1,2,\ldots$ Нахождение решения этой систем в явном виде является сложной задачей. Поэтому далее предлагается решать задачу, используя имитационное моделирование. С помощью имитационного модели будут посчитаны средние сроки окупаемости системы с заданными параметрами для разных долей нетерпеливых клиентов от общего количества заявок. Далее для каждого случая будет посчитан средний срок окупаемости в зависимости от управления. После анализа полученных результатов будет определена оптимальная стратегия управления входящим потоком заявок (стратегия, для которой средний срок окупаемости минимальный).

Прежде сформулируем задачу оптимизации. В качестве критерия эффективности выбрано среднее время до накопления заданного дохода. Пусть функционал $W(t)$ равен доходу системы в момент времени t. Пусть случайная величина η – время до накопления суммы S, тогда

$$\eta = min(t \mid W(t) = S).$$

Теперь введем функционал достижения – условное математическое ожидание случайной величины η:

$$L_i = E(\eta \mid \xi(0) = i), \qquad i \in E$$

Необходимо, чтобы математическое ожидание времени функционирования системы до заработка суммы S было минимальным. Тогда задача оптимизации выглядит следующим образом:

$$L_0 = E(\eta \mid \xi(0) = 0) \to \min_{p_v}.$$

Функционал достижения L является дробно-линейным относительно вероятностных мер, определяющих марковскую однородную рандомизированную стратегию управления [9]. Также согласно теореме из работы [8] известно, что если максимум дробно-линейного функционала существует, то оптимальную стратегию управления можно искать в классе вырожденных стратегий. Также сформулируем и докажем следующее утверждение.

Утверждение 2.

Оптимальную стратегию управления можно искать в классе вырожденных пороговых стратегий. То есть вектор $\{p_v\}_{v \in E}$ будет состоять из нулей и единиц, причем существует такое v^*, что если $v < v^*$, то $p_v = 1$, иначе $p_v = 0$.

Доказательство:

Пусть есть вектор $\{p_v\}$, который определяет вырожденную стратегию управления. Так как он состоит только из нулей и единиц, то на какой-то позиции v^* будет стоять первый ноль. Например: $\{1,0,\dots\}$, $\{1,1,0,\dots\}$, $\{1,1,1,0,\dots\}$, $\{1,1,\dots,1,0,\dots\}$.

Поскольку $p_{v^*} = 0$, то пришедшая $v^* + 1$ заявка будет отклонена. Это означает, что система никогда не перейдет в состояние выше, чем v^*. Значит, на все позиции, начиная с v^*, в векторе $\{p_v\}$ можно поставить нули. Таким образом, получаем пороговую вырожденную стратегию управления. ■

2.4 Построение имитационной модели

Имитационное моделирование это процесс создания и анализа цифровой модели некоторого физического объекта с целью предсказать или изучить поведение этого объекта в реальном мире. Такой метод исследования является достаточно удобным для изучения сложных систем, в том числе систем массового обслуживания. В данной работе имитационная модель исследуемой системы будет написана на языке программирования Python.

Первым этапом будет построение блок-схемы функционирования системы. Блок-схема представлена на рисунке 2:

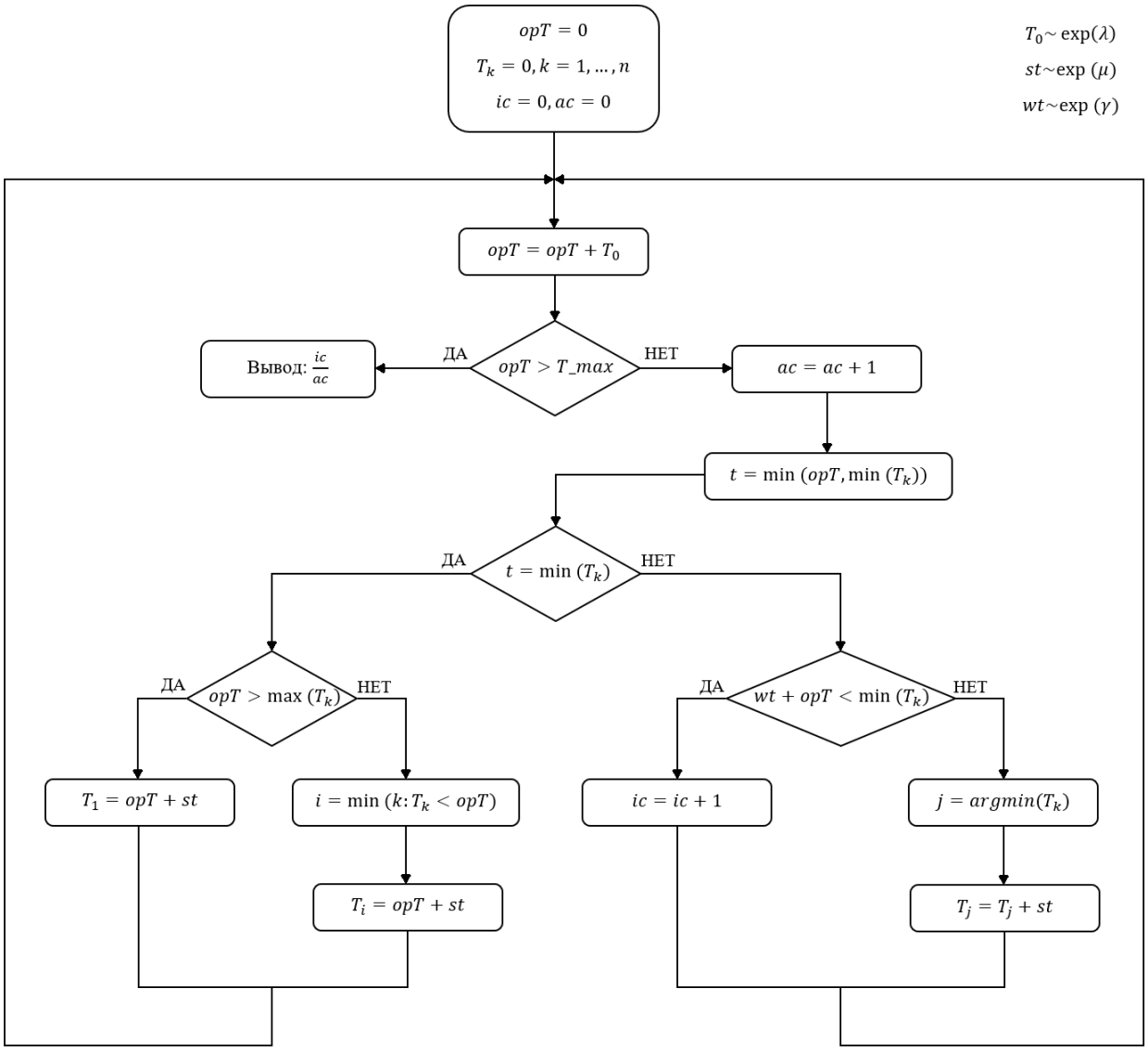


Рисунок. 2. Блок-схема функционирования системы.

Здесь opT – это время работы системы, T_k – очередные моменты освобождения k-ого канала, ic – количество нетерпеливых клиентов, ac – общее количество клиентов, пришедших в систему (далее подсчитаем вероятность потери клиента).

Блок-схема отображает функционирование системы для нахождения вероятности потери клиента. Однако функционирование системы с выходом из цикла к моменту накопления заданного уровня дохода (и подсчетом штрафов, прибыли) и окончанием работы по истечении заданного времени отличаются несущественно.

Вторым этапом будет написание программы, иллюстрирующей поведение системы. Сначала импортируем необходимые библиотеки:

```
In [1]: # импортируем необходимые библиотеки
        import numpy as np
        import random
        import matplotlib.pyplot as plt
```

В языке программирования Python есть библиотека $random$, содержащая все необходимые функции для моделирования случайных величин с разными функциями распределения (экспоненциальное, гамма и так далее). Однако во многих языках программирования подобных библиотек нет. В этом случае можно воспользоваться методом обратных функций [7], который удобен тем, что имея генератор равномерно распределенных чисел, можно построить генератор случайных величин с нужной функцией распределения. Суть этого метода заключается в следующем. Пусть есть строго возрастающая функция $F : \mathbb{R} \to [0,1]$ на всей своей области определения, тогда существует обратная ей функция $F^{-1} : [0,1] \to \mathbb{R}$. Пусть $U_1, \dots, U_n \sim U[0,1]$ есть выборка из стандартного непрерывного равномерного распределения. Тогда $X_1, \dots X_n$, где $X_i = F^{-1}(U_i)$, $i = 1, \dots, n$, есть выборка из интересующего нас распределения. Если функция F не является строго возрастающей, то используются другие инструменты.

В данной работе изучается система с произвольной функцией распределения интервалов между приходами заявок в систему. Для моделирования возьмем экспоненциальное распределения с заданным параметром λ. Для начала напишем

функцию $system1$ (код функции см. в приложении В), которая будет просто моделировать работу системы до истечения какого-то момента времени T_max. Для примера были выбраны следующие параметры системы: $\lambda = 5,\ \mu = 2,\ \gamma = 1,\ n = 4$. Здесь $\lambda,\ \mu,\ \gamma$ имеют размерность обратную времени, например: мин$^{-1}$, час$^{-1}$, день$^{-1}$ и так далее. Чтобы убедиться, что программа написана верно, можно, например, посчитать вероятность потери клиента аналитически и имитационно, а затем сравнить результаты. Чтобы посчитать эту вероятность имитационно достаточно поделить количество нетерпеливых клиентов на общее число пришедших в систему заявок: $P_{lc} = \frac{N_{ic}}{N_{ac}}$. Тогда статистическую оценку P_{lc} можно получить как оценку выборочным средним математического ожидания соответствующей случайной величины. Выборочное среднее является несмещенной и сильно состоятельной оценкой теоретического среднего. Будем использовать этот способ для получения оценки P_{lc} и всех других величин в последующих случаях. Прогоним программу 100 раз для $T\_max = 10000$ и посчитаем среднее значение:

```
In [3]: Plc = 0
        for i in range(100):
            Plc += system1(10000) / 100
        print('Plc =', Plc)

        Plc = 0.048218071292319115
```

Получим вероятность потери клиента: $P_{lc} \approx 0.04822$.

Теперь посчитаем аналитически. Так как процесс является ПГР при экспоненциальном распределении интервалов между приходом требований, то из теории массового обслуживания известно следующее выражение для вероятности потери клиента:

$$q = \frac{\gamma}{\lambda} E\chi = \frac{\gamma}{\lambda} \sum_{m=1}^{\infty} m\pi_{m+n},$$

где $E\chi$ – математическое ожидание длины очереди, а $\pi_i - i$-ая компонента предельного распределения, которая определяется по следующей формуле:

$$\pi_i = \frac{\lambda^j}{\mu^j n! \prod_{k=1}^{j-n}(n+\frac{\gamma}{\mu}k)} \frac{1}{\sum_{i=0}^{n}\frac{\lambda^i}{\mu^i i!} + \sum_{i=n+1}^{\infty}\frac{\lambda^i}{\mu^i n! \prod_{k=1}^{i-n}(n+\frac{\gamma}{\mu}k)}}, \qquad j > n.$$

