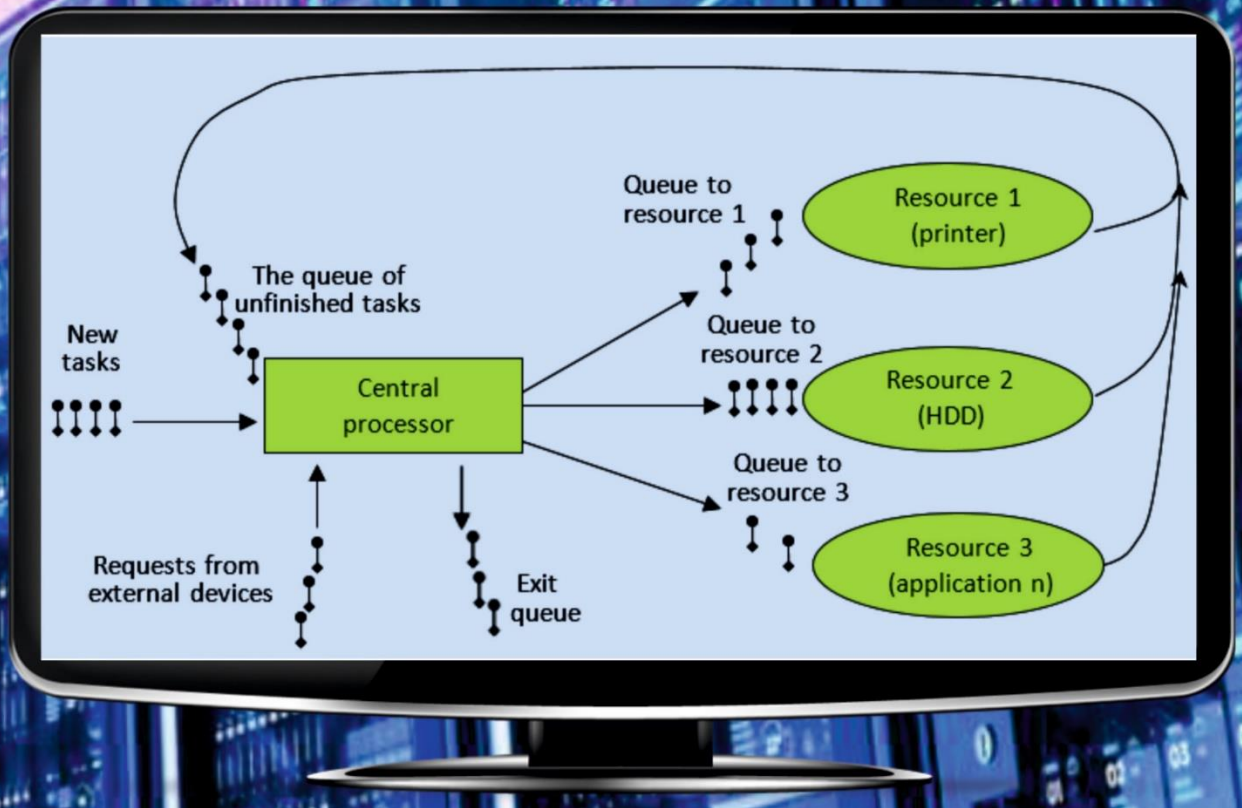
A. L. LITVINOV

QUEUEING SYSTEMS THEORY AND APPLICATIONS



MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

O. M. BEKETOV NATIONAL UNIVERSITY
of URBAN ECONOMY in KHARKIV

A. L. Litvinov

QUEUEING SYSTEMS
THEORY AND APPLICATIONS

MONOGRAPH

Kharkiv
O. M. Beketov NUUE
2026

UDC 519.872:004.6
L81

Author:

Anatoliy Leonidovich Litvinov, Doctor of Technical Sciences, professor of the department of computer science and information technology at the O. M. Beketov National University of Urban Economy in Kharkiv

Reviewers:

Fedorovych Oleh Yevhenovych, Doctor of Technical Sciences, professor, head of the department of computer science and information technology at the National Aerospace University "Kharkiv Aviation Institute" (KhAI);

Kostenko Oleksandr Borysovych, Candidate of Physical and Mathematical Sciences, Associate Professor of the Department of Computer Science and Information Technology at the O. M. Beketov NUUE

Recommended for publication by Academic Council of O. M. Beketov NUUE as a monograph (record № 9 of 27.04.2026)

У монографії викладено базові положення та методи аналізу систем масового обслуговування. Доведена необхідність використання теорії черг для аналізу ймовірнісних процесів у комп'ютерних системах. Наведено математичний апарат і базові методи, що використовується в теорії масового обслуговування. Проведені викладки доведено до розрахункових формул. Матеріал супроводжується значною кількістю реальних прикладів. Представлені у монографії теоретичні положення та практичні рекомендації можуть бути використані студентами, аспірантами, науковцями, практиками та іншими зацікавленими особами, які працюють у сфері застосування інформаційних технологій

Litvinov A. L.

L81 Queueing systems. Theory and applications : monograph / A. L. Litvinov ; O. M. Beketov National University of Urban Economy in Kharkiv. – Kharkiv : O. M. Beketov NUUE, 2026. – 177 p.

ISBN 978-966-695-655-5

The monograph presents the basic principles and methods of analysing queuing systems. The necessity of using queuing theory for analysing probabilistic processes in computer systems is proven. The mathematical apparatus and basic methods used in the theory of queuing are presented. The calculations are brought to the calculation formulas. The material is accompanied by a significant number of real examples. The theoretical principles and practical recommendations presented in the monograph can be used by students, postgraduates, scientists, practitioners and other interested persons working in the field of information technology applications.

UDC 519.872:004.6

ISBN 978-966-695-655-5

ЗМІСТ

INTRODUCTION 5

1 MATHEMATICAL BASICS OF QUEUE SYSTEMS THEORY. 10

1.1 Introduction to probability theory and mathematical statistics. 10
1.1.1 Basic concepts of probability theory 10
1.1.2 Discrete random variables. 14
1.1.3 Continuous random variables. 17
1.1.4 Mathematical statistics. 22
1.2 Principles of the theory of stochastic processes 31
1.2.1 Basic concepts of stochastic processes 31
1.2.2 Markov random processes with discrete time 34
1.2.3 Markov random processes with discrete states and continuous time. 38
1.3. Event flows and their characteristics 44

2 BASIC CONCEPTS OF QUEUING SYSTEMS AND METHODS OF THEIR RESEARCH 50

2.1 Basic concepts of queuing systems. 50
2.2 Performance indicators of queueing systems 54
2.2.1 Technical indicators of efficiency of queueing systems 54
2.2.2 Economic indicators of the efficiency of queueing systems 56
2.3 Methods of researching queueing systems 56
2.3.1 Differential method 56
2.3.2 Phase method 59
2.3.3 Imbedded Markov chains Method 63
2.3.4 Method of including additional variables. 67
2.3.5 Little's formulas 71

3 TYPICAL QUEUING SYSTEMS 73

3.1 Queuing system with failures 73
3.2 System with Poisson input flow and exponential service time (M/M/1) 77
3.3 Single-line system with an input flow of requests with hyperexponential distribution. 83
3.4 Multi-channel queuing system with limited queue. 87
3.5 Single-channel queuing system with arbitrary event streams. 93
3.6 Queuing system with a finite number of request sources 95
3.7 Absolute priority queueing system 100
3.8 Queueing systems networks 105

4 APPLICATION OF THE THEORY OF QUEUING SYSTEMS 111
4.1 Research and selection of parameters of the module for receiving pulse-number information. 111
4.2 Probabilistic modelling and assessment of the quality of functioning of information and management systems. 116
4.3 Probabilistic analysis and selection of I/O processor parameters for control computers. 124
4.4 Probabilistic modeling of the training test subdivision of device production 130
4.5 A queuing system with a finite number of sources as a model of a flexible manufacturing system. 137
5 SIMULATION MODELING OF QUEUEING SYSTEMS 148
5.1 Example and basic concepts of simulation modelling 148
5.2 Random Number Generation. 154
5.3 A Brief Overview of the SimPy Framework 156
5.4 Simulation of queueing systems using SimPy 163
REFERENCES. 169
ANNEX A Table of values of the Laplace function. 172
ANNEX Б Critical distribution points 173
ANNEX C Simulation model of the M/M/2/m queuing system[2]. 174

INTRODUCTION

In technical and organizational systems, when demand for processing or services exceeds the system's capacity, queues of various types form. A queue allows the system to "wait out" a short-term influx of requests without denying service to users. Developers don't need to build a system designed for a maximum peak (which occurs 1% of the time). Capacity for the average flow plus a buffer is sufficient. Let's consider the process of queue formation using the operation of computer systems as an example.

Modern computing systems consist of interconnected components: one or more processors, RAM, input and output devices (including a mouse, keyboard, scanner, and display), and external devices. These components can interact both jointly and independently, enabling multiprogramming under the control of the operating system. The operating system performs various tasks, including managing the execution of user programs, organizing input/output, processing external signals, and responding to emergency situations. The implementation of each task is carried out through the corresponding programs, called as needed. The basic functions of the operating system are provided by processes [1]. The concept of a process can be expanded to include the provision of user functions. Therefore, the execution of user programs is also a process. Several programs may be required to implement a process, and conversely, one program can serve multiple processes. For example, a sorting program can be used by any process working with a database. Different programs can simultaneously reside in RAM, initiating different processes and executing them in parallel. The term "resource" is used to denote any component that can be distributed within the system under the control of the operating system. Since the central processor is also a resource, its actions in executing a program constitute a process. Thus, a program is an ordered set of instructions, and a process is its execution on computer components. To start a process, a program issues a request. Typically, a resource can only handle one request at a time, and if a new request arrives while a request is being processed, it is placed in a queue. The mechanism for queuing requests to a resource is shown in Figure 1.

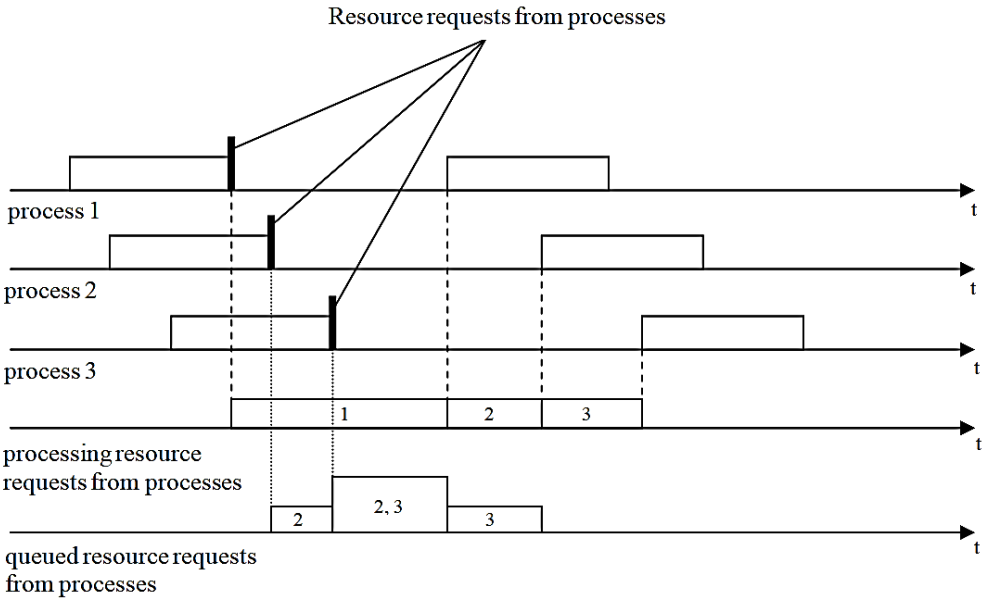


Figure 1 – Mechanism for forming a queue of resource requests

If a network printer receives print jobs from several computers of the network, one job is launched for printing, while the others wait in the queue to be processed. This creates separate queues of requests for different resources, as shown in Figure 2.

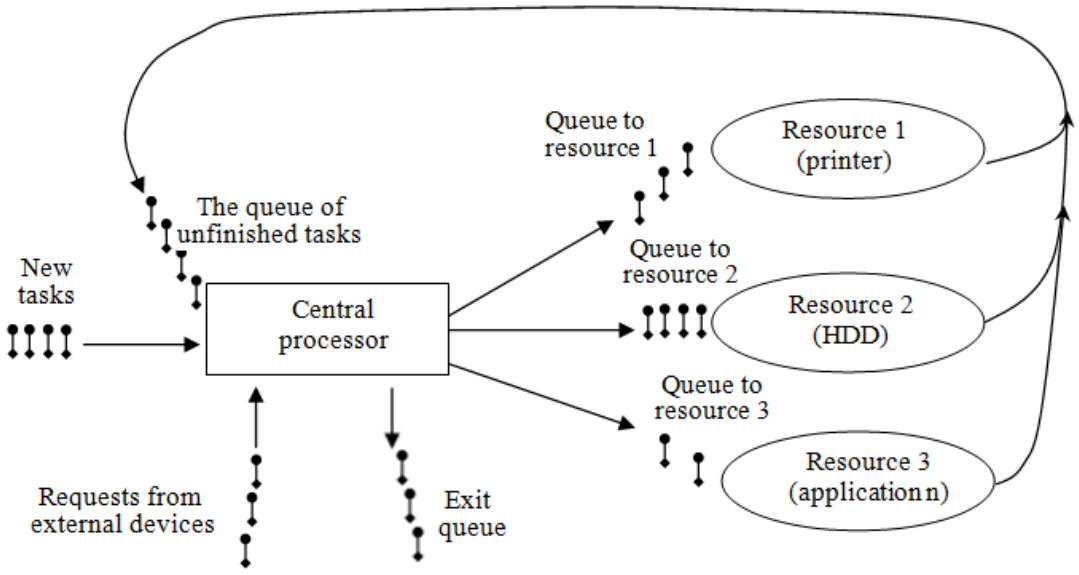


Figure 2 – Formation of queues for resources in a computer system

It's also worth paying attention to incoming requests from external devices, such as a mouse, keyboard, or communication modules with objects in real-time systems. These requests require a near-instantaneous response and are therefore given the highest processing priority. The priority levels of such requests can vary.

For example, a hard drive's read-read signal cannot tolerate even a brief pause, otherwise data loss is possible.

As can be seen from an analysis of Figure 1, resource access requests arise at random moments, and their execution times vary and are usually unpredictable. To synchronize the arrival and processing of requests and prevent their loss in computing systems, various queues are used, as shown in Figure 2. This figure depicts standard schemes with a massive influx of requests, one or more processing devices, and a waiting buffer. Such systems are called queueing systems. Queueing systems are ubiquitous in technology, science, and everyday life. A classic example is the line at the supermarket checkout, where cashiers act as service lines. At airports, passengers line up at check-in counters. The design of telecommunications and computer networks always takes into account the processes of mass service.

An examination of the actions shown in Figure 1 reveals the random nature of the flow of requests being processed. To compensate for this, developers often build additional capacity into the hardware. This is especially important for critical systems where downtime or lost requests are unacceptable, such as aircraft landing control systems [2]. As a result, the hardware is either overloaded or inoperative. Maintaining a buffer of requests in the queue also consumes resources. Therefore, queuing systems are associated with high costs, which has stimulated the development of a specialized theory for their analysis.

Queuing systems can be single-channel or multi-channel. For example, a single-channel system is a network printer that processes files from network computers only once. Or a single specialist queues computers in a department when they break down. A multi-channel system is illustrated by the work of a team of engineers in a company repairing broken computers: each repair request is processed by one of the team's specialists.

Queuing systems focuses on the study of real-world systems serving multiple clients. Researchers seek to understand how queue wait times and the likelihood of service failure depend on the intensity of incoming requests, their processing speed, and the nature of their distribution. The rules by which requests are selected, including the presence of priorities, significantly influence system behaviour. These rules include queue formation methods and request selection algorithms. Sometimes queues may not exist, and if all channels are busy, requests may be lost. Request selection occurs in various ways, one of which is the "first-in, first-out" (FIFO) principle. Other systems use the "last-in, first-out" (LIFO) principle.

Designing queuing systems is a much more complex task. Creating such systems requires addressing a multitude of complex issues. A key aspect is

selecting the optimal number of service channels based on the performance of each. Excessive number of channels can lead to extended downtime. The possibility of individual channel failure must also be considered. Even more complex challenges arise when combining multiple service systems into a network, where requests are processed sequentially at different stages. The development of such network approaches was driven by the advent of complex computing and telecommunications systems, including the Internet. In these networks, requests pass through multiple nodes and can change routes and processing algorithms, making their analysis difficult.

Figure 3 shows the process of receiving applications for processing in the system.

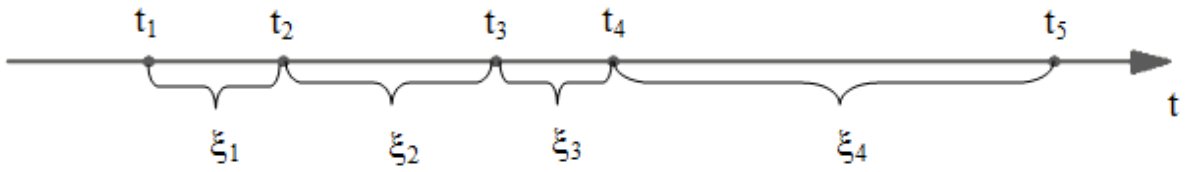


Figure 3 – The process of receiving applications for processing in the system

The timestamps of event arrivals are denoted as t_i where *i* takes the values 1, 2, and so on. The time intervals separating successive requests are defined as $\xi_1, \xi_2, \xi_3, \ldots$. These intervals are random, which entails a stochastic mode of operation of the queuing system. The characteristics of the incoming request flows significantly affect the functioning of the system. A flow can be ordinary, when only one request arrives at a particular time, or extraordinary, arriving either from a finite or infinite source of requests. Therefore, for queueing systems, a detailed flow analysis using probabilistic methods is necessary.