Посчитаем в несколько этапов с помощью Wolfram Alpha:

$$\sum_{i=0}^{n}\frac{\lambda^i}{\mu^i i!} + \sum_{i=n+1}^{\infty}\frac{\lambda^i}{\mu^i n! \prod_{k=1}^{i-n}(n+\frac{\gamma}{\mu}k)} = \sum_{i=0}^{4}\frac{2.5^i}{i!} + \sum_{i=5}^{\infty}\frac{2.5^i}{24\prod_{k=1}^{i-4}(4+0.5k)} \approx 12.5546,$$

$$\sum_{m=1}^{\infty} m\pi_{m+n} = \sum_{m=1}^{\infty}\frac{\lambda^{m+n}}{\mu^{m+n}n!\prod_{k=1}^{m}\left(n+\frac{\gamma}{\mu}k\right)}\frac{m}{12.5546} =$$

$$= \sum_{m=1}^{\infty}\frac{2.5^{m+4}}{24\prod_{k=1}^{m}(4+0.5k)}\frac{m}{12.5546} \approx 0.24251,$$

$$q = \frac{0.24251}{5} \approx 0.048501$$

Относительная погрешность примерно равна 0.0058. Можно считать, что программа работает корректно.

Третьим этапом будет вычисление среднего срока окупаемости системы для заданных параметров. Компоненты дохода: C_0 – прибыль за обслуживание заявки, C_1 – штраф за уход требования из накопителя, C_2 – штраф в единицу времени за ожидание заявки в очереди, C_3 – затраты в единицу времени на обслуживание приборов, C_4 – штраф за недопуск заявки в систему. Добавим в функцию $system1$ подсчет дохода, который получает система. Для этого определим параметры системы: $\lambda = 12,\ \mu = 2,\ \gamma = 5,\ n = 4$ и компоненты дохода: $C_0 = 7, C_1 = 5, C_2 = 3, C_3 = 20$ и сумму, которую необходим накопить: $S = 5000$. Параметры были подобраны так, чтобы в среднем число нетерпеливых клиентов составляло примерно 40%. Таким образом, написана новая функция $system2$ (код функции см. в приложении С), которая тоже будет принимать один параметр T_max. Хотя в этот раз сигналом к выходу из бесконечного цикла $while$ будет накопленный доход S, параметр T_max

все равно необходим, чтобы прекратить работу цикла в случае возникновения каких-либо ошибок. Прогоним программу 1000 раз для $T\_max = 10000$ и посчитаем среднее значение срока окупаемости:

```
In [5]: S = []; T = []; C = []
        for i in range(1000):
            s1, t1, c1, = system2(10000)
            S.append(s1); T.append(t1); C.append(c1)
        print('Полученный доход S =', sum(S) / 1000)
        print('Средний срок окупаемости T =', sum(T) / 1000)
        print('Доля нетерпеливых клиентов Pic =', sum(C) / 1000)

        Полученный доход S = 5002.634222268097
        Средний срок окупаемости T = 1587.7554322938947
        Доля нетерпеливых клиентов Pic = 0.40109922698259887
```

Видим, что система накопила необходимый доход $S = 5000$, при этом средний срок окупаемости η равен примерно 1588 единицам времени.

Четвертый этап – это написание функции $system3$ (код функции см. в приложении D), которая будет вычислять средний срок окупаемости, но уже с учетом управления. Параметры системы остаются неизменными, и определяется еще одна компонента дохода: $C_4 = 2$. Напомним, что управление входящим потоком выглядит следующим образом: перед приходом нового требования в систему вычисляется состояние (подсчитывается количество заявок), в котором находится система, и на основании этого принимается решение о допуске или недопуске пришедшей заявки. С вероятностью p_v заявка принимается, а с вероятностью $1 - p_v$ отклоняется (при этом система платит штраф C_4), v – количество заявок в системе.

Функция $system3$ будет принимать два параметра: $threshold$ и T_max. Первый параметр определяет стратегию управления. Второй параметр нужен, чтобы обработать случай сбоя работы программы. Таким образом, задача состоит в том, чтобы найти средней срок окупаемости для разных значений параметра $threshold$ и на основании этого выбрать оптимальную стратегию управления. В рассматриваемом случае система имеет $n = 4$ каналов обслуживания, поэтому логично начинать со значения параметра $threshold = 5$. Вычислительный эксперимент показал, что начиная со значения параметра $threshold$ равного 20 средний срок окупаемости не

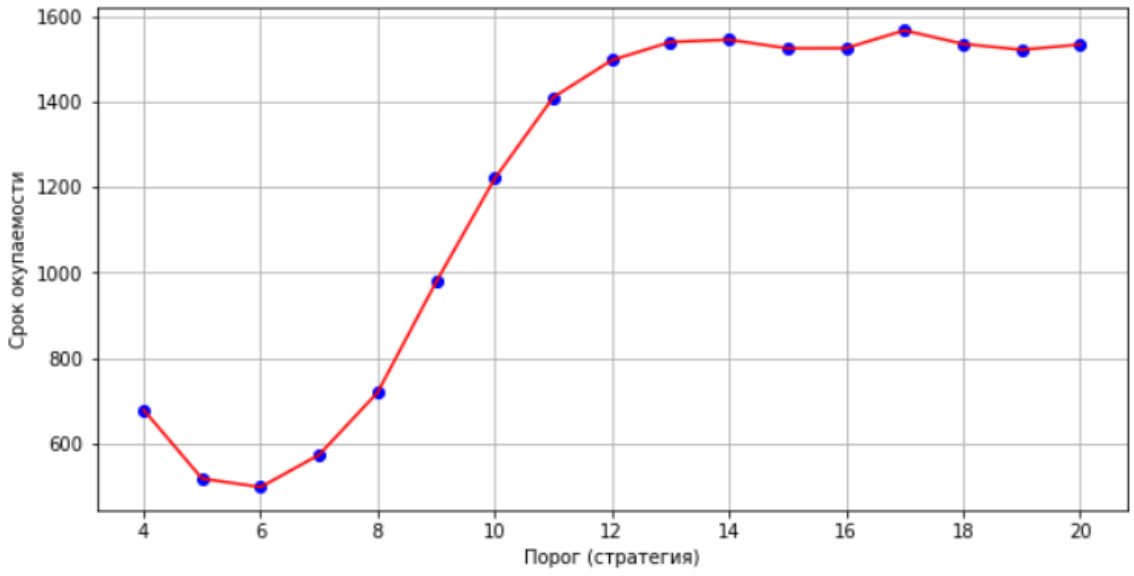
меняется, так как в системе практически не бывает более 20 заявок. В связи с этим было решено перебрать значения этого параметра в промежутке от 5 до 20 включительно и посчитать для каждого значения средний срок окупаемости, $T\_max = 2000$:

In [7]:
```
arr = []
for i in range(4, 21):
    t = 0
    for j in range(1000):
        t += system3(i, 2000)[1] / 1000
    arr.append(t)
print(arr)
```

```
[679.6357276927029, 518.9062912726683, 499.19633711424166, 574.6999944587061,
720.6852353945715, 978.8949376178502, 1220.2345782551888, 1409.8375275041353,
1496.6037521240494, 1538.9699975927813, 1544.441928227877, 1524.232724138061,
1524.6032744510298, 1566.1863028834323, 1534.5485711111592, 1520.90976083820
4, 1533.0318174997103]
```

По полученным данным можно построить график зависимости среднего срока окупаемости от стратегии управления (порога):

In [8]:
```
x = np.arange(4, 21)
plt.figure(figsize=(10, 5))
plt.plot(x, arr, 'bo')
plt.plot(x, arr, 'r-')
plt.xlabel('Порог (стратегия)')
plt.ylabel('Срок окупаемости')
plt.grid()
plt.show()
```



Из графика зависимости видно, что существует глобальный минимум с координатами $x = 6$ и $y \approx 500$, показывающий минимальный средний срок окупаемости системы для заданных параметров.

Таким образом, серия вычислительных экспериментов показала, что оптимальной стратегией управления из класса вырожденных пороговых стратегий для системы с выбранными параметрами является стратегия с порогом равным 6 (то есть $v^* = 6$), и средним сроком окупаемости равным 500 единиц времени. Далее будут проведены еще несколько экспериментов для других долей нетерпеливых клиентов.

Повторим все те же действия, но с другими данными. Параметры системы C, S, интенсивность обслуживания μ и количество каналов n оставим неизменными. Будем варьировать только интенсивность входящего потока λ и среднее время ожидания нетерпеливых клиентов в очереди γ. Пусть $\lambda = 10$, $\gamma = 3$. Параметры были подобраны так, чтобы доля нетерпеливых клиентов была примерно 30%:

In [9]:
```
S = []; T = []; C = []
for i in range(1000):
    s1, t1, c1, = system2(10000)
    S.append(s1); T.append(t1); C.append(c1)
print('Полученный доход S =', sum(S) / 1000)
print('Средний срок окупаемости T =', sum(T) / 1000)
print('Доля нетерпеливых клиентов Pic =', sum(C) / 1000)
```

```
Полученный доход S = 5002.633927900617
Средний срок окупаемости T = 480.81675731721344
Доля нетерпеливых клиентов Pic = 0.30326781158374766
```

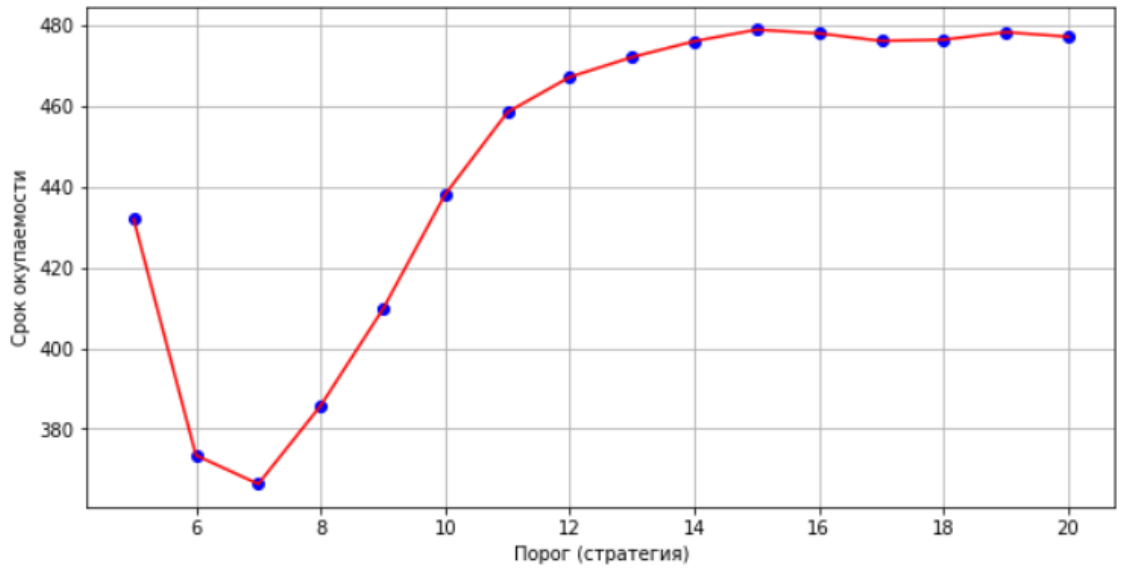
Видно, что средний срок окупаемости равен примерно 481 единице времени. Теперь добавим управление. Управлять входящим потоком будем по такому же принципу. Переберем значения параметра $threshold$ от 5 до 20 включительно и посчитаем для каждого из них средний срок окупаемости, $T\_max = 2000$:

In [10]:
```
arr = []
for i in range(5, 21):
    t = 0
    for j in range(1000):
        t += system3(i, 2000)[1] / 1000
    arr.append(t)
print(arr)
```

```
[431.9757955827831, 373.4140786617441, 366.2991249676423, 385.68224362480413,
409.6211256461241, 438.0968069515679, 458.4619155047927, 467.188086874933, 47
2.11267071165395, 476.0611857022643, 478.9206811014661, 478.0000394388659, 47
6.1234617240696, 476.43520021139244, 478.2920598319879, 477.16583361164675]
```