Figure 4 schematically shows the process of service requests arriving (indicated by points t_{ai} on the t_a axis), the process of completing service requests (indicated by points t_{ci} on the t_c axis), as well as the dynamics of changes in the number of requests in the service system L_s for a single-channel system.

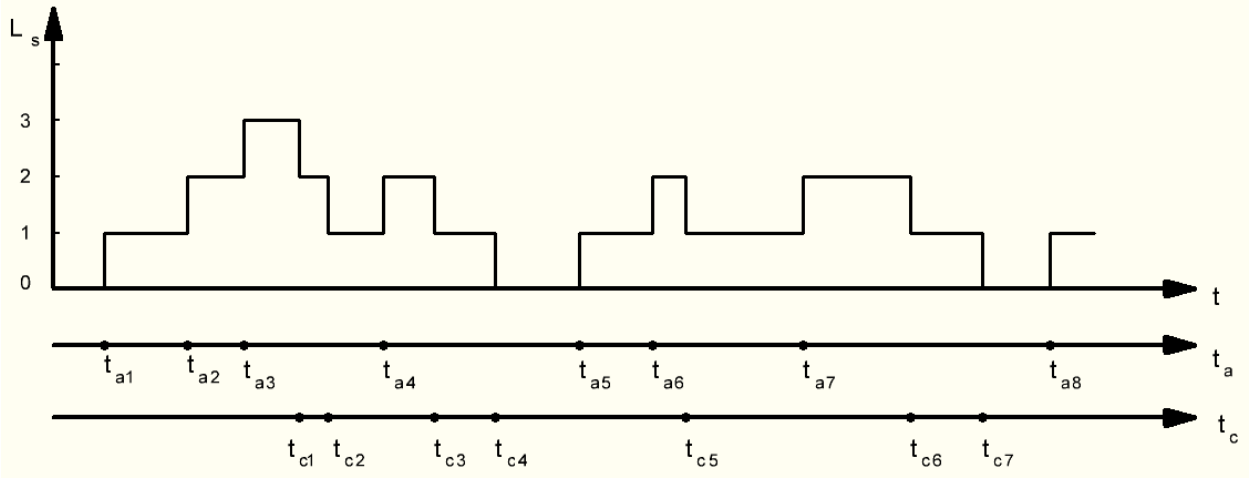


Figure 4 – Stochastic processes in a single-channel queuing system

The study presented in the diagram (Figure 4) clearly demonstrates that the operation of queuing systems is probabilistic in nature. Therefore, their analysis requires the use of the theory of random processes. Queueing systems theory is a distinct subfield of random process theory, forming an independent scientific field with its own terminology. The results obtained in this theory find wide application in various areas of human activity.

Numerous scientists have made significant contributions to the development of queuing systems. One of the pioneers was A. K. Erlang, who applied the methods of queuing system to study telephone networks. A significant contribution is due to L. Kleinrock, who applied queuing system to the analysis of computer systems [3]. Domestic scientists such as A. Ya. Khinchin, B. V. Gnedenko and I. M. Kovalenko also made an important contribution to the development of queuing system. The current state of queuing theory is reflected in the work [4]. The author of this work also participated in the development of queuing theory and its practical application. In particular, studies were conducted on a queuing system with a hyperexponential input flow, arbitrary service time, and a generalization of the famous Polyachek-Khinchin formula was obtained for this case. In [5], queueing systems were applied to calculate the parameters of a communication device with an object of computer systems operating in real time. The work [6] is devoted to the use of queueing systems for modeling complex production systems.

1 MATHEMATICAL BASICS OF QUEUE SYSTEMS THEORY

Queuing systems theory is a complex discipline based on a number of classical mathematical disciplines, namely probability theory, mathematical statistics, and the theory of random processes. Queuing processes can be described by systems of linear equations or systems of differential equations. Therefore, a researcher studying queuing systems should be familiar with these disciplines.

1.1 Introduction to probability theory and mathematical statistics

This subsection briefly presents the basic definitions and formulas of probability theory and mathematical statistics. A full coverage of this topic can be found in classic textbooks, such as, for example, the textbook by B. V. Gnedenko. [7]. The axiomatic foundations of probability theory are laid in the works of Academician A. N. Kolmogorov. Their essence is outlined in [8].

1.1.1 Basic concepts of probability theory

Probability theory in abstract form reflects the patterns inherent in random events (phenomena) of a mass nature. The idea of random events is formed on the basis of life experience. For example, we believe that such events as software glitch, device failure, and earthquake are random. So, in contrast to a reliable event, we call an event random if its occurrence cannot be guaranteed in advance, that is, if the event is characterized only by the fact that it is possible. The subject of probability theory is the study of probabilistic patterns of mass random phenomena.

The basic concepts of probability theory are a probabilistic experiment and an event. A probabilistic random experiment (test) is the implementation of a specific set of conditions, either artificially created or implemented independently of the experimenter's will, resulting in one event out of many possible ones. Let the products are produced in batches of n pieces each. Checking the quality of a product leads to its destruction, therefore, to check the quality of a batch, m products ($m<n$) are selected. The experiment consists of selecting items from a batch and testing them. The outcome (the result of the experiment) is the number of defective items.

In probability theory, events are typically denoted by capital letters of the Latin alphabet: *A, B, C,* etc. Events are classified according to a number of characteristics: compatible and incompatible, opposite, equally likely, and elementary [9]. Two basic operations are introduced over events: addition (union) and multiplication (section). **The sum** or union of events *A* and *B* is an event consisting of either A or B, or both simultaneously occurring. They are expressed

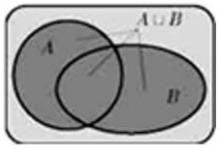


Figure 1.1

as follows: $A \cup B$ or A+B. The sum of the events consists of elementary events, each of which belongs to at least one of these events, as shown in Figure 1.1.

Example. Event A is when the dice rolls an even number of points, and event B is when no more than two points are rolled. Then $A + B = \{w_1, w_2, w_4, w_6\}$.

The product, or the intersection of two events is an event consisting of the joint occurrence of these events. It is denoted by: $A \cap B$ or AB. The product of two events consists of elementary events, each of which belongs to both events, as shown in Figure 1.2.

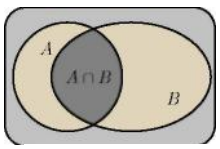


Figure 1.2

Example. Event A is when the dice rolls an even number of points, and event B is when at least two points roll. Then $AB = \{w_1, w_3, w_5\} \cdot \{w_2, w_3, w_4, w_5, w_6\} = \{w_3, w_5\}$.

Elementary events of a complete group of pairwise incompatible events are called favorable for the implementation of a composite event *A* if the implementation of any of these events entails event *A*.

Example. When throwing the event dice w_5, w_6 contribute to the occurrence of event B – the rolling of at least five points. The classical definition of probability is related to the latter definition.

Probability of an event A is the ratio of the number m of elementary events that favor the event *A* to the number *n* of all elementary events that form a complete group of events:

$$P(A) = \frac{m}{n}. \qquad (1.1)$$

The abbreviation *P* comes from the English word "probability," meaning likelihood. The following properties follow from the definition of probability: the probability of a certain event is one unit, the probability of an impossible event is zero, and the probability of a random event is a positive number between zero and one, i.e., $0 < P(A) < 1$. Thus, probability is a measure of randomness. Formula (1.1) is used to directly calculate probabilities in simple cases.

Example. A die is thrown. Let's find the probability of the outcome occurring if it lands on at least a three.

Solution. The total number of elementary events is $n = 6$, the number of elementary events favorable to event *A*, $m = 4$ ($\{w_3, w_4, w_5, w_6\}$), then the probability of event *A* – $P(A) = 4/6 = 0{,}667$.

If, when calculating the probability of an event *A*, no other restrictions are imposed except for conditions *S*, then such a probability is called unconditional; if other additional conditions are imposed, the probability of an event *A* is called conditional and is denoted as $P(A/B)$ or $P_B(A)$ – the probability of the occurrence of event *A* given that event *B* has already occurred.

Direct calculation of probabilities is extremely limited in its application and can be used only in the simplest cases. Random events, whose probabilities are predetermined, are combined into more complex events, the probabilities of which are calculated using the basic theorems of probability theory and a number of standard schemes for combining random events.

Product of Probabilities Theorem. The probability of the product of two events A and B is equal to the product of the probability of one of them and the conditional probability of the second, calculated with the assumption that the first event has already occurred:

$$P(A\cdot B)=P(A)P(B/A)=P(B)P(A/B). \qquad (1.2)$$

If event *B* does not depend on event *A* (P(B/A)=P(B)), then the probability of the product of two independent events is equal to the product of the probabilities of these events. That is, for independent events *A* and *B*, formula (1.2) can be rewritten

$$P(A\cdot B)=P(A)\cdot P(B). \qquad (1.3)$$

Probability addition theorem. The probability of the sum of two events *A* and B is equal to the sum of the probabilities of these events without the probability of their simultaneous occurrence:

$$P(A+B)=P(A)+P(B)-P(AB). \qquad (1.4)$$

Typically, both theorems must be applied simultaneously. In this case, the event whose probability needs to be determined is typically represented as the sum of several disjoint events (variants of the given event), each of which is in turn a product of events.

Example. Two shooters each fire one shot at a target. The probability of hitting the target for the first shooter is 0.8, for the second – 0.7. Determine the probability of hitting the target with one ball.

Solution. Let event *A* be the target being hit by one ball, *B* be the first shooter hitting the target, *C* be the second shooter hitting the target; $\overline{B}$ And $\overline{C}$ – events

opposite to events B and C, respectively. It is obvious that $A = B\bar{C} + \bar{B}C$. And $P(A) = P(B\bar{C} + \bar{B}C) = P(B\bar{C}) + P(\bar{B}C) = P(B)P(\bar{C}) + P(\bar{B})P(C)$. Because $P(B) = 0{,}8$, $P(C) = 0{,}7$, than $P(\bar{B}) = 1 - 0{,}8 = 0{,}2$, $P(\bar{C}) = 1 - 0{,}7 = 0{,}3$. So, $P(A) = 0{,}8 \cdot 0{,}3 + 0{,}2 \cdot 0{,}7 = 0{,}24 + 0{,}14 = 0{,}38.$

If one wants to determine the probability of an event *A* that can occur with one of *n* events, which are called hypotheses and form a complete group of incompatible events, one should use the formula for total probability

$$P(A) = \sum_{i=1}^{n} P(H_i)P(A/H_i). \qquad (1.5)$$

It should be taken into account that the events $H_1, H_2, ..., H_n$ in this experiment form a complete group of incompatible events and as a result of the experiment only one of them will necessarily occur. It is obvious that $P(H_1) + P(H_2) + ... + P(H_n) = 1.$

Example. Suppose two factories produce identical products, which are then delivered to a warehouse and mixed. The probability of defective products at the first factory is 0.1, while at the second factory it is 0.2. Determine the probability that a randomly selected product will be defective if the first factory produces 60% of the products and the second factory produces 40%.

Solution. Let us select the events for consideration: *A* – a randomly selected product is defective, H_1– the product was manufactured at the first plant, H_2 - the product was manufactured at the second plant. According to the terms of the assignment $P(H_1) = 0{,}6;\ P(H_2) = 0{,}4;\ P(A/H_1) = 0{,}1;\ P(A/H_2) = 0{,}2.$

So $P(A) = P(H_1)P(A/H_1) + P(H_2)P(A/H_2) = 0{,}6 \cdot 0{,}1 + 0{,}4 \cdot 0{,}2 = 0{,}14$.

Usually, in the conditions of the previous problem, it is necessary to determine the probability $P(H_k/A)$ that event A, which is considered to have occurred, occurred simultaneously with the event H_k. This probability is determined by Bayes' formula:

$$P(H_k / A) = \frac{P(H_k)P(A/H_k)}{\sum_{i=1}^{n} P(H_i)P(A/H_i)}. \qquad (1.6)$$

Example. According to the conditions of the previous problem, it is necessary to determine the following probability: if a randomly selected product turns out to be defective, then it was manufactured at the second plant.

Solution:

$$P(H_2 / A) = \frac{P(H_2)P(A/H_2)}{P(H_1)P(A/H_1) + P(H_2)P(A/H_2)} = \frac{0{,}4 \cdot 0{,}2}{0{,}6 \cdot 0{,}1 + 0{,}4 \cdot 0{,}2} = 0{,}556.$$

Most applications of probability theory involve random variables. A random variable is a quantity that, in an experiment, acquires one of its values, and which one is unknown in advance. For example, the number of students present in a class; the number of calls received at a telephone exchange during a given period of time. It is generally accepted that random variables are denoted by capital letters, and their possible values by corresponding lowercase letters of the Latin alphabet with subscripts. For example, X, Y, Z and, accordingly, x_i, y_j, z_k .A distinction is made between discrete (discontinuous) and continuous random variables.

A random variable is completely described by a distribution law, which establishes a one-to-one correspondence among the possible values of the random variable and its probabilities.

1.1.2 Discrete random variables

Discrete random variables can only take on a finite or countable set of values. For example, the number of defective items in a tested batch or the number of hits in n shots. One way to represent the distribution law of a discrete random variable is as a distribution series, written as a table listing the possible values of the random variable in ascending order and their corresponding probabilities (Table 1.1).

Table 1.1 – Representation of a discrete random variable

x	x_1	x_2	…	x_n
p	p_1	P_2	…	p_n

Here $p_i = P(X = x_i),\ i = 1,2,...,n$, besides $\sum_{i=1}^{n} p_i = 1$, since events $X = x_i\ (i = 1,2,...,n)$ form a complete group of incompatible events. The last equality is used to verify the correctness of the distribution series construction.

For clarity, a distribution series is often depicted graphically as a distribution polygon: the value of a random variable is plotted along the abscissa axis, the probabilities of these values are plotted along the ordinate axis, and the resulting points are connected by straight line segments, as shown in Figure 1.3.

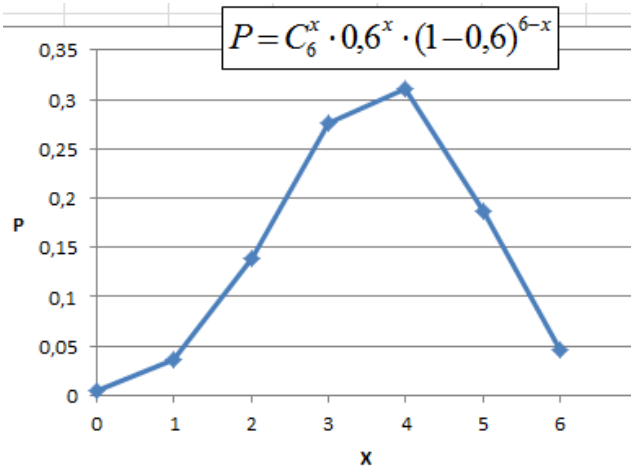


Figure 1.3 – Distribution polygon

The representation of a discrete random variable in the form of a table or distribution polygon becomes overwhelming with a large amount of data, therefore, to describe the entire set of values of a discrete random variable, a number of generalizing non-random numerical characteristics are used: mathematical expectation, dispersion, standard deviation, variation coefficient, mode.

The mathematical expectation of a random variable is its mean value. The mathematical expectation of a discrete random variable X is denoted as $M(X)$, $M[X]$, M_x, m_x or simply m, if it is known to what quantity it belongs, and is calculated using the formula:

$$m_x = \sum_{i=1}^{n} x_i p_i \tag{1.7}$$

Dispersion. The mathematical expectation of a random variable X is the squared deviation of the random variable relative to its mathematical expectation. The variance of a random variable X is denoted by D_x or simply D if the variable it refers to is known, and is calculated using the formula:

$$D_x = \sum_{i=1}^{n} (x_i - m_x)^2 p_i \,. \tag{1.8}$$

After simple transformations, we obtain another expression for calculating the variance:

$$D_x = \sum_{i=1}^{n} x_i^2 p_i - m_x^2 = \mu_{2x} - m_x^2 \,, \tag{1.9}$$

Where $\mu_{2x} = \sum_{i=1}^{n} x_i^2 p_i$ – the second initial moment – is another characteristic of the discrete random variable X used in complex scientific applications. Expression

(1.9) allows for a reduction in the number of computational operations compared to expression (1.8), which is essential when calculating manually.

Variance is expressed in squares of the unit of measurement of the random variable X, which can lead to confusion. For example, if the random variable is measured in meters, then the variance will be measured in square meters, which is the unit of measurement for area. Therefore, in addition to variance, the standard deviation, usually denoted as σ_x or σ and calculated using formulas

$$\sigma_x = \sqrt{D_x} = \sqrt{\sum_{i=1}^{n}(x_i - m_x)^2 p_i} = \sqrt{\sum_{i=1}^{n} x_i^2 p_i - m_x^2} = \sqrt{\mu_{2x} - m_x^2}\,. \qquad (1.10)$$

Note. In foreign literature and in computer mathematics systems, for example Maple, instead of the term root mean square deviation, the term standard deviation is used, which has the same meaning.

To assess the degree of dispersion of a random variable relative to the mathematical expectation in dimensionless quantities, the variation coefficient is used, calculated using the formula:

$$v_x = \frac{\sigma_x}{m_x}\,. \qquad (1.11)$$

The higher the value of the coefficient of variation, the greater the spread and the less uniformity of the random variable. The coefficient of variation is typically used to compare the dispersion of two or more features with different units of measurement.

The mode of a discrete random variable *X* is its most probable value.

The mathematical expectation has the following properties:

1. The mathematical expectation of a constant value C is equal to the constant value itself: $M(C) = C$.

2. The constant factor can be taken outside the sign of the mathematical expectation: $M(CX) = CM(X)$.

3. The mathematical expectation of the algebraic sum of two random variables is equal to the algebraic sum of the mathematical expectations of the corresponding random variables: $M(X \pm Y) = M(X) \pm M(Y)$.

4. The mathematical expectation of the product of two random variables is equal to the product of the mathematical expectations of the corresponding random variables: $M(X \cdot Y) = M(X) \cdot M(Y)$.

The dispersion has the following properties:

1. The dispersion of a constant value C is equal to zero: $D_x(C) = 0$.

2. The constant factor is taken out squared outside the dispersion sign: $D(CX) = C^2 D(X)$.

3. The variance of the algebraic sum of two random variables is equal to the sum of the variances of the corresponding random variables: $D(X \pm Y) = D(X) + D(Y)$.

4. If a constant is algebraically added to a random variable, the variance will not change: $D(X \pm C) = D(X)$.

1.1.3 Continuous random variables

Continuous random variables can take any value within a certain interval. For example, the deviations in the dimensions of parts from the required values, the weighing error of an object on an analytical balance, or the velocity of a moving object at a given moment in time. Continuous random variables are described by either a distribution function or a distribution density function.

The distribution function of a random variable *X* is a function $F(x)$, which is equal to the probability that this random variable in the experiment will take a value less than *x*, i.e.

$$F(x) = P(X < x).$$

The distribution function has the following properties, which follow from its definition:

- $0 \le F(x) \le 1$, that is, its values belong to the segment [0; 1].
- $F(+\infty) = 1.$
- $F(-\infty) = 0$.
- The distribution function F(x) is a non-decreasing function.

From the last property of the distribution function it follows: the probability that a random variable X will take a value from the interval (a, b) is equal to the increment of the distribution function on this interval:

$$P(a \le X < b) = F(b) - F(b). \qquad (1.12)$$

A typical distribution function graph is shown in Figure 1.4.

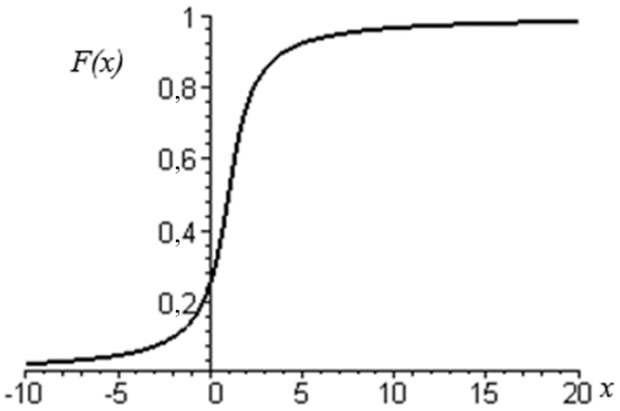


Figure 1.4 – Typical appearance of the distribution function graph

Distribution density of a continuous random variable X – is the derivative of its distribution function: $f(x) = F'(x)$. From here, $F(x) = \int_{-\infty}^{x} f(x)dx$. A typical graph of the distribution density of a random continuous variable (lognormal distribution) is shown in Figure 1.5.

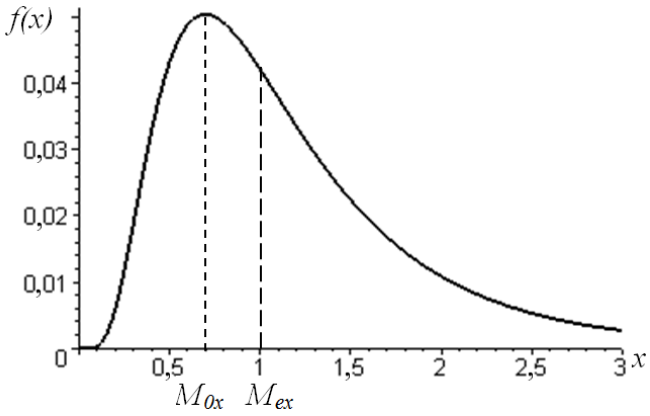


Figure 1.5 – Typical appearance of a distribution density graph

From the properties of the distribution density it follows that the probability that a continuous random variable *X* will take a value from the interval (*a*, *b*) is equal to a certain integral of the distribution density, taken within the range from *a* to *b*:

$$P(a < x < b) = \int_{a}^{b} f(x)dx. \qquad (1.13)$$

Hence, the area of a curvilinear trapezoid bounded by the axis $0,x$ and a distribution density curve equal to one.

As for discrete random variables, for continuous ones, generalizing non-random numerical characteristics are widely used: mathematical expectation, variance, standard deviation, coefficient of variation, which have the same meaning.

The mathematical expectation is calculated using the formula

$$m = \int_{-\infty}^{\infty} xf(x)dx. \tag{1.14}$$

The variance of a continuous random variable X is calculated using the formula:

$$D = \int_{-\infty}^{\infty} (x-m)^2 f(x)dx. \tag{1.15}$$

The standard deviation and the coefficient of variation are calculated in a similar way to the calculation of discrete random variables: $\sigma = \sqrt{D}$, $\nu = \sigma / m$.

It should be noted that for most distribution laws, expressions of these characteristics have already been found and given in the relevant textbooks and reference books, for example in [8, 10].

Although there are many potential distribution laws, only a small subset are widely used in practice. Particularly common are the normal, exponential, and uniform distribution laws.

Normal is the probability distribution of a continuous random variable X, which is described by distribution density

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-a)^2/2\sigma^2}. \tag{1.16}$$

Numerical characteristics: mathematical expectation – $m_x = a$, dispersion – $D_x = \sigma^2$, standard deviation – σ, fashion – $M_{ox} = a$, median – $M_{ex} = a$, the asymmetry coefficient is $As_x = 0$, the excess coefficient is $Ex = 0$.

The graph of the density of a normal distribution is called a normal curve (Gaussian curve). Figure 1.6 shows graphs of the density of a normal distribution for different values of the mathematical expectation (parameter a) and the standard deviation (parameter σ).

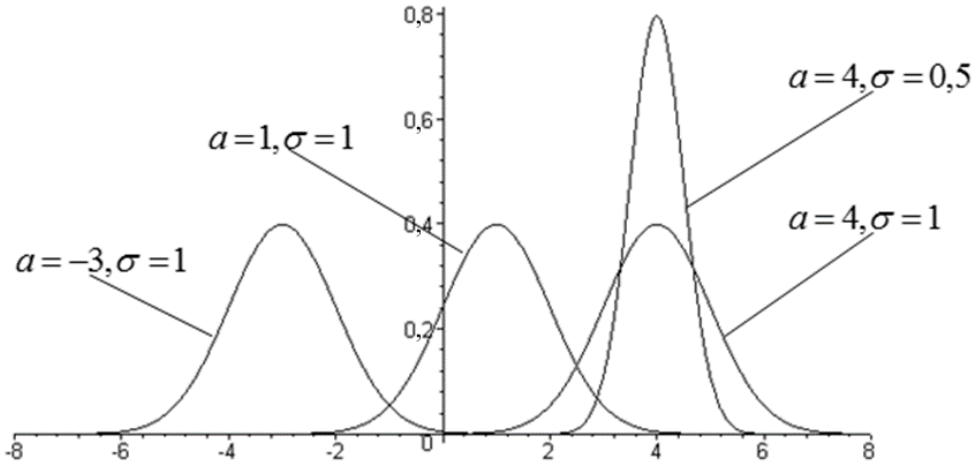


Figure 1.6 – Gaussian curve

For a normally distributed random variable X, the probability that X will take a value belonging to the interval (α, β), is equal to

$$P(\alpha < X < \beta) = \frac{1}{\sigma\sqrt{2\pi}} \int_{\alpha}^{\beta} e^{-(x-a)^2/2\sigma^2} dx = \Phi\left(\frac{\beta - a}{\sigma}\right) - \Phi\left(\frac{\alpha - a}{\sigma}\right), \qquad (1.17)$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{0}^{x} e^{-\frac{t^2}{2}} dt$ – Laplace function, the value of which is given in Appendix A.

Exponential is called the probability distribution of a continuous random variable *X*, which is described by the distribution density:

$$f(x) = \begin{cases} 0, & x < 0, \\ \lambda e^{-\lambda x}, & x \geq 0, \end{cases} \qquad (1.18)$$

where λ – constant positive value.

Exponential distribution function

$$F(x) = \begin{cases} 0, & x < 0, \\ 1 - e^{-\lambda x}, & x \geq 0. \end{cases} \qquad (1.19)$$

The graphs of the distribution density and distribution function for the exponential law are shown in Figure 1.7.

Numerical characteristics: mathematical expectation – $m_x = 1/\lambda$, dispersion – $D_x = 1/\lambda^2$, standard deviation – $1/\lambda$, fashion – $M_{0x} = 0$, median – $M_{ex} = \ln 2/\lambda$, the asymmetry coefficient is $As_x = 2$, the excess coefficient is $Ex = 6$, coefficient of variation – $v_x = 1$.

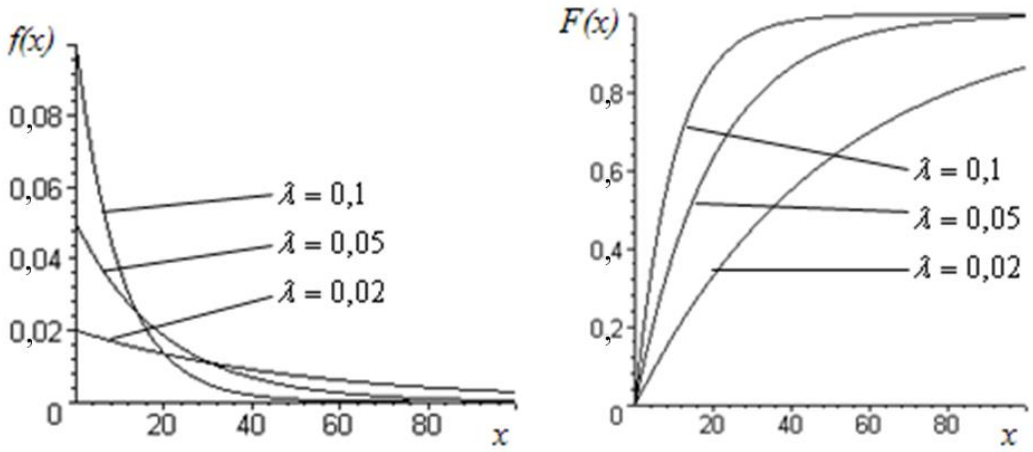


Figure 1.7 – Exponential distribution law

For an exponentially distributed random variable X, the probability that X will take a value belonging to the interval (α, β), equals

$$P(\alpha < X < \beta) = \int_{\alpha}^{\beta} \lambda e^{-\lambda x} dx = e^{-\lambda\alpha} - e^{-\lambda\beta}. \quad (1.20)$$

The main property of the exponential distribution is the absence of aftereffect (or memory, characterized by the equality $P(X > t+s \,|\, X > s) = P(X > t)$). This means that the probability of an event occurring in the future does not depend on how much time has passed. The exponential distribution is fundamental in queueing theory, and most models studied are based on this distribution.

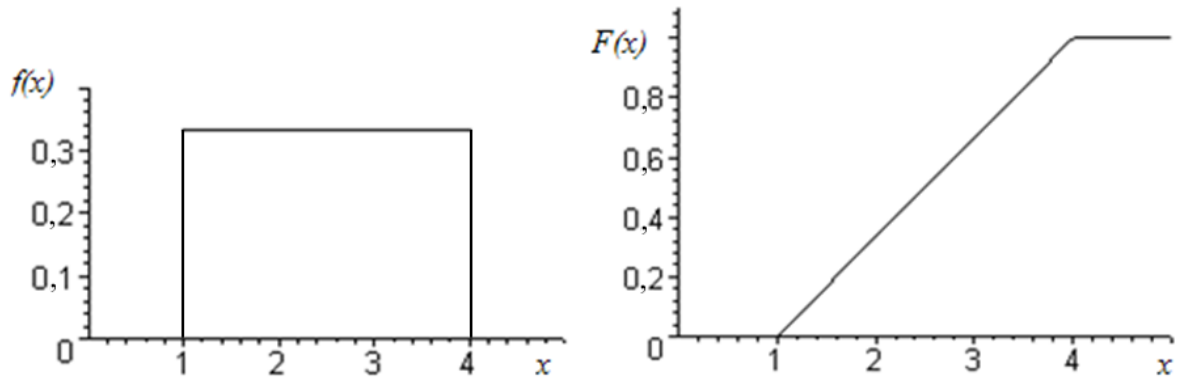
The probability distribution of a continuous random variable X is called uniform if on the interval (a,b), to which all possible values of a random variable belong, the distribution density maintains a constant value, namely:

$$f(x) = \begin{cases} 0, & x < a, \\ \dfrac{1}{b-a}, & a \le x \le b, \\ 0, & x > b. \end{cases} \quad (1.21)$$

Uniform distribution function

$$F(x) = \begin{cases} 0, & x < a, \\ \dfrac{x-a}{b-a}, & a \le x \le b, \\ 1, & x > b. \end{cases} \quad (1.22)$$

The graphs of the distribution density and distribution function for the uniform law are shown in Figure 1.8.



a *b*

Figure 1.8 – Uniform distribution law:
a – graph of distribution density, *b* – graph of distribution function

Numerical characteristics: mathematical expectation – $m_x = (b+a)/2$, dispersion – $D_x = (b-a)^2/12$, standard deviation – $(b-a)/\sqrt{12}$, median – $M_{ex} = (b+a)/2$, the asymmetry coefficient is $As_x = 0$, the excess coefficient is $Ex = -1{,}2$, coefficient of variation $-v_x = (b-a)/\sqrt{3}(b+a)$.

To calculate the probability of a random variable *X*, uniformly distributed, falling into an interval (α, β) you can use formula (1.12) or (1.13).

1.1.4 Mathematical statistics

The mathematical laws of probability theory are mathematical expressions (models) of real-world patterns that exist in mass random phenomena. Indeed, the frequency of landing on heads in repeated coin tosses tends in probability to the corresponding probability:

$$p^* = \frac{m_{\text{nominal}}}{n_{\text{throwing}}} \xrightarrow{\text{by probability at } n\to\infty} p = \frac{m_{\text{favorable}}}{n_{\text{total}}} = \frac{1}{2}.$$

In probability theory, we've used various distribution laws and their characteristics. A logical question arises: where do these laws come from? The answer is simple: all distribution laws are derived from experience and practice. Every study in the field of random events is based either on experimentation or on the observation of random events, which results in the formation of so-called statistical data. The development of methods for recording, describing, and analysing statistical experimental data is the subject of a specialized science: mathematical statistics.

The main task of mathematical statistics is analysis of statistical data depending on the research objectives. This includes:

a) estimation of an unknown probability of an event; estimation of an unknown distribution function; estimation of the parameters of a distribution whose type is known; testing of statistical hypotheses about the type of an unknown distribution or about the values of the parameters of a distribution whose type is known.

Let's say we need to study a set of homogeneous objects with respect to some qualitative or quantitative characteristic that characterizes them. For example, if we have a batch of products, the qualitative characteristic might be the grade of a unit of production or its colour, and the quantitative characteristic might be the controlled weight of a unit of production. The entire set of objects to be studied is called the general population.

Typically, a limited number of objects are selected from a population and subjected to study. A sample population, or simply a sample, is a set of randomly selected objects from a population. The size of a population (either a sample or a general population) is the number of objects in that population. For example, if 100 parts are selected for study out of 1,000, then the general population size is $V = 1000$, and the sample size is $N = 100$.

To construct models based on experimental data in queuing theory, the primary focus is on the type of distribution law of a random variable and its parameters. For this purpose, during the system survey stage, an interval distribution and a so-called relative frequency density histogram are constructed. For this, the interval containing all observed values of the attribute, H, is divided into several partial intervals m of length $h = H / m$, and the value of the distribution function is found for each partial interval n_i – the sum of the frequencies of the variants that fall into the i-th interval and w_i – relative frequencies equal to the ratio $w_i = n_i / N$.·The results of such processing can be presented in the form of an interval distribution table:

Table 1.2 – Interval distribution of feature values

x_i , x_{i+1}	x_1 , x_2	x_2 , x_3	….	x_{m-1} , x_m
n_i	n_1	n_2	….	n_m
w_i	w_1	w_2	….	w_m

Note that $w_1 + w_2 + \ldots + w_m = 1$.

*A relative frequency density histogram is a stepped figure consisting of rectangles***,** whose bases are partial intervals of length h, and whose heights are equal to the densities of relative frequencies - ρ_i, calculated using the formula:

$$\rho_i = \frac{n_i}{N \cdot h}. \qquad (1.23)$$

If we consider ρ_i as a function of *x*, then this will be the empirical distribution density, and the relative frequency density histogram will be its graph. To construct a relative frequency density histogram, partial intervals are plotted on the *x*-axis, and line segments are drawn above them, parallel to the *x*-axis at a distance ρ_i..The area of the *i*-th partial rectangle is equal to $h \cdot \frac{n_i}{N \cdot h_i} \Big/ h = w_i$ – the relative frequency of the variants falling within the *i*-th interval. Consequently, the

area of the histogram of relative frequencies is equal to the sum of all relative frequencies, i.e. one.

Example A. The time intervals between successive requests entering the system are fixed at ξ_i. Construct a histogram of the relative frequencies of the random time intervals included in the sample.

7.35 8 .97 70 .5 28 .0 5.27 18.3 17.6 3.9 17.5 21.3 59.7 1.73 22.4 63 .1 7.82 34.2
7.39 28.3 86.2 10.2 3.30 7.70 13.9 .701 63.5 2.38 65.1 24.2 .116 38.3 37.3 30.8
74.7 23.4 59.3 20.1 25.1 76.6 10.9 20.1 1.05 25.9 5.16 67.0 33.1 21.1 11.0 43.5
26.9 27.7 31.5 0.30 9.01 20.6 14.3 49.3 6.85 2.54 62.4 15.5 44.7 60.4 65.8 17.5
4.58 119. 6.04 8.74 18.7 1.61 40.0 26.8 35.8 26.6 16.0 38.5 0.389 110. 17.9 1.89
27.9 5.24 6.62 25.6 1.41 47.4 13.1 3.25 31.6 58.1 21.0 39.7 43.9 27.6 24.0 18.0
7.78 21.7 1.93 26.7

Solution. Based on the sample, we find the minimum value of the option - $\xi_{\min} = 0{,}12$ and the maximum value option $\xi_{\max} = 119$. Thus, all options are in the interval [0; 119] of length 119. Let's split it into 10 equal-length ($h = 119/10 = 11{,}9$) partial half-intervals: [0; 11.9), [11.9; 23.8), [23.8; 35.7), [35.7; 47.6), [47.6; 59.5), [59.5; 71.4), [71.4; 83.3), [83.3; 95.2),[95.2; 107.1), [107.1; 119]

We assign each sample value to one of these partial half-intervals and sum the number of variants that fall into the first, second, third, …, and tenth half-intervals.