По полученным данным можно построить график зависимости среднего срока окупаемости от стратегии управления:

In [11]:
```
x = np.arange(5, 21)
plt.figure(figsize=(10, 5))
plt.plot(x, arr, 'bo')
plt.plot(x, arr, 'r-')
plt.xlabel('Порог (стратегия)')
plt.ylabel('Срок окупаемости')
plt.grid()
plt.show()
```



Из графика видно, что оптимальная стратегия управления имеет порог равный 7 со средним сроком окупаемости 366 е. в. Теперь пусть $\lambda = 8$, $\gamma = 2$. Параметры были подобраны так, чтобы доля нетерпеливых клиентов была примерно 20%:

In [12]:
```
S = []; T = []; C = []
for i in range(1000):
    s1, t1, c1, = system2(10000)
    S.append(s1); T.append(t1); C.append(c1)
print('Полученный доход S =', sum(S) / 1000)
print('Средний срок окупаемости T =', sum(T) / 1000)
print('Доля нетерпеливых клиентов Pic =', sum(C) / 1000)
```

```
Полученный доход S = 5002.531126144343
Средний срок окупаемости T = 334.59036656500064
Доля нетерпеливых клиентов Pic = 0.19392839828439873
```

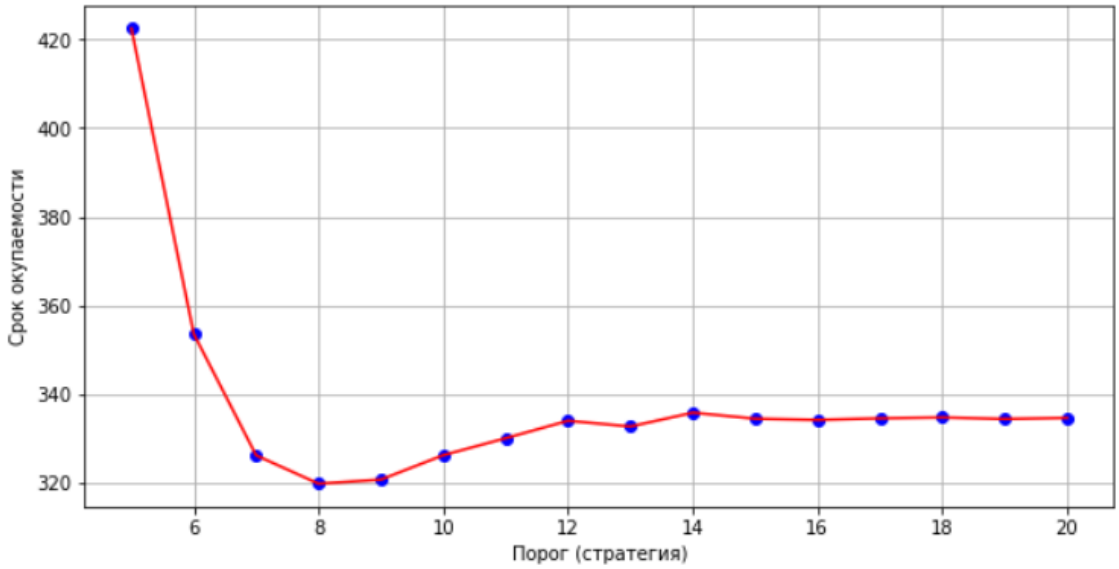
Видно, что средний срок окупаемости равен примерно 335 единицам времени. Аналогично:

In [13]:
```
arr = []
for i in range(5, 21):
    t = 0
    for j in range(1000):
        t += system3(i, 2000)[1] / 1000
    arr.append(t)
print(arr)
```

```
[422.516516654286, 353.5219996698188, 326.0878424726697, 319.68349409900475,
320.6122481429234, 326.1277280959982, 329.961376297483, 333.8984011409608, 33
2.6042761013321, 335.72703231648194, 334.37155473679445, 334.09562040380956,
334.4601533233938, 334.6605631004629, 334.31664269279963, 334.5607296679668]
```

По полученным данным можно построить график зависимости среднего срока окупаемости от стратегии управления:

In [14]:
```
x = np.arange(5, 21)
plt.figure(figsize=(10, 5))
plt.plot(x, arr, 'bo')
plt.plot(x, arr, 'r-')
plt.xlabel('Порог (стратегия)')
plt.ylabel('Срок окупаемости')
plt.grid()
plt.show()
```



Из графика видно, что оптимальная стратегия управления имеет порог равный 8 со средним сроком окупаемости 320 е. в. Теперь пусть $\lambda = 6$, $\gamma = 2$. Параметры были подобраны так, чтобы доля нетерпеливых клиентов была примерно 10%:

In [15]:
```
S = []; T = []; C = []
for i in range(1000):
    s1, t1, c1, = system2(10000)
    S.append(s1); T.append(t1); C.append(c1)
print('Полученный доход S =', sum(S) / 1000)
print('Средний срок окупаемости T =', sum(T) / 1000)
print('Доля нетерпеливых клиентов Pic =', sum(C) / 1000)
```

```
Полученный доход S = 5002.479305574325
Средний срок окупаемости T = 373.28793861421923
Доля нетерпеливых клиентов Pic = 0.10499528343662207
```