The result is: $n_1 = 33,\ n_2 = 21,\quad n_3 = 19,\ n_4 = 10,\ n_5 = 3,\quad n_6 = 9,\ n_7 = 2,$ $n_8 = 1,\ n_9 = 0,\ n_{10} = 2$.

Sample size $N = 100$. Using formula (1.23), we calculate the relative frequency densities ρ_i for each half-interval: .

$\rho_1 = 0{,}02773;\ \rho_2 = 0{,}01765; \rho_3 = 0{,}01597;\ \rho_4 = 0{,}0084;\ \rho_5 = 0{,}00252;\ \rho_6 = 0{,}00756;$
$\rho_7 = 0{,}00168;\ \rho_8 = 0{,}0084;\ \rho_9 = 0{,}00;\ \rho_{10} = 0{,}00168.$

On the abscissa axis we plot partial intervals of length 11.9, and above them we draw segments parallel to the abscissa axis at a distance of the calculated ρ_i. The desired histogram of relative frequencies is shown in Figure 1.9.

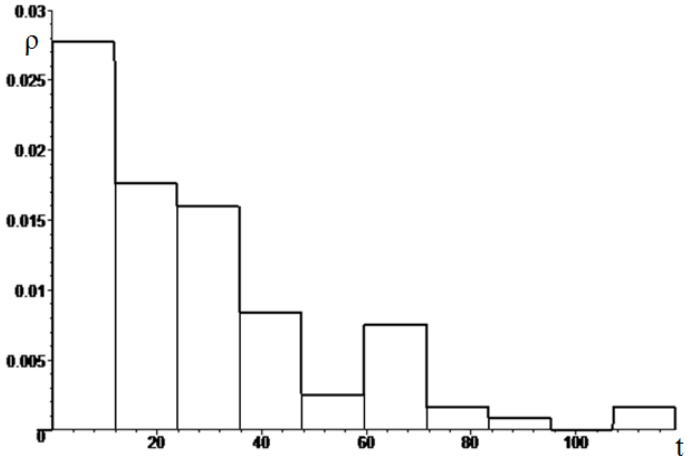


Figure 1.9 – Histogram of relative frequencies

Based on the appearance of the relative frequency histogram, one can hypothesize about the distribution law that governs the sample data. For example, Figure 1.9 suggests that the random variable – the time intervals between requests entering the system – observes an exponential law (see figure 1.7).

Constructing statistical distributions for a sample and plotting them graphically is only the first step in analyzing a statistical data series. The next step is finding numerical characteristics that compactly express the most important features of the sample's statistical distribution. It should be noted that any numerical characteristic calculated based on a limited sample contains an element of randomness and will vary from sample to sample. An approximate value of any parameter in a statistical sample is called an estimate of the parameter. Using an approximate parameter value instead of an exact one introduces errors. It is desirable to select estimates so that errors are minimized.

A number of requirements are imposed on the numerical characteristics of a sample: unbiasedness, efficiency, and consistency [8]. In queuing theory, the main numerical characteristics of a sample are the estimate for the mathematical expectation and the estimate for the variance.

An unbiased and consistent estimate for the mathematical expectation is the sample mean, calculated using the formulas:

$$m^* = \bar{x} = \frac{1}{N}\left(x_1 + x_2 + ... + x_N\right) = \frac{1}{N}\sum_{i=1}^{N} x_1, \tag{1.24}$$

where N is the sample size, and all options are different and

$$m^* = \bar{x} = \frac{1}{N}\sum_{i=1}^{m} n_i x_i, \tag{1.25}$$

if there are repeating variants, the number of which is m.

An unbiased and consistent estimate for the variance is calculated using the formulas:

$$D^* = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - m^*\right)^2 \text{ or } D^* = \frac{1}{N-1}\sum_{i=1}^{m} n_i\left(x_i - m^*\right)^2. \tag{1.26}$$

To estimate the standard deviation, the value is used σ^*, equal to the square root of the sample variance: $\sigma^* = \sqrt{D^*}$.

To compare several statistical distributions in terms of their dimensionality relative to the sample mean, the coefficient of variation is used, which is equal to the ratio of the estimates for the standard deviation and the mathematical expectation:

$$v^* = \frac{\sigma^*}{m^*}. \quad (1.27)$$

Based on the analysis of the distribution histogram, a theoretical distribution can be selected that describes the population and depends on one or more parameters. In some cases, the distribution type is known a priori. For example, the time between device failures is typically subject to an exponential distribution, while the deviations of measurement results from the true value are normally distributed. It is necessary to estimate the distribution parameters. For this purpose, the method of moments proposed by K. Pearson and the maximum likelihood method developed by R. Fisher are used.

The method of moments consists of equating the initial or central moments (mathematical expectation, variance, etc.) of a theoretical distribution, which are functions of the distribution parameters, to the corresponding moments calculated from a sample, and solving one or more equations with respect to the distribution parameters.

Example B. From the analysis of the histogram constructed based on the sample of previous example and shown in fig. 1.9, it was concluded that the random variable Ξ (xi) – the time intervals between the moments of arrival of requests in the system – obeys an exponential law with a distribution density $f(t) = \begin{cases} 0, & t < 0, \\ \lambda e^{-\lambda t}, & t \geq 0, \end{cases}$. It is necessary to find a point estimate for the parameter λ of the distribution using the method of moments.

Solution. From paragraph 1.3 it is known that the parameter λ of the exponential distribution is defined as $\lambda = \frac{1}{m}$. Using the data and formula 1.21, we find an estimate for the mathematical expectation of the sample – $m^* = 27{,}054$, respectively $\lambda = \lambda^* = \frac{1}{27{,}054} = 0{,}03696$. Figure 1.10 shows a graph of the density distribution of a random variable – the time intervals between the moments of arrival of requests with an estimated parameter into the system, superimposed on a histogram of relative frequencies (Figure 1.10).

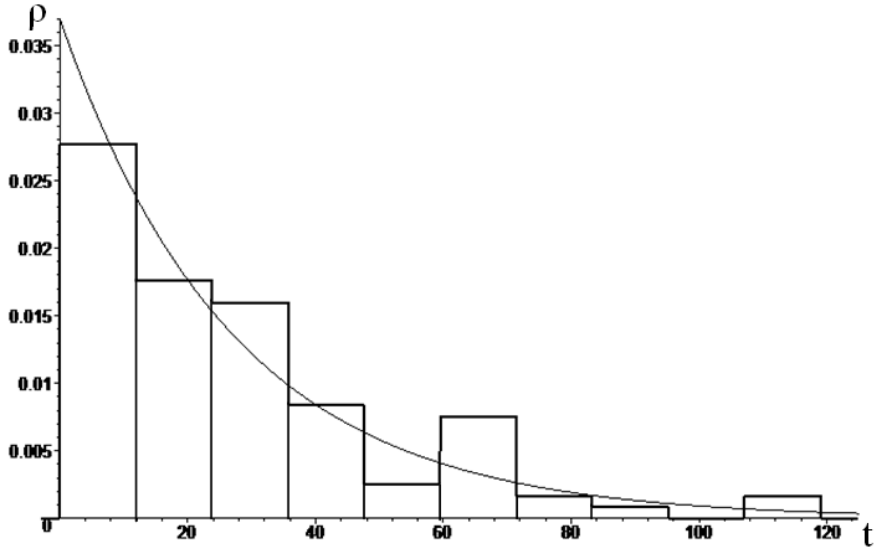


Figure 1.10 – Summary graph of experimental and theoretical divisions

A good match between the graphs of theoretical and experimental distribution densities is evident.

Above, based on an analysis of the relative frequency histogram constructed from the sample data, it was hypothesized that the distribution of the random variable Ξ – the time intervals between the arrival of requests into the system – follows an exponential law. Such hypotheses are called statistical hypotheses in mathematical statistics.

Statistical called a hypothesis about the type of an unknown distribution, or about the parameters of known distributions.

Along with the proposed hypothesis, a contradictory hypothesis is also considered. If the proposed hypothesis is rejected, then the contradictory hypothesis holds. For this reason, it is useful to distinguish between these hypotheses. The proposed hypothesis is called the null (main) hypothesis H_0.

Competing (alternative) call the hypothesis H_1, which contradicts the zero one.

For example, if the null hypothesis is that the mean *m* of the exponential distribution is 10, then a competing hypothesis might, in particular, be that $m \neq 10$.

A hypothesis may be correct or incorrect, so it needs to be tested. Since the testing is performed using statistical methods, it is called statistical. As a result of statistical testing of a hypothesis, an incorrect decision may be made in two cases, i.e., two types of errors may be made.

Type I error is that the correct hypothesis will be rejected.

Type II error is that the wrong hypothesis will be accepted.

It should be emphasized that the consequences of these errors can be quite different. For example, if the correct decision to "continue construction of a residential building" is rejected, this Type I error will result in material damage; however, if the incorrect decision to "continue construction" is made despite the risk of collapse, this Type II error can result in loss of life. There are examples where a Type I error has more serious consequences than a Type II error.

The correct decision can also be made in two cases:

1) the hypothesis is accepted, and in fact it is correct;

2) the hypothesis is rejected, and in fact it is incorrect.

If the main hypothesis H_0 contains an assumption about the parameters of a known distribution, then the statistical criterion for testing it is called parametric. If the hypothesis concerns an unknown distribution, then the corresponding criterion is called the goodness-of-fit criterion.

Definition. The probability of making a type I error is usually denoted by α; it is called the significance level. Most often, the significance level is taken to be 0.1; 0.05; 0.01, or 0.001. If, for example, a significance level of 0.05 is adopted, this means that in five cases out of a hundred there is a risk of making a type I error (rejecting the correct hypothesis).

To test the null hypothesis, a specially selected random variable is used, the exact or approximate distribution of which is known. This variable is denoted by U or Z if it is normally distributed, F or v^2 – according to the Fisher-Snedecor law, T – according to the Student's law, χ^2 – according to the chi-square law, etc. For the sake of generality, we will denote this value by K.

Statistical criterion called the random variable K, which serves to test the null hypothesis.

To test a hypothesis based on sample data, the partial values of the quantities included in the criterion are calculated and thus the partial (observed) value of the criterion is obtained.

Observed value K_{obs} is the value of a statistical criterion calculated from samples. For example, if corrected sample variances are found for two samples $s_1^2 = 20$ and $s_2^2 = 5$, then the observed value of the F criterion is equal to the ratio of two variances, $F_{набл} = s_1^2 / s_2^2 = 20/5 = 4$.

After selecting a particular criterion, the set of all its possible values is divided into two disjoint subsets: one of them contains the criterion values for which the null hypothesis is rejected, and the other contains those for which it is accepted.

Critical region is called the set of criterion values for which the null hypothesis is rejected.

The area of acceptance of the hypothesis (the range of acceptable values) is the set of criterion values for which the hypothesis is accepted.

In queuing theory, a hypothesis about the distribution of a general population is subject to statistical testing. Statistical criteria for testing this hypothesis are specially called goodness-of-fit criteria. One such criterion is the χ^2 (chi-square), proposed by the American statistician K. Pearson and named after him.

In this criterion, the measure of the divergence of the distribution put forward by the hypothesis (theoretical distribution) from the real one is the value χ^2_{obs}:

$$\chi^2_{obs} = \sum_{i=1}^{m} \frac{(n_i - Np_i)^2}{Np_i} = \sum_{i=1}^{m} \frac{(n_i - n_i^{'})^2}{n_i^{'}}, \qquad (1.28)$$

Where $n_i^{'} = Np_i$ -theoretical frequencies of values of a random variable;

n_i – empirical frequencies of values of a random variable;

m – the number of different values or groups of values of a random variable.

Size χ^2_{obs} – a random variable with a chi-square distribution with the number of degrees of freedom k:

$$k = m - s - 1, \qquad (1.29)$$

where m is the number of groups of the empirical distribution;

s – the number of parameters of the theoretical distribution that are estimated from the empirical distribution.

Based on the selected significance level α and the number of degrees of freedom $k = m - s - 1$ in Appendix 2 of the critical points of the distribution χ^2, a value χ^2_{cr} is found such that $P\{\chi^2_{obs} > \chi^2_{cr}\} = \alpha$. If it turns out that the value of the quantity calculated from the sample χ^2_{obs} got into the range of acceptable values, i.e. $\chi^2_{obs} < \chi^2_{cr}$, then the sample is considered to confirm the hypothesis that the general population has a distribution function $F(x)$. If the numerical value χ^2_{obs} will be in the critical region, i.e. $\chi^2_{obs} > \chi^2_{cr}$, then the formulated hypothesis is rejected.

It should be borne in mind that the adoption of a chi-square distribution for the quantity χ^2_{obs} is considered acceptable if for all values $i = 1, 2, ..., m \quad N \cdot p_i \geq 5$.

Example. At a significance level of 0.05, test the hypothesis that, based on the sample data from example *A*, the distribution of the random variable Ξ – the time intervals between the moments of requests arriving in the system – obeys an exponential law.

Solution. In example B, the exponential law parameter was obtained $\lambda = 0{,}03696$. We will find the theoretical frequencies using the formula for calculating the probability of a random variable distributed according to an exponential law falling into the interval (a_{i-1}, a_i) – (1.20):

$$n'_i = N \cdot P(a_{i-1} < \Xi < a_i) = N(\exp(-\lambda a_{i-1}) - \exp(-\lambda a_i)) \cdot$$

We have:

$n'_1 = N \cdot P(0 < \Xi < 11{,}9) = 100 \cdot (\exp(-0{,}03696 \cdot 0) - \exp(-0{,}03696 \cdot 11{,}9)) \approx 35{,}6;$

$n'_2 = N \cdot P(11{,}9 < \Xi < 23{,}8) = 100 \cdot (\exp(-0{,}03696 \cdot 11{,}9) - \exp(-0{,}03696 \cdot 23{,}8)) \approx$

$\approx 22{,}9;$

$n'_3 = N \cdot P(23{,}8 < \Xi < 35{,}7) \approx 14{,}7$; $n'_4 = N \cdot P(35{,}7 < \Xi < 47{,}6) \approx 9{,}5$;

$n'_5 = N \cdot P(47{,}6 < \Xi < 59{,}5) \approx 6{,}1$; $n'_6 = N \cdot P(59{,}5 < \Xi < 71{,}4) \approx 3{,}9$;

$n'_7 = N \cdot P(71{,}4 < \Xi < 83{,}3) \approx 2{,}5$; $n'_8 = N \cdot P(83{,}3 < \Xi < 95{,}2) \approx 1{,}6$.

$n'_9 = N \cdot P(95{,}2 < \Xi < 107{,}1) \approx 1{,}1$; $n'_{10} = N \cdot P(107{,}1 < \Xi < 119) \approx 0{,}7$.

Let's compare the empirical and theoretical frequencies using Pearson's test:

a) calculate the observed value of the Pearson criterion

$$\chi^2_{набл} = \sum_{i=1}^{10} \frac{(n_i - N \cdot p_i)^2}{N \cdot p_i} = \sum_{i=1}^{10} \frac{(n_i - n'_i)^2}{n'_i} = \frac{(33-35{,}6)^2}{33{,}6} + \frac{(21-22{,}9)^2}{22{,}9} + \frac{(19-14{,}7)^2}{14{,}7} +$$

$$+ \frac{(10-9{,}5)^2}{9{,}5} + \frac{(3-6{,}1)^2}{6{,}1} + \frac{(9-3{,}9)^2}{3{,}9} + \frac{(2-2{,}5)^2}{2{,}5} + \frac{(1-1{,}6)^2}{1{,}6} +$$

$$+ \frac{(0-1{,}1)^2}{1{,}1} + \frac{(2-0{,}7)^2}{0{,}7} \approx 13{,}72.$$

b) using formula (1.26) we determine the number of degrees of freedom: $k = 10 - 1 - 1 = 8$;

c) according to the table of Appendix 2 of critical distribution points χ^2, for the significance level $\alpha = 0{,}05$ and the number of degrees of freedom $k = 8$ let's find the critical point of the right-hand critical region: $\chi^2_{kp}(0{,}05; 8) = 15{,}5$.

Because $\chi^2_{obs} < \chi^2_{cr}$, then we accept the hypothesis of the exponential distribution of the random variable Ξ – the time intervals between the moments of arrival of service requests in the system.

1.2 Principles of the theory of stochastic processes

1.2.1 Basic concepts of stochastic processes

A single-line queuing system (then the queue system) functions as follows. A service request that arrives in the system when the service device is free is accepted for service. The remaining requests that arrive in the system when the service device is busy are placed in a queue. After the next request is serviced, the service device selects a request from the queue and starts servicing it. If there are no requests, the service device is idle, waiting for the next request to arrive. That is, in a queuing system, both when requests arrive and after they are finished, the number of requests in the queue and in the system (the number of requests in the queue and in service) is continuously changing.

As is known, a process is understood as a sequential change in the states of an object over time [11]. Thus, various processes occur in queuing systems. Figure 1.11 shows the time diagrams of the basic processes in a single-line queuing system.

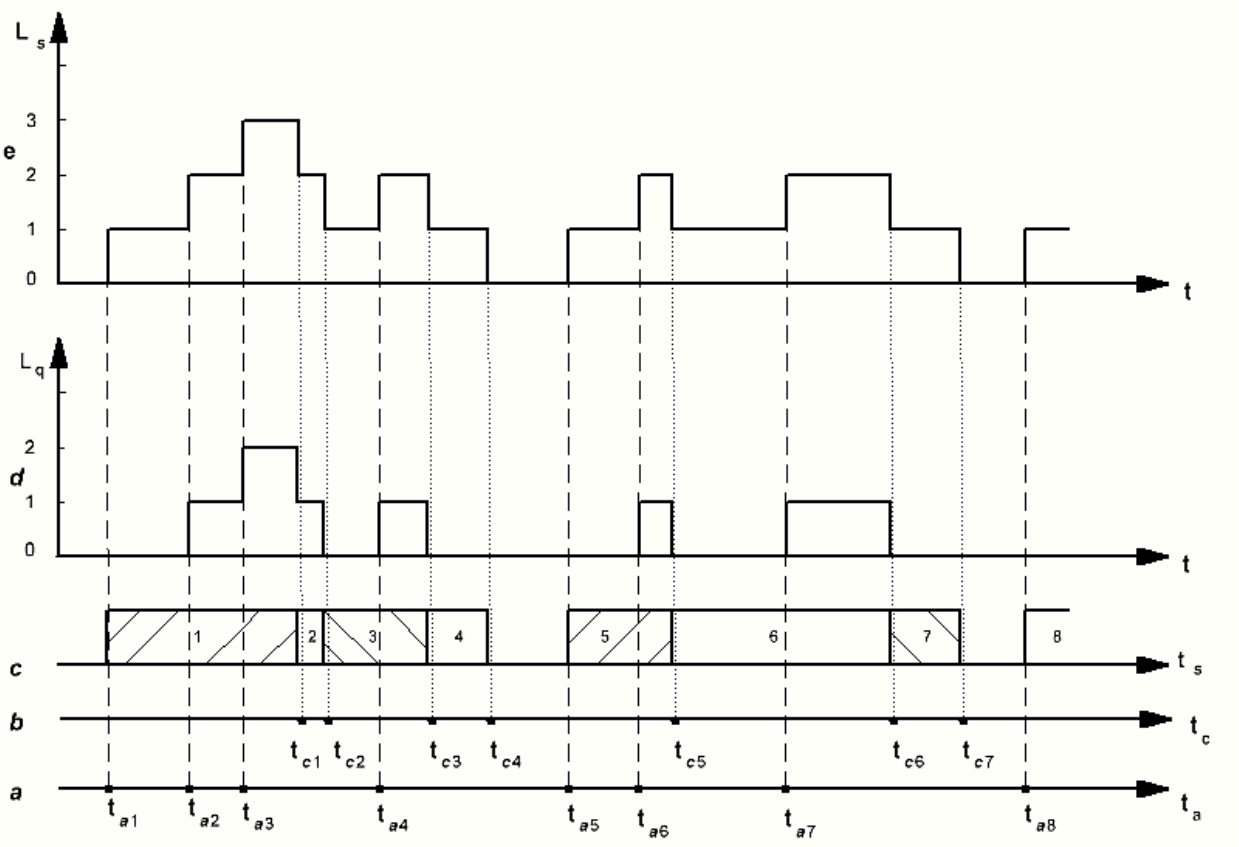


Figure 1.11 – Processes in a single-line queuing system

So, Figure 1.11,a shows the process of requests arriving in the system. The moments of arrival are marked with points t_{ai}. For each incoming request, the number of requests in the system increases, and if the serving device is busy, the

length of the queue also increases. Figure 1.11,b shows the moments of the service requests completing (marked with points t_{ci}). Accordingly, the length of the queue and the number of requests in the system decrease. Figure 1.11,c shows the process of sequential service of requests on the serving device. The numbers indicate the numbers of requests that are served. Figure 1.11,d shows a time diagram of the change in the number of requests in the queue, and figure 1.11,e shows a time diagram of the change in the number of requests in the system. From figure 1.11 it follows that the states of the queuing system change at random moments of time, accordingly, all processes are probabilistic and, in order to study them, it is necessary to use the theory of stochastic processes.

Definition. In mathematics, a random process is a function of two arguments, the values of which are random variables [7]:

$$\xi(t) = \varphi(n,t),\ n = 0,1,...,N,\ t \in T, \tag{1.30}$$

where $N+1$– the number of elementary events (it can be infinite); t is a parameter; T is a set of values of the parameter T.

For each value of the parameter t, the function $\varphi(n,t)$ is a function of n only and is therefore a random variable. For each fixed value of the argument n (i.e., for each elementary event) $\varphi(n,t)$ depends only on t and is, therefore, simply a function of one real argument. Each such function is an implementation of a random process. Figure 1.11,d shows the implementation of a random process of changing the number of requirements in the system, where at each time the number of requirements in the system is a random variable. As the parameter t in relation to the queuing system, time is used.

Depending on the type of the set T of values of the argument t, random processes can be divided into four types:

– processes with discrete states and discrete time;

– processes with discrete states and continuous time;

– processes with continuous states and discrete time;

– processes with continuous states and continuous time.

Example. Let us have several implementations of random processes $\zeta(t) = \varphi(n,t): x_1(t),\ x_2(t),...,x_n(t)$, presented in Figure 1.12.

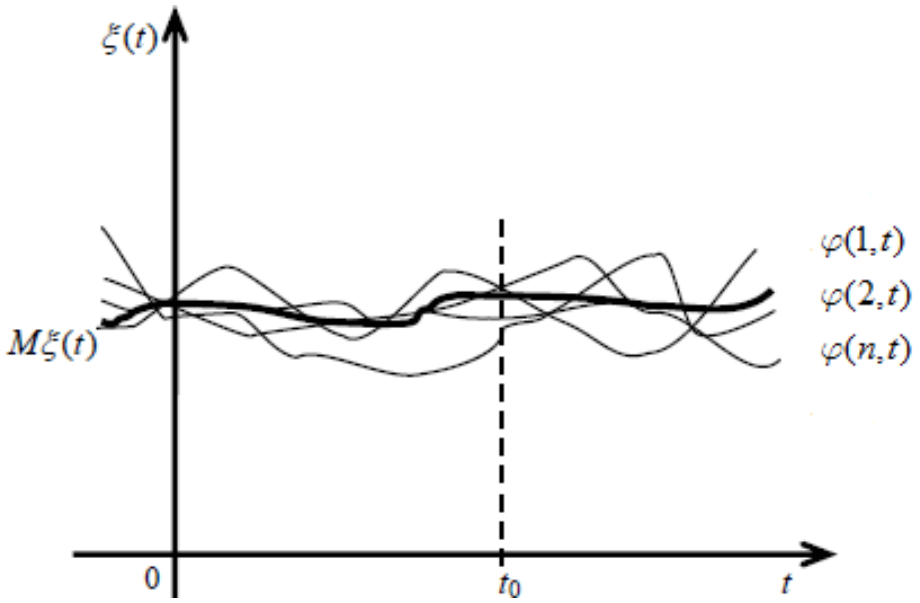


Figure 1.12 – Implementation of a random process $\zeta(t) = \varphi(n,t)$

The bold line indicates a function around which different realizations are possible. This function is called the mathematical expectation of a random process. To obtain the function, a mathematical description of the random process is required, and it can be different for each type of random process.

From the analysis of probabilistic processes of queuing systems (Fig. 1.11) it follows that they are processes with discrete states and continuous time. The state of the system can be considered the number of requests in the system. Depending on the type of queuing system, it can be finite (for systems with a limited queue) or infinite. From Figure. 1.12 it is clear that for a specific value of the parameter *t* the state of the system changes from implementation to implementation, and therefore only its probability or the corresponding moments (mathematical expectation, variance) can be estimated.

Once a request enters the system, it spends some time in the queue, in service, and in the system as a whole. These characteristics can be studied using the theory of stochastic processes with discrete states and continuous time.

The most studied are the following types of random processes: processes with independent increments, processes with uncorrelated values, normal random processes, Markov processes. In the theory of queuing, due to the specificity of the nature of the flow of requests and service time, most problems are solved with the involvement of the theory of Markov processes. Its advantages are a perfect mathematical apparatus, the ability to study both non-stationary and stationary processes, and a visual interpretation of the research results.

A random process is called a Markov process (or a process without aftereffects) if, for each time instant *t*, the probability of any state of the system in the future is determined only by its current state and does not depend on how the system acquired this state. The time parameter *t* can be considered both as discrete and as continuous, which results in different research methods.

1.2.2 Markov random processes with discrete time

Markov random processes, or Markov chains, are named after our compatriot who first studied them. Let the system under study be in states $\varphi(n,t), n = 0, 1, 2,..., N; t = 0, 1, 2, 3,....$ Possible transitions from state $\varphi(i,t)$ in the state $\varphi(j,t), 0 \le i, j \le N$, they occur at discrete moments in time $t = 0, 1, 2, 3,....$ Let $p_n(t)$ denotes the probability of the state $\varphi(n,t)$ at time *t*. Then the vector $[P(t)] = [p_0(t), p_1(t),..., p_N(t)]$ describes the probabilities of the system being in a particular state at a discrete time *t*. Let us denote by $p_{ij}(t), 0 \le i, j \le N,$ conditional probability that the system, being at time t in the state $\varphi(i,t)$, will enter the state $\varphi(j,t+1)$ at time $t+1$ (so-called transition probabilities), then

$$p_j(t+1) = \sum_{i=0}^{N} p_i(t) \cdot p_{ij}(t), j = 0,1,2,...,N . \qquad (1.31)$$

Equation (1.31) and the initial conditions completely describe the Markov chain. If the transition probabilities are time-independent ($p_{ij}(t) = p_{ij}$), then such a chain is called homogeneous (otherwise inhomogeneous). Mostly homogeneous Markov chains are used. For a homogeneous Markov chain, equation (1.31) can be written as follows:

$$p_j(t+1) = \sum_{i=0}^{N} p_i(t) \cdot p_{ij}, j = 0,1,2,...,N . \qquad (1.32)$$

Matrix

$$[\Pi] = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdots & p_{0N} \\ p_{10} & p_{11} & p_{12} & \cdots & p_{1N} \\ p_{20} & p_{21} & p_{22} & \cdots & p_{2N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{N0} & p_{N1} & p_{N2} & \cdots & p_{NN} \end{bmatrix} \qquad (1.33)$$

is called a one-step transition probability matrix or a stochastic matrix. Its row probabilities have the following property:

$$\sum_{j=0}^{N} p_{ij} = 1. \qquad (1.34)$$

Equation (1.32) in matrix form can be written as follows:

$$[P(t+1)] = [P(t)] \cdot [\Pi]. \qquad (1.35)$$

Expression (1.35) allows us to find the probabilities of the system state in one step. If we want to find the probabilities of the system in two steps, we do it like this:

$$[P(t+2)] = ([P(t)] \cdot [\Pi]) \cdot [\Pi] = [P(t)] \cdot [\Pi]^2. \qquad (1.36)$$

By induction we obtain

$$[P(t+k)] = ([P(t)] \cdot [\Pi]) \cdot [\Pi] = [P(t)] \cdot [\Pi]^k, \qquad (1.37)$$

from which the most important characteristic of a Markov chain follows – the transition matrix from state $\varphi(i,t)$ in the state $\varphi(j,t+k)$ in *k* steps

$$[P_{ij}^k] = [\Pi]^k. \qquad (1.38)$$

Example. Let there be three alternative smartphone models: a, b, c. A survey was conducted among consumers regarding their attitude towards these models. The proportion (frequency) of consumers who prefer one or another model was determined: $p_a(0) = 0{,}6;\ p_b(0) = 0{,}3;\ p_c(0) = 0{,}1$. Over the course of a month, consumers are surveyed about whether they continue to prefer their chosen models or whether they would like to change the type of model according to their preferences. The frequencies obtained can be considered conditional probabilities of switching:

$$p_{aa} = 0{,}2;\ p_{ab} = 0{,}5;\ p_{ac} = 0{,}3;\ p_{ba} = 0{,}4;\ p_{bb} = 0{,}1;\ p_{bc} = 0{,}5;$$
$$p_{ca} = 0{,}5;\ p_{cb} = 0{,}2;\ p_{cc} = 0{,}3.$$

A new survey was conducted a month later, etc. Assuming that consumer preferences do not change over time, we obtained a homogeneous Markov chain with a transition probability matrix:

$$[\Pi] = \begin{bmatrix} 0{,}2 & 0{,}5 & 0{,}3 \\ 0{,}4 & 0{,}1 & 0{,}5 \\ 0{,}5 & 0{,}2 & 0{,}3 \end{bmatrix}.$$

For clarity, it is convenient to present the studied process in the form of a graph (Figure 1.13).

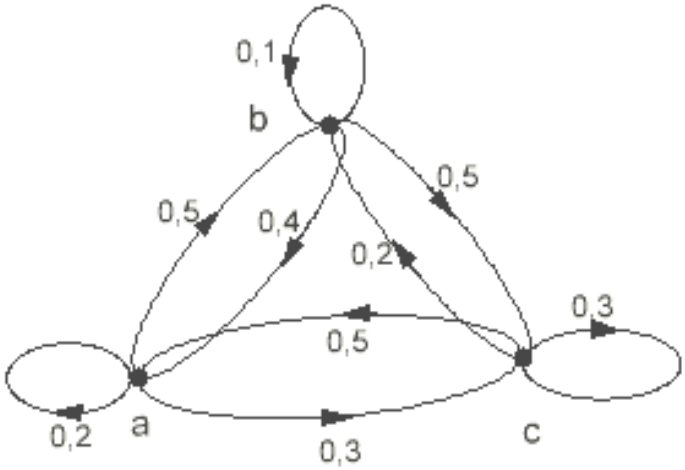


Figure 1.13 – System state graph

Using formula (1.35), we determine the consumer benefits per month:

$$[P(t+1)] = [P(t)] \cdot [\Pi] = [0{,}6; 0{,}3; 0{,}1] \cdot \begin{bmatrix} 0{,}2 & 0{,}5 & 0{,}3 \\ 0{,}4 & 0{,}1 & 0{,}5 \\ 0{,}5 & 0{,}2 & 0{,}3 \end{bmatrix} = [0{,}29; 0{,}35; 0{,}36].$$

So, 29% of all consumers voted for model a, and 29% for model b.

b – 35%, for model c – 36%. In two months we will get:

$$[P(t+2)] = [P(t+1)] \cdot [\Pi] = [0{,}29; 0{,}35; 0{,}36] \cdot \begin{bmatrix} 0{,}2 & 0{,}5 & 0{,}3 \\ 0{,}4 & 0{,}1 & 0{,}5 \\ 0{,}5 & 0{,}2 & 0{,}3 \end{bmatrix} = [0{,}378; 0{,}252; 0{,}37].$$

In three months we will receive:

$$[P(t+3)] = [P(t)] \cdot [\Pi]^3 = [0{,}6; 0{,}3; 0{,}1] \cdot \begin{bmatrix} 0{,}2 & 0{,}5 & 0{,}3 \\ 0{,}4 & 0{,}1 & 0{,}5 \\ 0{,}5 & 0{,}2 & 0{,}3 \end{bmatrix}^3 = [0{,}361; 0{,}288; 0{,}351].$$

In six months –

$$[P(t+6)] = [P(t)] \cdot [\Pi]^6 = [0{,}6; 0{,}3; 0{,}1] \cdot \begin{bmatrix} 0{,}2 & 0{,}5 & 0{,}3 \\ 0{,}4 & 0{,}1 & 0{,}5 \\ 0{,}5 & 0{,}2 & 0{,}3 \end{bmatrix}^6 = [0{,}362; 0{,}281; 0{,}357].$$

We can conclude that probabilities tend to their limits when $k \to \infty$. This property is called ergodic, according to which a system described by a Markov chain eventually transitions into a stable, stationary state, for the existence of which certain conditions are necessary.

Let us introduce some concepts. If the stochastic matrix $[\Pi]$ looks like $\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$, where *A* and *B* are square submatrices, then it is called decomposable, otherwise it is indecomposable. A system in state *A* will never go to state *B* and vice versa. Example of a decomposable matrix:

$$\begin{bmatrix} p_{11} & p_{12} & 0 & 0 \\ p_{21} & p_{22} & 0 & 0 \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & p_{43} & p_{44} \end{bmatrix}.$$

If the stochastic matrix $[\Pi]$ looks like $\begin{bmatrix} 0 & C \\ D & 0 \end{bmatrix}$, where *C* and *D* are square submatrices, then it is called periodic, otherwise it is non-periodic. The system will periodically transition from state *C* to state *D*, and vice versa.