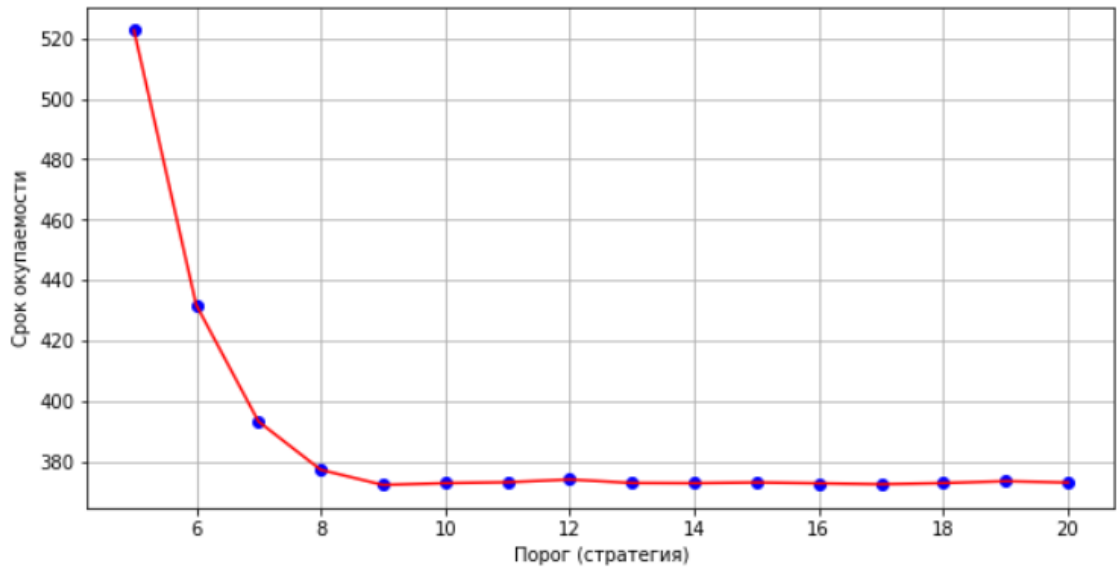
Видно, что средний срок окупаемости равен примерно 373 единицам времени. Аналогично:

In [16]:
```
arr = []
for i in range(5, 21):
    t = 0
    for j in range(1000):
        t += system3(i, 2000)[1] / 1000
    arr.append(t)
print(arr)
```

```
[522.6585713361894, 431.82455885605486, 393.27700055414266, 377.297639658799
6, 372.2815005373501, 372.8838333570684, 373.1794125895983, 374.091974111473
8, 372.93546856627455, 372.87597962772503, 373.10226030904715, 372.8361387257
767, 372.5196861705085, 372.9095750671276, 373.5348117841375, 373.08556934676
693]
```

По полученным данным можно построить график зависимости среднего срока окупаемости от стратегии управления:

In [17]:
```
x = np.arange(5, 21)
plt.figure(figsize=(10, 5))
plt.plot(x, arr, 'bo')
plt.plot(x, arr, 'r-')
plt.xlabel('Порог (стратегия)')
plt.ylabel('Срок окупаемости')
plt.grid()
plt.show()
```



Из графика видно, что средний срок окупаемости увеличивается, если ограничивать входящий поток заявок. Значит, оптимальной стратегией управления является та, при которой в систему допускается каждая пришедшая заявка (то есть порог отсутствует), и средний срок окупаемости равен 373 е. в. В главе 2.5 подведены итоги вычислительных экспериментов.

2.5 Анализ полученных результатов

Для имитационного моделирования были выбраны следующие параметры системы:

$$\mu = 2 \text{ е.в.}^{-1}, n = 4 \text{ шт.},$$

$$C_0 = 7 \text{ у.е.}, C_1 = 5 \text{ у.е.}, C_2 = 3 \text{ у.е.}, C_3 = 20\frac{\text{у.е.}}{\text{е.в.}}, C_4 = 2 \text{ у.е.},$$

$$S = 5000 \text{ у.е.},$$

а λ и γ варьировались для того, чтобы получить разные доли нетерпеливых клиентов (у. е. – условная единица: рубли, доллары и т. д., е. в. – единица времени: минуты, часы и т. д.). Были получены результаты, представленные в таблице 1:

Номер эксперимента	Доля нетерпеливых клиентов, %	Срок окупаемости без управления, е. в.	Порог оптимальной стратегии управления	Срок окупаемости при оптимальном управлении, е. в.
1	40	1588	6	500
2	30	481	7	366
3	20	335	8	320
4	10	373	-	373

Таблица 1. Результаты вычислительного эксперимента

Из таблицы 1 видно, что чем меньше доля нетерпеливых клиентов, тем больший порог имеет оптимальная стратегия управления. Также можно сделать вывод, что если доля нетерпеливых клиентов достаточно велика (более 20%), то эффективное управление существенно уменьшает средний срок окупаемости системы, для доли нетерпеливых клиентов 10-20% средний срок окупаемости при оптимальной

стратегии управления уменьшается несущественно, а для доли в 10% и менее оптимальной стратегией является та, при которой в систему допускается каждая пришедшая заявка.

Таким образом, с помощью имитационного моделирования было показано, что существует оптимальная стратегия управления входящим потоком заявок в классе вырожденных пороговых стратегий, позволяющая уменьшить средний срок окупаемости системы для выбранных параметров. Подведем итог. Все зависит от выбранных параметров и компонент дохода. Например, если доля нетерпеливых клиентов велика, а штрафы за их уход из очереди существенные (как и плата за ожидание заявок в очереди), то ограничение на число заявок в системе поможет сократить срок окупаемости. Если наоборот нетерпеливых клиентов мало и штрафы за их уход являются несущественными, то система быстрее получит необходимый доход, если будут допускаться все пришедшие заявки.