If a Markov chain is indecomposable or non-periodic, then for the system it describes there is a stationary state [12] and

$$\lim_{k\to\infty}\left[P(t+k)\right]=\lim_{k\to\infty}\left[p_0(t+k),\, p_1(t+k),...,\, p_N(t+k)\right]=\left[\pi_0,\pi_1,\pi_2,...,\pi_N\right]. \quad (1.39)$$

Let us denote the stationary probabilities $\pi_1,\pi_2,...,\pi_N$ or (in matrix form) – $\left[p(\infty)\right]=\left[\pi_1,\pi_2,...,\pi_N\right]$. To determine the stationary probabilities, we use expression (1.35), which can be written as follows:

$$\left[p(\infty)\right]=\left[p(\infty)\right]\cdot\left[\Pi\right], \quad (1.40)$$

it needs to be solved relatively $\left[p(\infty)\right]$. Let us write equation (1.40) as follows:

$$\left[p(\infty)\right]=\left[p(\infty)\right]\cdot\left[\Pi\right]\Rightarrow 0=\left[p(\infty)\right]\cdot\left[\Pi\right]-\left[p(\infty)\right]=0\Rightarrow\left[p(\infty)\right]\left(\left[\Pi\right]-I\right)=0,$$

where I is the identity matrix. Expanding the matrices, we obtain:

$$\left[\pi_0,\pi_1,...,\pi_N\right]\cdot\begin{bmatrix} p_{00}-1 & p_{01} & ... & p_{0N} \\ p_{10} & p_{11}-1 & ... & p_{1N} \\ ... & ... & ... & ... \\ p_{N0} & p_{N1} & ... & p_{NN}-1 \end{bmatrix}=0. \quad (1.41)$$

Finally, we will obtain the following system of equations:

$$
\begin{aligned}
&(p_{00}-1)\pi_0 + p_{10}\pi_1 + p_{20}\pi_2 ... + p_{N0}\pi_N = 0,\\
&p_{01}\pi_0 + \ (p_{11}-1)\pi_1 + p_{21}\pi_2 ... + p_{N1}\pi_N = 0,\\
&p_{02}\pi_0 + \quad p_{12}\pi_1 + (p_{22}-1)\pi_2 ... + p_{N2}\pi_N = 0,\\
&.....\\
&p_{0N}\pi_0 + \ p_{1N}\pi_{12} + p_{2N}\pi_2 ... + (p_{NN}-1)\pi_N = 0.
\end{aligned}
\qquad (1.42)
$$

In the system of equations (1.42), one of the equations must be replaced by the so-called normalization condition:

$$\pi_0 + \pi_1 + \pi_2 + ... + \pi_N = 1. \qquad (1.43)$$

Let us define the stationary probabilities for a Markov chain according to example. The system of equations (1.42) for it will have the following form:

$$
\begin{aligned}
&(0{,}2-1)\pi_a + 0{,}4\pi_b + 0{,}5\pi_c = 0,\\
&0{,}5\pi_a + \ (0{,}1-1)\pi_b + 0{,}2\pi_c . = 0,\\
&0{,}3\pi_a + \ 0{,}5\pi_b + (0{,}3-1)\pi_c = 0.
\end{aligned}
$$

Normalization condition: – $\pi_a + \pi_b + \pi_c = 1$. Replacing, for example, the last equation by the normalization condition, we obtain a system of equations for π_a, π_b, π_c

$$
\begin{aligned}
-0{,}8\pi_a + 0{,}4\pi_b + 0{,}5\pi_c &= 0,\\
0{,}5\pi_a - 0{,}9\pi_b + 0{,}2\pi_c . &= 0,\\
1\cdot\pi_a \ + \ 1\cdot\pi_b \ + 1\pi_c &= 1.
\end{aligned}
$$

Let's solve it using Cramer's rule through determinants:

$$\pi_a = \frac{\begin{vmatrix} 0 & 0{,}4 & 0{,}5 \\ 0 & -0{,}9 & 0{,}2 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} -0{,}8 & 0{,}4 & 0{,}5 \\ 0{,}5 & -0{,}9 & 0{,}2 \\ 1 & 1 & 1 \end{vmatrix}} = \frac{0{,}53}{3{,}12} = 0{,}363;\ \pi_b = \frac{\begin{vmatrix} -0{,}8 & 0 & 0{,}3 \\ 0{,}5 & 0 & 0{,}2 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} -0{,}8 & 0{,}4 & 0{,}5 \\ 0{,}5 & -0{,}9 & 0{,}2 \\ 1 & 1 & 1 \end{vmatrix}} = 0{,}281;\ \pi_c = \frac{\begin{vmatrix} -0{,}8 & 0{,}4 & 0 \\ 0{,}5 & -0{,}9 & 0 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} -0{,}8 & 0{,}4 & 0{,}5 \\ 0{,}5 & -0{,}9 & 0{,}2 \\ 1 & 1 & 1 \end{vmatrix}} = 0{,}356.$$

The probabilities calculated in the sixth step of the example practically coincide with the stationary probabilities when studying the process of changing consumer preferences over time.

1.2.3 Markov random processes with discrete states and continuous time

As can be seen from Figure 1.11, the transition of a system from one state to another does not occur at fixed, but at random moments in time. To describe such processes, Markov processes with discrete states and continuous time are used.

Such processes describe the functioning of systems that acquire a finite number of states during operation $\varphi(n,t), n = 0,1, 2,..., N$ and move from one state to another – $\varphi(k,t), k = 0, 1, 2,..., N$ randomly at an arbitrary time *t*. Therefore, the time the system spends in any state is a continuous random variable.

A continuous-time random process is called a Markov process with discrete states and continuous time if the behaviour of the system after an arbitrary time instant t is determined only by the state of the process at time instant t and does not depend on the history of the process preceding time instant t.

The most important characteristic of a system is the probability that at time t the system will be in the state $\varphi(n,t), n = 0, 1, 2,..., N$ – $p_i(t)$. It is obvious that for any time t the sum of all probabilities is equal to unity. Let us denote by $p_{nk}(t,\Delta t)$ probability of the system transitioning from state $\varphi(n,t), n = 0, 1, 2,..., N$ in the state $\varphi(k,t), k = 0, 1, 2,..., N$ in an infinitesimally small amount of time Δt. We denote by $\lambda_{nk}(t)$ such a limit:

$$\lambda_{nk}(t) = \lim_{\Delta t \to 0} \frac{p_{nk}(t, \Delta t)}{\Delta t}. \qquad (1.44)$$

This characteristic is called the transition intensity or transition probability density and generally depends on *t*. It follows from formula (1.44) that for a small Δt transition probability with accuracy to infinitely small higher orders

$$p_{nk}(t, \Delta t) = \lambda_{nk}(t)\Delta t, \Delta t \to 0. \qquad (1.45)$$

If everyone $\lambda_{nk}(t) = \lambda_{nk}$ do not depend on time, then the Markov process is called homogeneous, otherwise it is inhomogeneous. Note that in the theory of queuing systems, mostly homogeneous Markov processes are used.

Similar to discrete-time Markov processes, graphs are widely used to represent continuous-time processes. Each state is represented as a vertex of the graph; between the vertices are marked arcs that reflect the possibility of the system transitioning from the *n*-th state to the *k*-th in an infinitely small interval Δt; above the arcs are placed the values of the transition intensities λ_{nk}. Such a graph is usually called a graph of states and transitions, based on this graph, Kolmogorov's differential equations can be derived [13], which completely describe the behavior of the system.

Example. Let the system S consist of two nodes – *A* and *B*, each of which may fail during operation. The following system states are possible:

– Ψ_0 – both nodes are working;

– Ψ_1 – node A failed and is recovering, node *B* is working;

– Ψ_2 – node A is working, node *B* has failed and is recovering;

– Ψ_3 – both nodes are restored, and the end time of the restoration of node *A* does not coincide with the end time of the restoration of node *B*.

The intensities of transitions between states are known. The graph of states and transitions will have the form shown in figure 1.14.

Let us derive the equation for the probability of state Ψ_0 – $p_0(t)$. To do this, we consider the probabilities of different transitions from the state Ψ_0 and vice versa in the time interval $(t, t+\Delta t)$.

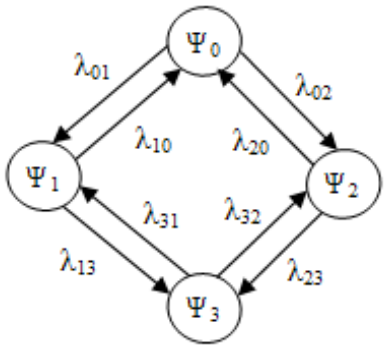


Figure 1.14 – State and transition graph for a system with failures

At time *t* the system was in state Ψ_0; the probability that it transitioned from it to some other state is equal to $p_0 \cdot (\lambda_{01} + \lambda_{02})\Delta t$, and the fact that she remained in it – $1 - p_0(t) \cdot (\lambda_{01} + \lambda_{02}) \cdot \Delta t$. The probability that the system was in state Ψ_1 at time *t* and transitioned to state Ψ_0 over time Δt is equal to $p_1(1) \cdot \lambda_{10} \cdot \Delta t$. The probability that the system at time *t* was in state Ψ_2 and transitioned to state Ψ_0 is equal to $p_2(t) \cdot \lambda_{20} \cdot \Delta t$. Then, using the formulas for adding and multiplying probabilities, we get:

$$p_0(t+\Delta t) = (1-(\lambda_{01}+\lambda_{02})\Delta t)\, p_0(t) + \lambda_{10} \cdot p_1(t) \cdot \Delta t + \lambda_{20} \cdot p_2(t) \cdot \Delta t .$$

Let us open the brackets, move $p_0(t)$ from the right side to the left, divide the entire equation by Δt and find the limit of the left and right parts at $\Delta t \to 0$. We get:

$$\lim_{\Delta t \to 0} \frac{p_0(t+\Delta t) - p_0(t)}{\Delta t} = -(\lambda_{01}+\lambda_{02})\, p_0(t) + \lambda_{10} \cdot p_1(t) + \lambda_{20} \cdot p_2(t) .$$

The left-hand side is the derivative of $p_0(t)$ by t. Finally we get:

$$p_0'(t) = -(\lambda_{01}+\lambda_{02})\, p_0(t) + \lambda_{10} \cdot p_1(t) + \lambda_{20} \cdot p_2(t) .$$

Generalizing the process of constructing a differential equation for the probability of state Ψ_0 – $p_0(t)$, it is possible to compose a mnemonic rule for

composing a differential equation for an arbitrary state Ψ_i using the marked graphs of states and transitions – $p_i(t)$ [12]:

derivative of the probability of any state Ψ_i is equal to the product of the sum of the transition intensities from the state Ψ_i into other states and probabilities of state Ψ_i, taken with a negative sign, plus the sum of the products of the transition intensities from other states to state Ψ_i on the probabilities of the corresponding states.

For other states, the differential equations for the state probabilities will be as follows:

$$p_1'(t) = -(\lambda_{10} + \lambda_{13})p_1(t) + \lambda_{01} \cdot p_0(t) + \lambda_{31} \cdot p_3(t),$$
$$p_2'(t) = -(\lambda_{20} + \lambda_{23})p_2(t) + \lambda_{02} \cdot p_0(t) + \lambda_{32} \cdot p_3(t),$$
$$p_3'(t) = -(\lambda_{31} + \lambda_{32})p_3(t) + \lambda_{13} \cdot p_1(t) + \lambda_{23} \cdot p_2(t).$$

At the initial point in time, the system was operational, therefore

$$p_0(0) = 1, p_1(0) = 0, p_2(0) = 0, p_3(0) = 0.$$

As a result, we obtain a system of differential equations:

$$\begin{aligned} p_1'(t) &= -(\lambda_{10} + \lambda_{13})p_1(t) + \lambda_{01} \cdot p_0(t) + \lambda_{31} \cdot p_3(t), \\ p_2'(t) &= -(\lambda_{20} + \lambda_{23})p_2(t) + \lambda_{02} \cdot p_0(t) + \lambda_{32} \cdot p_3(t), \\ p_3'(t) &= -(\lambda_{31} + \lambda_{32})p_3(t) + \lambda_{13} \cdot p_1(t) + \lambda_{23} \cdot p_2(t), \\ p_3'(t) &= -(\lambda_{31} + \lambda_{32})p_3(t) + \lambda_{13} \cdot p_1(t) + \lambda_{23} \cdot p_2(t) \end{aligned} \quad (1.46)$$

with initial conditions

$$p_0(0) = 1, p_1(0) = 0, p_2(0) = 0, p_3(0) = 0.$$

Using the technology of solving systems of linear differential equations [15, p. 54] for specific values λ_{ij}, one can find an analytical solution to this system. For example, when

$$\lambda_{01} = 0{,}8;\ \lambda_{02} = 1{,}5;\ \lambda_{10} = 1{,}6;\ \lambda_{13} = 2{,}0;\ \lambda_{20} = 1{,}3;\ \lambda_{23} = 2{,}5;\ \lambda_{31} = 3{,}8;\ \lambda_{32} = 4{,}6$$

the solution of system (1.46) will look like this:

$$p_0(t) = 0{,}8544\sin(0{,}2541t + 0{,}8943)e^{-3{,}4892t} + 0{,}01e^{-11{,}12t} + 0{,}32788,$$
$$p_1(t) = -0{,}9290\sin(0{,}2541t + 0{,}2094)e^{-3{,}48t} - 0{,}03e^{-11{,}12t} + 0{,}22471,$$
$$p_2(t) = -0{,}5642\sin(0{,}2541t + 2{,}6592)e^{-3{,}48t} - 0{,}041e^{-11{,}12t} + 0{,}30371,$$
$$p_3(t) = -0{,}1244\sin(0{,}2541t + 1{,}0199)e^{-3{,}48t} + 0{,}062e^{-11{,}12t} + 0{,}14385.$$

Function dependency graphs $p_0(t), p_1(t), p_2(t), p_3(t)$ over time is shown in Figure 1.15.

It is clear that in the case of unlimited growth of t the function $p_i(t)$ cease to depend on time and the system enters a stationary regime. That is, $\lim\limits_{t\to\infty} p_i(t) = p_i, i = 0,1,2,3,$ where p_i – constants. From the analysis of expressions for functions $p_i(t)$ it follows that when $t \to \infty$ they begin to equal the corresponding constants on their right-hand side:

$$p_0 = 0{,}327\,88,\ p_1 = 0{,}224\,71,\ p_2 = 0{,}303\,71,\ p_3 = 0.143\,85.$$

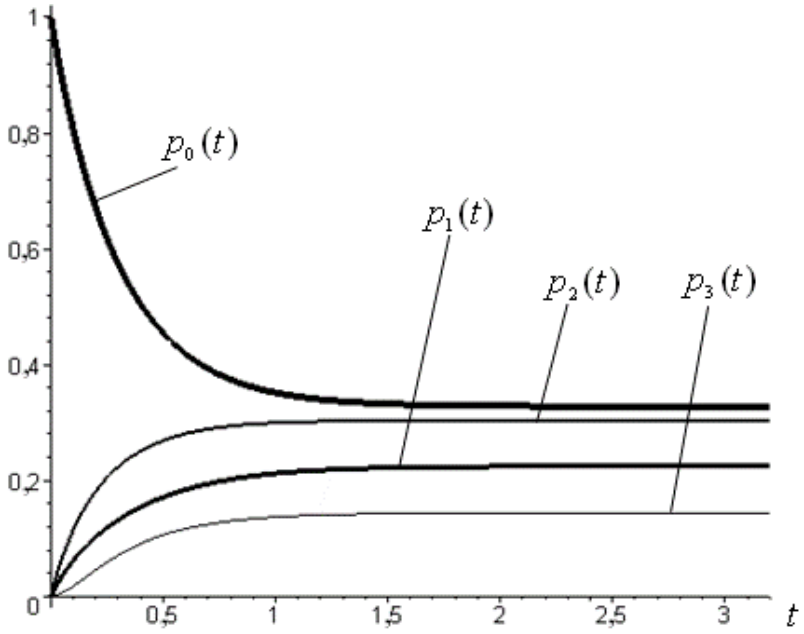


Figure 1.15 – Dependences of the probabilities of system states on time

The probabilities of the states of the system in the steady state can be calculated by a simpler method. Given that $\lim\limits_{t\to\infty} p_i'(t) = \lim\limits_{t\to\infty} p_i = 0$, the system of equations (1.46) changes as follows:

$$\begin{aligned}
&-(\lambda_{01} + \lambda_{02})p_0 + \lambda_{10} \cdot p_1 + \lambda_{20} \cdot p_2 = 0,\\
&-(\lambda_{10} + \lambda_{13})p_1 + \lambda_{01} \cdot p_0 + \lambda_{31} \cdot p_3 = 0,\\
&-(\lambda_{20} + \lambda_{23})p_2 + \lambda_{02} \cdot p_0 + \lambda_{32} \cdot p_3 = 0,\\
&-(\lambda_{31} + \lambda_{32})p_3 + \lambda_{13} \cdot p_1 + \lambda_{23} \cdot p_2 = 0,
\end{aligned} \qquad (1.47)$$

that is, it will pass into a system of linear algebraic equations with respect to $p_i, i = 0,1,2,3$, which is much easier to solve than a system of differential equations. To find the probabilities of stationary states $p_i, i = 0,1,2,3$ it is necessary to solve the system of linear algebraic equations (1.47) by replacing one of the equations

with the normalization condition $p_0 + p_1 + p_2 + p_3 = 1$. As a result, we obtain the following system of linear algebraic equations regarding probabilities:

$$\begin{gathered} -(\lambda_{01} + \lambda_{02})\, p_0 + \lambda_{10} \cdot p_1 + \lambda_{20} \cdot p_2 = 0, \\ -(\lambda_{10} + \lambda_{13})\, p_1 + \lambda_{01} \cdot p_0 + \lambda_{31} \cdot p_3 = 0, \\ -(\lambda_{20} + \lambda_{23})\, p_2 + \lambda_{02} \cdot p_0 + \lambda_{32} \cdot p_3 = 0, \\ p_0 + p_1 + p_2 + p_1 = 1. \end{gathered} \tag{1.48}$$

At $\lambda_{01} = 0{,}8;\ \lambda_{02} = 1{,}5;\ \lambda_{10} = 1{,}6;\ \lambda_{13} = 2{,}0;\ \lambda_{20} = 1{,}3;\ \lambda_{23} = 2{,}5;\ \lambda_{31} = 3{,}8;\ \lambda_{32} = 4{,}6$ the solution to the system of linear algebraic equations will look like this:

$$p_0 = 0{,}327\,89,\ p_1 = 0{,}224\,70,\ p_2 = 0{,}303\,56,\ p_3 = 0{,}143\,85$$

This result, with an accuracy of 10^{-4}, coincides with the result obtained by solving the system of differential equations at $t \to \infty$.

From the analysis of the graphs in Figure 1.15 it follows that most of the operating time of the system described by the Markov process is in a stationary state. Therefore, such systems are usually studied in a stationary state, which is described by a system of linear algebraic equations. If a marked graph of states and transitions is constructed, then for it a system of linear algebraic equations can be constructed regarding stationary probabilities. In this system, for the probabilities of each state $\Psi_i - p_i$ a linear algebraic equation is formed according to the following rule:

zero is placed on the right side, and one on the left – the product of the sum of the transition intensities from state Ψ_i to other states by the probability p_i, *taken with a minus sign, plus the sum of the products of the transition intensities from other states to state Ψ_i and the probabilities of the corresponding states.*

Mathematically, this can be written as follows:

$$-(\lambda_{ie} + \lambda_{id} + ... + \lambda_{is}) \cdot p_i + \sum_k \lambda_{ki} \cdot p_k = 0. \tag{1.49}$$

One of the equations must be replaced by the normalization condition

$$\sum_k p_k = 1. \tag{1.50}$$

It is advisable to solve the formed system of linear algebraic equations using a software application with appropriate capabilities, for example, the Matlab computer mathematics system.