Заключение

В работе была рассмотрена система массового обслуживания типа G|M|n с бесконечной очередью и нетерпеливыми клиентами. Основной целью данного исследования было изучение возможности уменьшения среднего срока окупаемости этой системы за счет ограничения приема заявок. Были выполнены следующие этапы: построена математическая модель системы в виде случайного процесса; построена вложенная цепь Маркова; найдена матрица переходных вероятностей вложенной марковской цепи; выписаны выражения для функций распределения срока окупаемости; сформулирована задача оптимизации. Решение задачи оптимизации не удалось найти аналитически, поэтому на языке программирования Python была построена имитационная модель. Был рассмотрен частный случай: экспоненциальная функция распределения интервалов между приходами заявок в систему, заданные параметры системы и компоненты дохода. Были получены зависимости средних сроков окупаемости от стратегий управления для разных долей нетерпеливых клиентов. Анализируя результаты имитационного моделирования, можно сделать вывод, что оптимальная стратегия управления зависит от характеристик системы. Если нетерпеливых клиентов достаточно мало, а плата за их уход невелика, то минимальный срок окупаемости системы достигается без какого-либо управления (то есть вектор $\{p_v\}$ состоит только из единиц). Если наоборот – доля нетерпеливых клиентов существенна, и плата за их уход велика, то оптимальная стратегия управления резко сокращает средний срок окупаемости.

Список литературы

[1] Я. М. Агаларов, Оптимизация емкости основного накопителя в системе массового обслуживания типа G|M|1|K с дополнительным накопителем, Информ. и её примен., 2020, том 14, выпуск 2, 72–79.

[2] В. В. Афонин, В. В. Никулин, Оптимизация многоканальных систем массового обслуживания при больших загрузках, Вестн. Астрахан. гос. техн. ун-та. Сер. управление, вычисл. техн. информ., 2020, номер 3, 105–115.

[3] Я. М. Агаларов, М. Я. Агаларов, В. С. Шоргин, Об оптимальном пороговом значении длины очереди в одной задаче максимизации дохода системы массового обслуживания типа M/G/1, Информ. и её примен., 2016, том 10, выпуск 2, 70–79.

[4] Г. И. Ивченко, В. А. Каштанов, И. Н. Коваленко, Теория массового обслуживания: Учебное пособие для вузов. – М.: Высшая школа, 1982, 265 с.

[5] П. П. Бочаров, А. В. Печинкин, Теория массового обслуживания: Учебник. – М.: РУДН, 1995, 529 с.

[6] М. А. Плескунов, Операционное исчисление: Учебное пособие. – Екатеринбург.: Изд-во Уральского унив-та., 2014, 145 с.

[7] И. М. Соболь, Численные методы Монте-Карло: учебное пособие. – Изд-во «Наука», 1973, 303 с.

[8] В. А. Каштанов, А. И. Медведев, Теория надежности сложных систем. – М.: Физматлит, 2010

[9] Ю. Б. Гришунина, Исследование показателей безопасности в счетных полумарковских моделях сложных систем. – В кн.: Фундаментальные проблемы системной безопасности Вып. 5. Елец: Елецкий государственный университет им. И. А. Бунина, 2014. С. 24-30.

Приложения

Приложение А

Обратное преобразование Лапласа.

Input

$$\mathcal{L}_s^{-1}\left[\frac{1}{s+n\,m+(k-n)\,g}\times\frac{n\,m+(k-n)\,g}{s+n\,m+(k-1-n)\,g}\right](t)$$

$\mathcal{L}_s^{-1}[f(s)](t)$ is the inverse Laplace transform of $f(s)$ with positive real variable t

Alternate forms

$$\frac{\left(e^{g\,t}-1\right)(g\,(k-n)+m\,n)\,e^{t\,(-(g\,(k-n)+m\,n))}}{g}$$

Input

$$\mathcal{L}_s^{-1}\left[\frac{1}{s+n\,m+(k-n)\,g}\times\frac{n\,m+(k-n)\,g}{s+n\,m+(k-1-n)\,g}\times\frac{n\,m+(k-1-n)\,g}{s+n\,m+(k-2-n)\,g}\right](t)$$

$\mathcal{L}_s^{-1}[f(s)](t)$ is the inverse Laplace transform of $f(s)$ with positive real variable t

Alternate forms

$$\frac{\left(e^{g\,t}-1\right)^2(g\,(k-n-1)+m\,n)\,(g\,(k-n)+m\,n)\,e^{t\,(-(g\,(k-n)+m\,n))}}{2\,g^2}$$

Input

$$\mathcal{L}_s^{-1}\left[\frac{1}{s+n\,m+(k-n)\,g}\times\frac{n\,m+(k-n)\,g}{s+n\,m+(k-1-n)\,g}\times\frac{n\,m+(k-1-n)\,g}{s+n\,m+(k-2-n)\,g}\times\frac{n\,m+(k-2-n)\,g}{s+n\,m+(k-3-n)\,g}\right](t)$$

$\mathcal{L}_s^{-1}[f(s)](t)$ is the inverse Laplace transform of $f(s)$ with positive real variable t

Alternate forms

$$\frac{\left(e^{g\,t}-1\right)^3(g\,(k-n-2)+m\,n)\,(g\,(k-n-1)+m\,n)\,(g\,(k-n)+m\,n)\,e^{t\,(-(g\,(k-n)+m\,n))}}{6\,g^3}$$

Приложение В

Моделирование работы системы. Вероятность потери клиента.

In [2]:
```
# Функционирование системы. Вероятность потери клиента.

def system1(T_max):

    l = 5 # интенсивность входящего потока
    m = 2 # интенсивоность обслуживания
    g = 1 # "интенсивность" ожидания
    n = 4 # количество каналов

    opT = 0 # время работы системы
    T0 = 0 # момент прихода очередной заявки
    Tk = np.zeros(n) # очередные моменты освобождения k-ого канала

    ac = 0 # всего клиентов
    ic = 0 # количесвто нетерпеливых клиентов
    nsc = 0 # количество необслуженных клиентов
    sc = 0 # количество обслуженных клиентов: sc=ac-ic-nsc

    while True:

        T0 = random.expovariate(lambd=l) # пришла заявка
        opT += T0
        if opT >= T_max:
            sc = ac - ic - nsc
            break

        ac += 1

        # функционирование системы
        t = min(opT, min(Tk))
        if t == min(Tk): # канал освободился?
            i = np.nonzero(Tk < opT)[0][0]
            Tk[i] = opT + random.expovariate(lambd=m)
            if Tk[i] > T_max:
                nsc += 1

        else: # свободных каналов нет
            wt = random.expovariate(lambd=g)
            if wt + opT < min(Tk): # до освобождения канала больше времени,
                                   # чем готов ждать нетерпеливый клиент?
                ic += 1

            else: # нетерпеливый клиент успеет обслужиться
                j = np.argmin(Tk)
                Tk[j] += random.expovariate(lambd=m)
                if Tk[j] > T_max:
                    nsc += 1

    return ic / ac
```

Приложение С

Доход. Срок окупаемости. Без управления.

In [4]:
```
# Доход. Срок окупаемости. Нетерпеливые клиенты.

def system2(T_max):

    l = 12 # интенсивность входящего потока
    m = 2 # интенсивоность обслуживания
    g = 5 # количсевто нетерпеливых клиентов в ед. времени
    n = 4 # количество каналов

    S = 5000 # начальный капитал
    c0 = 7 # прибыль за обслуживание одной заявки
    c1 = 5 # штраф за уход требования
    c2 = 3 # штраф в единицу времени за ожидание заявки
    c3 = 20 # затраты в единицу времени на обслуживание приборов
    income = 0

    opT = 0 # время работы системы
    T0 = 0 # момент прихода очередной заявки
    Tk = np.zeros(n) # очередные моменты освобождения k-ого канала

    ac = 0
    ic = 0

    while True:

        T0 = random.expovariate(lambd=l)
        opT += T0
        if opT >= T_max or income >= S:
            break

        ac += 1
        income -= T0 * c3 # затраты на обслуживание системы

        # функционирование системы
        t = min(opT, min(Tk))
        if t == min(Tk): # канал освободился?
            i = np.nonzero(Tk < opT)[0][0]
            Tk[i] = opT + random.expovariate(lambd=m)
            income += c0 # прибыль за обслуживание

        else: # свободных каналов нет
            wt = random.expovariate(lambd=g)
            if wt + opT < min(Tk): # до освобождения канала больше времени,
                                   # чем готов ждать нетерпеливый клиент?
                ic += 1
                income -= c1 # штраф за ушедшего клиента
                income -= wt * c2 # штраф за ожидание в очереди

            else: # нетерпеливый клиент успеет обслужиться
                income -= (min(Tk) - opT) * c2 # штраф за ожидание в очереди
                j = np.argmin(Tk)
                Tk[j] += random.expovariate(lambd=m)
                income += c0 # прибыль за обслуживание

    return income, opT, ic / ac
```

Приложение D

Доход. Срок окупаемости. Управление.

In [6]:
```
# Доход. Срок окупаемости. Управление.

def system3(threshold, T_max):

    l = 12 # интенсивность входящего потока
    m = 2 # интенсивоность обслуживания
    g = 5 # количсевто нетерпеливых клиентов в ед. времени
    n = 4 # количество каналов

    S = 5000 # начальный капитал
    c0 = 7 # прибыль за обслуживание одной заявки
    c1 = 5 # штраф за уход требования
    c2 = 3 # штраф в единицу времени за ожидание заявки
    c3 = 20 # затраты в единицу времени на обслуживание приборов
    c4 = 2 # штраф за недопуск заявки в систему
    income = 0

    opT = 0 # время работы системы
    T0 = 0 # момент прихода очередной заявки
    Tk = np.zeros(n) # очередные моменты освобождения k-ого канала

    state = [0, 0] # количество заявок на каналах и в очереди
                   # перед приходом очередного требования
    L = 0 # длина очереди
    mQ = [] # время ухода клиента из очереди

    pk = 0 # вероятность допуска заявки в систему

    ac = 0 # всего клиентов
    ic = 0 # нетерпеливые клиенты
    rc = 0 # недопущенные к обслуживанию клиенты

    while True:

        # определение состояния системы
        if L == 0: # очередь пуста?
            state[0] = n - np.nonzero(Tk <= opT)[0].size # всего - свободные
            state[1] = L # длина очереди 0

        else: # очередь не пуста
            i = np.nonzero(np.array(mQ) <= opT)[0]
            j = np.nonzero(np.array(mQ) <= opT)[0].size
            mQ = list(np.delete(mQ, i))
            L -= j
            state[0] = n
            state[1] = L
```

```
        T0 = random.expovariate(lambd=l)
        opT += T0
        if opT >= T_max or income >= S:
            break

        ac += 1
        income -= T0 * c3 # затраты на обслуживание системы

        # управление
        if sum(state) < threshold:
            pk = 1
        else:
            pk = 0
        if pk == 0:
            income -= c4
            rc += 1
            continue

        # функционирование системы
        t = min(opT, min(Tk))
        if t == min(Tk): # канал освободился?
            i = np.nonzero(Tk < opT)[0][0]
            Tk[i] = opT + random.expovariate(lambd=m)
            income += c0 # прибыль за обслуживание

        else: # свободных каналов нет
            L += 1
            wt = random.expovariate(lambd=g)
            if wt + opT < min(Tk): # до освобождения канала больше времени,
                                   # чем готов ждать нетерпеливый клиент?
                income -= c1 # штраф за ушедшего клиента
                income -= wt * c2 # штраф за ожидание в очереди
                mQ.append(wt + opT)
                ic += 1

            else: # нетерпеливый клиент успеет обслужиться
                income -= (min(Tk) - opT) * c2 # штраф за ожидание в очереди
                income += c0 # прибыль за обслуживание

                j = np.argmin(Tk)
                Tk[j] += random.expovariate(lambd=m)

                mQ.append(min(Tk))

    return income, opT, ic / ac, rc
```