1.3 Event flows and their characteristics

The process of receiving requests for service in the queuing system is probabilistic and forms a stream of homogeneous or heterogeneous events that occur at random intervals of time (Figure 1.16, where ti are the moments of receipt of requests that form a stream of random events; ξi are the time intervals between neighboring requests).

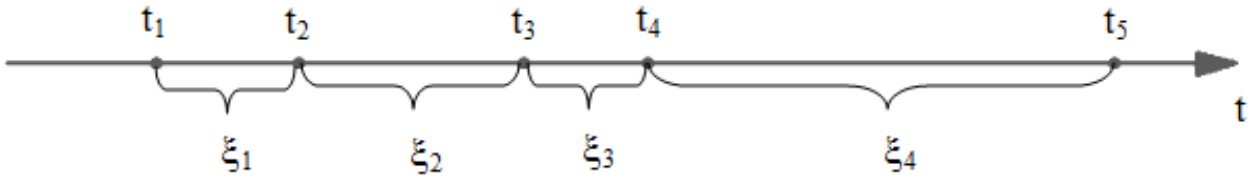


Figure 1.16 – Process of receiving service requests

The basic parameter of the flow is the flow intensity λ(t) – the mathematical expectation of the number of events per unit of time:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{m_k(t, t+\Delta t)}{\Delta t}, \qquad (1.51)$$

that is, it is the limit of the ratio of the mathematical expectation of the number of events – $m_k(t, t+\Delta t)$ on the interval (t, t + Δt) to the length of this interval approaching zero.

If the values ξ_i are independent and identically distributed, then such a flow is called a flow with bounded aftermath (Palma flow). Thus, the basic description of a flow is the distribution function $A(t) = P\{\xi < t\}$. Such flows are central to the theory of queuing.

Each request received for service is on the service device for a random amount of time η, which is described by the distribution function $B(t) = P\{\eta < t\}$. If we assume that there are requests at the input to the service device all the time, then at the output of the service device a flow of serviced requests with the distribution function B(t) will be formed. Therefore, service occurs as if a service flow is directed to a request that has arrived for service. This formulation of the problem is very convenient methodologically, since it allows for a generalized consideration of the processes of receipt and service. Thus, in the queueing system, two types of flows operate – receipt and service, which are characterized by different distribution functions.

Random time intervals between the occurrence of events in a flow can be characterized by different distribution laws. However, most works on the theory of queuing, especially applied ones, consider the Poisson (simplest) flow, in which the

probability that exactly k requests will occur in a time interval t is given by the Poisson formula:

$$p_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \qquad (1.52)$$

where $\lambda > 0$ is a flow parameter.

The simplest flow is characterized by three main properties: stationarity, absence of aftereffects, and ordinality.

A random flow is called stationary if the probability of receiving a certain number of requests during a certain time interval depends on its value and does not depend on the start of its reference on the time axis. This means: if we set aside non-intersecting levels, time intervals τ (Figure 1.17), on the time axis, then the probability of the appearance of a certain number of requests for this flow in these intervals depends on the value τ and does not depend on the location of this interval on the time axis (from the moments of time t_1, t_2, t_3....).

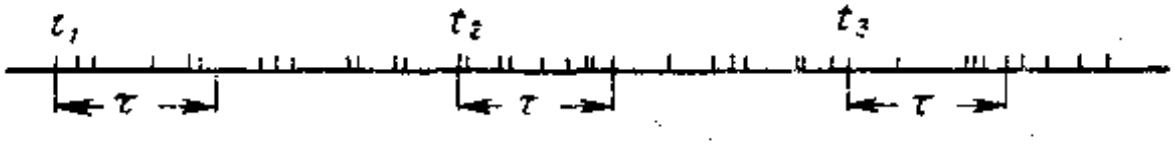


Figure 1.17 – Steady flow of homogeneous events

For steady flow

$$\lambda(t) = \lambda = 1/m \ , \qquad (1.53)$$

where m is the mathematical expectation of the time intervals between neighboring flow events, which is calculated by formula (1.14).

Thus, the two simplest flows differ only in their parameters, so to specify the simplest flow, it is sufficient to specify only its parameter λ.

The absence of after effect is that the probability of receiving a certain number of requests during a time interval τ does not depend on how many requests have already entered the system, that is, it does not depend on the history of the phenomenon under study. The absence of aftereffect implies mutual independence of the process during non-overlapping time intervals.

The ordinariness of the flow of requirements means the practical impossibility of two or more requirements appearing at the same time.

Thus, the simplest flow is a stationary, ordinary flow without aftereffects.

Let's calculate the intensity of the simplest flow. In the interval Δt either one or an infinite number of events can occur. According to formula (1.7),

$$m_k(t,t+\Delta t)=\sum_{k=1}^{\infty}kp_k(\Delta t)=\sum_{k=1}^{\infty}k\frac{(\lambda\Delta t)^k}{k!}e^{-\lambda\Delta t}=e^{-\lambda\Delta t}\sum_{k=1}^{\infty}k\frac{(\lambda\Delta t)^k}{k!}=e^{-\lambda\Delta t}\left(\lambda\Delta te^{\lambda\Delta t}\right)=\lambda\Delta t.$$

From here $\lambda=m_k(t,t+\Delta t)/\Delta t=\lambda\Delta t/\Delta t=\lambda$, that is, the intensity of the simplest flow coincides with its parameter.

Let's calculate the distribution function of a random variable ξ – time intervals between neighboring events of a Poisson flow $A(t)=P\{\xi<t\}$. Function $A(t)=P\{\xi<t\}$ is the probability that at least one flow event will occur in the time interval t adjacent to the moment of the next event. The probability of this is $\sum_{k=1}^{\infty}p_k(t)$. Because $\sum_{k=0}^{\infty}p_k(t)=1$, then $A(t)=1-p_0(t)=1-\frac{(\lambda t)^0}{0!}e^{-\lambda t}$. Finally – $A(t)=1-e^{-\lambda t}$, that is, the distribution function of the random variable – the time intervals between neighboring events of the Poisson flow is given by an exponential distribution. The density of the distribution is $a(t)=A'(t)=\lambda e^{-\lambda t}$, mathematical expectation – $m=1/\lambda$, dispersion – $D=1/\lambda^2$ standard deviation – $\sigma=1/\lambda$, coefficient of variation – $\nu=\sigma/m=1/$

The exponential distribution has one property: if an exponentially distributed time interval lasts for time τ, then this does not affect the distribution function of the remaining interval *T*. It will be the same as the distribution function of the entire interval, that is: if the time interval varies exponentially, then any information about the time of the passage on some interval does not affect the distribution function in the remaining part. It has been proven that only the exponential distribution has this property. This greatly simplifies mathematical calculations when analysing queuing systems.

The simplest flow in the theory of queuing plays the same role as the normal law of distribution of random variables in the theory of probability. In the case of summing (mutual superposition) of a large number of ordinary stationary flows, a flow is formed, which is close to the simplest, with virtually any distributions [14].

The simplest flow plays an important role in modelling theory. First, the simplest and near-simplest event flows are very common in practice. Second, even with a flow of events that is different from the simplest, one can usually find results that are satisfactory in terms of accuracy by replacing the flow of any structure with the simplest flow with the same average intensity.

In modelling practice, the following distributions are used to describe real flows, in addition to the exponential one:

1. Erlang with distribution function:

$$A(t) = 1 - \sum_{i=1}^{k} \frac{(\lambda t)^{i-1}}{(i-1)!} e^{-\lambda t}. \quad (1.54)$$

The Erlang distribution is formed by summing *k* intervals, each of which is distributed according to an exponential law with parameter λ. Therefore, the flow of events described by the Erlang distribution of the *k*-th order is formed from the Poisson distribution by removing all events contained between the *l*-th and (*l* + *k*)-th events. At *k* = 1, the Erlang distribution becomes exponential.

Let's find the density of the Erlang distribution $a(t)$:

$$a(t) = A'(t) = -\sum_{i=1}^{k} \left(\frac{(i-1)\lambda^{i-1} t^{i-2}}{(i-1)!} e^{-\lambda t} - \frac{\lambda^i t^{i-1}}{(i-1)!} e^{-\lambda t} \right) =$$

$$= -e^{-\lambda t} \sum_{i=1}^{k} \left(\frac{(i-1)\lambda^{i-1} t^{i-2}}{(i-1)!} - \frac{\lambda^i t^{i-1}}{(i-1)!} \right) = -e^{-\lambda t} \left(0 + \lambda + \lambda^2 t + \frac{1}{2}\lambda^3 t^2 + \frac{1}{6}\lambda^4 t^3 + \ldots \right) +$$

$$+ e^{-\lambda t} \left(\lambda + \lambda^2 t + \frac{1}{2}\lambda^3 t^2 + \frac{1}{6}\lambda^4 t^3 + \ldots + \frac{1}{(k-1)!}\lambda^k t^{k-1} \right).$$

Finally we will get

$$a(t) = \frac{1}{(k-1)!} \lambda (\lambda t)^{k-1} e^{-\lambda t}. \quad (1.55)$$

Mathematical expectation – $m = k/\lambda$, standard deviation $\sigma = \sqrt{k}/\lambda$, coefficient of variation – $\nu = \sqrt{k}/k \le 1$.

This distribution can be used to describe quite accurately the flows of events with a variation coefficient ν less than one.

At $k \to \infty$ an Erlang flow becomes regular when events occur at regular intervals.

Figure 1.18 shows the density graphs of the Erlang distribution at $\lambda = 0{,}1$ and different *k*.

Unlike the exponential distribution (k = 1), the density graph of the Erlang distribution has a maximum (mode). As *k* increases, the mode shifts to the right.

2. Hyperexponential – with distribution function

$$A(t) = 1 - \sum_{i=1}^{k} a_i e^{-\lambda_i t}, \ \sum_{i=1}^{k} a_i = 1 \quad (1.56)$$

and distribution density

$$a(t) = \sum_{i=1}^{k} a_i \lambda_i e^{-\lambda_i t}, \ \sum_{i=1}^{k} a_i = 1. \quad (1.57)$$

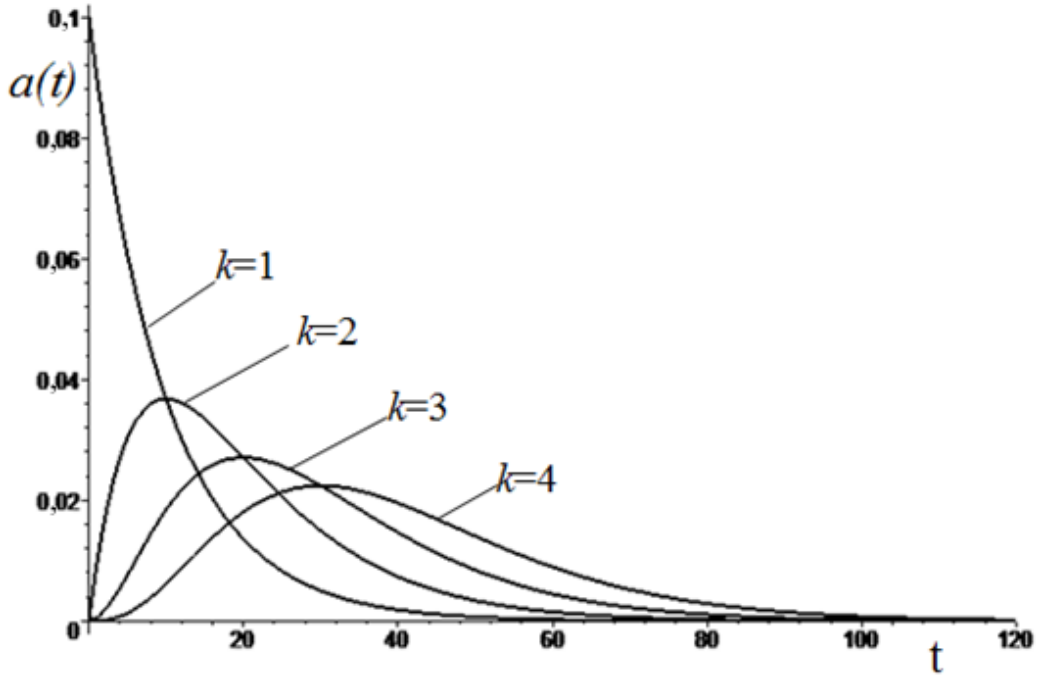


Figure 1.18 – Density plots of the Erlang distribution

Mathematical expectation – $m = \sum_{i=1}^{k} \frac{a_i}{\lambda_i}$,

coefficient of variation – $\nu = \left[2\sum_{i=1}^{k} \frac{a_i}{\lambda_i^2} - \left(\sum_{i=1}^{k} \frac{a_i}{\lambda_i} \right)^2 \right]^{0,5} \Bigg/ \sum_{i=1}^{k} \frac{a_i}{\lambda_i}$, $\nu \geq 1$.

It is recommended to use a second-order hyperexponential distribution, which has only two parameters λ and φ with distribution function [20]:

$$A(t) = 1 - \varphi e^{-2\varphi\lambda t} - (1-\varphi)e^{-2(1-\varphi)\lambda t},\ 0 < \varphi \leq 0,5, \qquad (1.58)$$

That is $a_1 = \varphi,\ a_2 = 1-\varphi,\ \lambda_1 = 2\varphi\lambda t,\ \lambda_2 = 2(1-\varphi)\lambda t$. In this case, the mathematical expectation is $m = 1/\lambda$, coefficient of variation – $\nu = \left[1 + \frac{(1-2\varphi)^2}{2\varphi(1-\varphi)} \right]^{0,5}$.

If the mathematical expectation of the flow *m* and its coefficient of variation *v* are given, then the parameters λ and φ can be determined by the formulas

$$\lambda = 1/m, \quad \varphi = 0,5 - 0,5\sqrt{\frac{\nu^2 - 1}{\nu^2 + 1}}. \qquad (1.59)$$

The flow of events described by a hyperexponential distribution is formed if with probability a_i, $\sum a_1 = 1$, the next time interval between the occurrence of adjacent events is chosen to be distributed according to an exponential law with parameter λ_i. At *k* = 1, the hyperexponential distribution turns into an exponential

one. The hyperexponential distribution describes quite accurately the flows of events that have a coefficient of variation greater than unity.

3. Pareto – with distribution function

$$\begin{cases} F(t) = \left\{1 - (t_m / x)^{\alpha}\right\}, t \geq t_m, \\ 0, \qquad\qquad t < t_m, \end{cases} \tag{1.60}$$

and distribution density

$$\begin{cases} a(t) = \dfrac{\alpha \cdot t_m^{\alpha}}{t^{\alpha+1}}, \quad t \geq t_m, \\ 0, \qquad\qquad t < t_m. \end{cases} \tag{1.61}$$

Mathematical expectation – $m = \dfrac{\alpha \cdot t_m^{\alpha}}{\alpha - 1}$, exists when $\alpha > 1$.

Standard deviation – $\sigma = \dfrac{x_m}{\alpha - 1}\sqrt{\dfrac{\alpha}{\alpha - 2}}$, exists when $\alpha > 2$.

Coefficient of variation – $\nu = \dfrac{1}{\sqrt{\alpha(\alpha - 2}}$.

If the mathematical expectation *m* and the variation coefficient ν are given, then the parameters α and t_m can be determined using the following formulas:

$$\alpha = 1 + \sqrt{1 + \frac{1}{\nu^2}}, \quad t_m = m \cdot \left(1 - \frac{1}{1 + \sqrt{1 + \dfrac{1}{\nu^2}}}\right).$$

The distribution has two parameters: x_m is the minimum possible value of the random variable, and α is the shape or Pareto index, a coefficient that determines the rate of decay of probability. The smaller α, the heavier the tail of the distribution (greater inequality). The Pareto distribution is a continuous probability distribution that describes social, economic, and natural phenomena in which a small number of causes produce the majority of consequences. It is the mathematical basis for the famous "Pareto rule" (the 80/20 principle), which states that 20% of the effort produces 80% of the results, or that 20% of the population possesses 80% of the wealth. It is widely used to describe highly uneven and "heavy-tailed" random processes occurring in information technology and the internet (number of links to pages, file size on the web).

2 BASIC CONCEPTS OF QUEUING SYSTEMS AND METHODS OF THEIR RESEARCH

2.1 Basic concepts of queuing systems

Formally, a queueing system (here in after referred to as a queueing system) is understood to be a complex system consisting of one or more sources of requests (applications, requirements, customers) for the performance of certain actions (service), several service devices (service lines, service channels) that perform these actions in accordance with certain rules (service discipline) for requests received by the system (Figure 2.1).

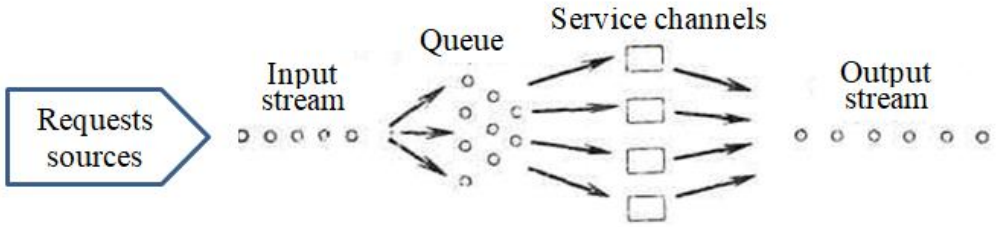


Figure 2.1 – Structure of a queuing system

A characteristic feature of queuing systems is the probability of the processes that occur and the possibility of forming a queue of service requests. Let's consider the basic concepts of queuing systems.

Source of requests. A query source is defined as a system external to the queuing system from which queries are received for service. A source is called infinite or finite depending on whether it contains an infinite or finite number of queries. If a source contains a finite but sufficiently large number of queries, then it is usually considered infinite. For example, although the number of users of the information search engine Google is finite, it is assumed that they form an infinite source.

Input stream. Requests coming from an infinite source arrive at the service channel at times $t_0 < t_1 < ... < t_k < ....$ Time intervals $\xi_k = t_k - t_{k-1} (k > 1)$ between consecutive moments of request arrival are random variables. It is assumed that ξ_k form a sequence of independent and identically distributed random variables with distribution function $A(t) = P\{\xi_k < t\}$. In the mathematical theory of queuing, a limited set of distribution laws is used to describe incoming flows. The main ones are the Poisson (the simplest), which is given by the Poisson formula (1.52).

Service process. Let η_k be the duration of service of the k-th request. It is assumed that $\eta_k, (k = 1,2,3,...)$ are independent identically distributed random

variables with the distribution function $B(t) = P\{\eta_k < t\}$. The function B(t) is called the service duration distribution. We assume that there exists a probability density $b(t) = B'(t)$. As for describing incoming request flows, queuing theory uses a limited set of distribution laws to describe service times. The most common is the exponential distribution law with the distribution function $B(t) = 1 - \exp(-\mu t)$. The parameter μ is interpreted as the intensity of service. The average service time is 1/μ. In addition to the exponential distribution, the Erlang and hyperexponential distributions are used, which are obtained by transforming the exponential law.

Service channels. A queuing system may have one or more service channels (lines, devices). Queuing systems with one channel are called single-channel or single-line, while service systems containing more service channels are called multi-channel or multi-line. Service devices can be homogeneous or heterogeneous. In a queuing system with homogeneous devices, all devices service requests equally. In a queuing system with heterogeneous devices, devices differ from each other in some parameters, such as service intensity.

Service discipline. The rule according to which requests are selected for service constitutes a service discipline. Conventionally, all service disciplines are divided into two groups according to the service preferences provided: non-priority and priority. Each of these groups is divided into a number of subgroups.

Non-priority service disciplines are divided into first-come, first-served, random-choice, and cyclic service disciplines.

The discipline of first-in, first-out (FIFO – First Input First Output) is the most common, most research on queuing theory is done for this discipline. It is widely used in operating systems.

The discipline of servicing in reverse (inverse) order (in the international notation LIFO – Last Input First Output) is used in operating systems during interrupt handling and stack organization.

Cyclic service disciplines are used during time-sharing information processing.

In priority service disciplines, requests with the highest priority are selected from the service queue first. They are divided into disciplines with fixed and dynamic priorities [15].

With the service discipline with relative priority, interruption of service of a request on the channel is not allowed.

If a system with absolute priority service discipline receives a request with a higher priority than the one being serviced, it will stop servicing that request and go

to service. Absolute priority systems are distinguished by the number of priority levels, as well as by the algorithms for servicing interrupted requests.

In service disciplines with dynamic priority, the priority of specific requests changes depending on changes in some quantities, such as waiting time in queues.

According to the presence of a certain characteristic, a queueing system can be classified as follows:

1. By the number of requests received per unit of time, into systems with ordinary and extraordinary flows of requests. If the probability of receiving two or more requests simultaneously is zero or is so small that it can be neglected, then we will obtain a system with an ordinary flow of requests. For example, the flow of requests – airplanes arriving at the runway of an airfield can be considered ordinary.

2. By the connection between the requirements – into systems without after-effect and with after-effect. If the probability of requirements entering the system at some point in time does not depend on how many requirements have already been received, that is, it is not related to the history of the process under study, then we will get a system without after-effect, otherwise – with after-effect. An example of a system with after-effect can be the flow of students passing a test to a teacher.

3. According to the response of the request to the busyness of the channels – to systems with failures and waits. If the request that arrived for service found all the channels busy and was forced to leave the system, then we will get a system with failures.

4. Systems with waiting are divided into systems with limited and unlimited waiting. If a request leaves the system when the queue has reached a certain size, then we get a system with limited waiting. An example is a dump truck with mortar. If the waiting time is so long that the mortar can harden, then it is appropriate to unload the dump truck elsewhere. If the request that has arrived finds all the channels busy and is forced to wait for its turn until it is served, then we get a system with unlimited waiting. Example: an airplane at an airfield waiting for the runway to be cleared.

5. According to the method of selection, service requirements are distributed as follows:

- with priority of requirements;
- in the process of receiving claims;
- with a random selection of requirements;
- the last requirement is served first.

If the queuing system covers several categories of requirements and the order in which they are selected for service is determined by some criteria, then we will get a system with priority of requirements. For example, when products arrive at a construction site, those that are determined by the construction technology are installed first.

If the channel that has been released serves a request that arrived in the system earlier than the others, then we will get a system that services requests as they arrive. For example, the buyer who approached the seller first is served first.

If requests from the queue arrive at the service channel randomly, then we will have a system with a random selection of service requests. Example: a plumber choosing one of several requests from residents, the time of receipt of which he is not familiar with.

If the last request received is selected for service, we get a system with a "last-served-first-served" selection. For example, when stacking construction products, it is more convenient to select the last product from the stack (queue).

6. By the time of service of the request – into systems with deterministic and random service time. If the time interval between the moment of receipt of the request in the service channel and the moment of the request's exit from the channel is constant, then we will get a system with deterministic service time, otherwise – with random. For example, car washing is a service system with deterministic service time.

7. By the number of service channels – into single and multi-channel systems. For example, during the installation of a house, one lifting crane (one service channel) or several (many service channels) can be used.

8. By the number of service stages – into single- and multi-phase systems. If the service channels are arranged sequentially and heterogeneously, then we get a multi-phase service system. An example of such a system is car service at a service station (washing, diagnostics, filter replacement, etc.).

9. According to the homogeneity of requirements – into systems with homogeneous and heterogeneous flows of requirements. For example, if vans of the same carrying capacity arrive for loading, then we will get a system with a homogeneous flow of requirements, if different – with heterogeneous.

10. By channel load – into ordered and unordered systems. In ordered systems, the serving channels are loaded unevenly. The incoming request is served by a clearly defined channel from the available free ones, namely the channel with the lowest number (it is assumed that all channels are numbered). In unordered

systems, all channels are the same and the incoming request is served by one of the free channels without any advantages.

For the abbreviated designation of open non-priority unordered queuing systems, D. Kendall proposed the following notation: $A/B/n/m$. A and B describe, respectively, the laws of distribution of time intervals between events of the incoming request flow and the service time, n specifies the number of service channels, m is the number of waiting places.

Certain distribution laws are denoted as follows: M – exponential distribution, E or E_k – Erlang distribution; H or H_r – hyperexponential distribution; D – regular event flow; G – general distribution. For example, the entry $D/H_2/3/10$ denotes a system with three service channels, constant (deterministic) time between requests of the input flow, service time distributed according to the second-order hyperexponential law and the number of waiting places – 10. If the parameter m is absent, this means that the number of waiting places is infinite.

2.2 Performance indicators of queueing systems

The efficiency indicators of queuing systems are divided into technical ones, which characterize the quality and operating conditions of the service system, and economic ones, which reflect the economic features of the system.

The indicators of the first group are usually formed on the basis of the values of the system state probabilities obtained from calculations. The indicators of the second group are based on the indicators of the first group.

2.2.1 Technical indicators of efficiency of queueing systems

Among the technical indicators, the following can be distinguished:

1) Probability of service failure. The probability that a request entering the system will refuse to join the queue and be lost by the system is Pfail. This indicator for a queuing system with failures is equal to the probability that the system contains as many requests as there are service channels:

$$P_{fail} = P_n, \tag{2.1}$$

where n is the number of service channels.

For a system with a finite queue length, Pvidm is equal to the probability that the system contains n + m requests:

$$P_{fail} = P_{n+m}, \tag{2.2}$$

where m is the permissible queue length.

The opposite indicator is the probability of servicing the requirement:

$$P_{serv} = 1 - P_{fail}; \quad (2.3)$$

2) average number of requests awaiting service:

$$L_q = \sum_{i=n+1}^{n+m} (i-n) p_i , \quad (2.4)$$

where pi is the probability that the system contains and requirements;

3) relative and absolute system throughput, which are determined by the following formulas:

– relative throughput:

$$Q = 1 - P_{fail}, \quad (2.5)$$

– absolute throughput:

$$A = \lambda \cdot Q; \quad (2.6)$$

4) average number of channels occupied by service for G/G/n/m systems (i.e. n – service devices, m – waiting places):

$$\bar{n} = \sum_{i=1}^{n-1} i p_i + n \sum_{i=n}^{n+m} p_i . \quad (2.7)$$

For queued systems with failures, the average number of channels occupied by service can be found by the formula:

$$\bar{n} = \sum_{i=1}^{n} i \cdot p_i ; \quad (2.8)$$

5) the total number of requirements contained in the system L_s. This indicator is defined as follows:

– for queueing systems with failures:

$$L_s = \bar{n}; \quad (2.9)$$

– for queue-limited queuing systems:

$$L_s = \bar{n} + L_q; \quad (2.10)$$

6) average waiting time for service start requests. If the probability distribution function for waiting time for service start requests W(t) = P(Tq <t) is known, then the average waiting time for service start requests Tq is defined as the mathematical expectation of the random variable T_q:

$$\tau_q = M[T_q] = \int_0^{\infty} t dW(t). \quad (2.11)$$

With an exponential law, the distribution of requirements in the input stream τ_q can be determined by the formula

$$\tau_q = \frac{L_q}{\lambda}. \quad (2.12)$$

2.2.2 Economic indicators of the efficiency of queueing systems

Indicators characterizing the economic features of queueing systems are usually formed in accordance with a certain type of system and its purpose. One of the general indicators is the economic efficiency of the system:

$$G = \lambda P_{serv} C_s T - E, \quad (2.13)$$

Where C_s – average economic effect obtained when servicing one request; T – considered time interval; E – value of losses in the system.

The latter value (loss) can be determined as follows:

– for systems with failures

$$E = (c_o \bar{n} + \lambda c_{loss} P_{fail} + c_{idle} \bar{n}_{idle}) T, \quad (2.14)$$

where c_o – cost of operating one channel per unit of time; c_{loss} – cost of losses due to demands leaving the system per unit of time; c_{idle} – cost of unit of service channel downtime; $\bar{n}_{idle}$ – average number of free channels (idle), $\bar{n}_{idle} = n - \bar{n}$;

– for systems with waiting:

$$E = (c_o \bar{n} + c_q L_q \lambda + c_{idle} \bar{n}_{idle}) T, \quad (2.15)$$

where c_q is the cost of losses associated with the request being idle in the queue per unit of time.

2.3 Methods of researching queueing systems

The main task of the study of queueing systems is to establish the dependence of the basic characteristics, namely: average waiting time, probability of service failure on the system parameters – service intensities on devices, number of service devices, service discipline. The diversity of queueing systems makes it impossible to use a single method for analysis. Although in practice, one method is usually used for a certain group of queueing systems, which allows to minimize calculations and reduce the time for determining basic dependencies.

2.3.1 Differential method

The differential method can be used to study the queueing systems with Poisson event flows, i.e. the service time and the time intervals between the arrival of requests are subject to an exponential distribution. In this case, the process of

changing the number of requests in the system is Markov and can be studied using the methodology described in subsection 1.3. The sequence of the study is as follows:

– from the analysis of the service system, the states of the system, possible transitions between states and their intensities are separated and a graph of states and transitions is drawn up (Figure 1.14);

– Each state is associated with a corresponding probability and a system of differential equations for the probabilities of the states of the system is constructed using the method described in subsection 1.2.3 (hence the name of the method). It is advisable to solve and investigate the complex system using one of the computer mathematics systems, for example Maple.

If the study of the system in the stationary mode is planned in advance, then again (using the method of subsection 1.2.3) a system of linear algebraic equations for the stationary probabilities of states is compiled. Usually the solution can be reduced to calculation formulas.

Let us consider the use of the differential method for the analysis of the queueing systems on a service system with a group selection of requests for service This queueing systems consists of a storage, into which a Poisson flow of requests with intensity λ enters. After some time intervals, the accumulated requests are selected for service. The capacity of the storage is *k* requests. The intensity of the selection of requests from the storage is μ_i (i = 0,1, .., k), i.e. it is determined by the number of requests in it. This queueing system can be used, for example, as a model of the operation of a matching buffer for an information system.

Let s_i – the state when the storage contains *i* requests, the probability of this $p_i, i = 1, 2, ...k$. This queueing system, in accordance with the receipt of a new request, transitions from the state s_i in the state s_{i+1}, the intensity of this transition is λ. When selecting requests from the storage, a transition occurs from the state s_i $(i = 1, 2, ...k)$ into the state s_0. The intensity of this transition is μ_i. The graph of states and transitions for this queueing system at $k = 4$ is shown in Figure 2.2.

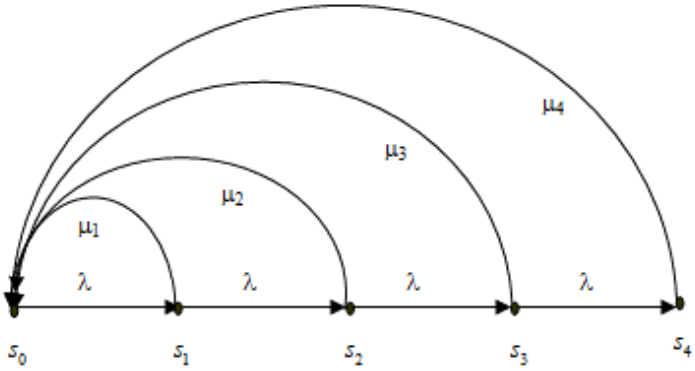


Figure 2.2 – State and transition graph for a group choice system

Having compiled a system of differential equations for the studied queueing system, we obtain:

$$\begin{aligned} & p_0'(t) = -\lambda p_0(t) + \sum_{i=1}^{k} \mu_i p_i(t), \\ & p_i'(t) = -(\lambda + \mu_i) p_i(t) + \lambda p_{i-1}(t), \ 1 \le i \le k, \\ & p_k'(t) = -\mu_k p_k(t) + \lambda p_{k-1}(t), \end{aligned} \tag{2.16}$$

Initial conditions – $p_0(0) = 1, \ p_i(0) = 0, \ i \neq 0.$

For the steady state, system (2.16) becomes as follows:

$$\begin{aligned} & -\lambda p_0 + \sum_{i=1}^{k} \mu_i p_i = 0, \\ & -(\lambda + \mu_i) p_i + \lambda p_{i-1} = 0, \ 1 \le i \le k, \\ & -\mu_k p_k + \lambda p_{k-1} = 0. \end{aligned} \tag{2.17}$$

The normalization condition is

$$p_0 + p_1 + p_2 + ... + p_k = 1. \tag{2.18}$$

From the equations of the system (2.17), starting from the second one, we sequentially obtain:

$$p_1 = \frac{\lambda}{(\lambda + \mu_1)} p_0, \qquad p_2 = \frac{\lambda}{(\lambda + \mu_2)} p_1 = \frac{\lambda^2}{(\lambda + \mu_1)(\lambda + \mu_2)} p_0,$$

$$p_i = \frac{\lambda^i}{\prod_{e=1}^{i} (\lambda + \mu_1)} p_0, \quad p_k = \frac{\lambda^k}{\mu_k \prod_{e=1}^{k-1} (\lambda + \mu_e)} p_0 .$$

By substituting the values p_i, expressed through p_0, into the normalization condition (2.18), we obtain:

$$p_0 = \left[1 + \sum_{i=1}^{k-1} \lambda^i \Big/ \prod_{e=1}^{i} (\lambda + \mu_e) + \lambda^k \Big/ \mu_k \prod_{e=1}^{k-1} (\lambda + \mu_e) \right]^{-1}.$$

Basic characteristics of the system under study:

– probability of service failure – $P_{fail} = p_k$;

– average number of requests in the drive – $L_q = \sum_{i=1}^{k} i p_i$;

– average time requests stay in the storage – $\tau_q = \sum_{i=1}^{k} \frac{p_i}{\mu_i}$.

Using these expressions, you can choose the buffer size so that the probability of its overflow is within specified limits.

2.3.2 Phase method

The phase method allows us to study a more general type of queueing systems than the Poisson one by introducing additional states. This allows us to use the method of formulating equations for additional states, discussed in section 1.2.3, for such an extended system. The phase method is usually used when the event flows in the queueing system are described by Erlang or hyperexponential distributions. For the phase method, four basic ways of transitioning to an extended system of equations can be distinguished:

1. The service time is distributed according to the *k*-order Erlang distribution (1.54) with parameter μ. In this case, the service channel is conditionally replaced by *k* separate fictitious channels (phases) arranged sequentially. Each request accepted for service passes sequentially through these *k* phases, spending a period of time in each phase that obeys an exponential distribution with parameter μ. A new request is not serviced until the previous one has passed all *k* service phases. The queueing system in this case is described by a Markov process with states s_{ij}, $1 \le j \le k$. Condition s_{ij} means that the system contains *i* requests, and the next request is served in the *j*-th phase. The state probabilities corresponding to this process are $p_{ij}(t)$. Figure 2.3 shows the state and transition graph for a single-line queueing system with a second-order Erlang service flow $M/E_2/1$, for which a system of equations can be easily constructed. If the probabilities $p_{ij}(t)$ are found, then the probabilities $p_i(t)$ then it can be determined by the formula $p_i(t) = \sum_{i=1}^{k} p_{ij}(t)$.

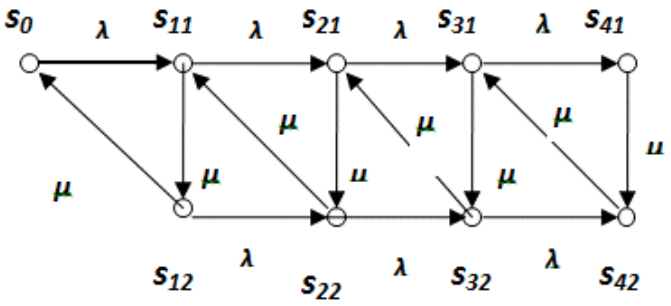


Figure 2.3 – Extended state and transition graph of the M/E$_2$/1 system

2. The service time is distributed according to a hyperexponential distribution of the *k*-th order of the form (1.56) with parameters b_i , μ_i . In this case, the service channel is conditionally replaced by separate service phases with parameters μ_i, located in parallel. With probability b_i each request is served in the *i*-th phase. State and transition graph for a single-line queueing system with second-order hyperexponential service flow $M/H_2/1$ shown in Figure 2.4.

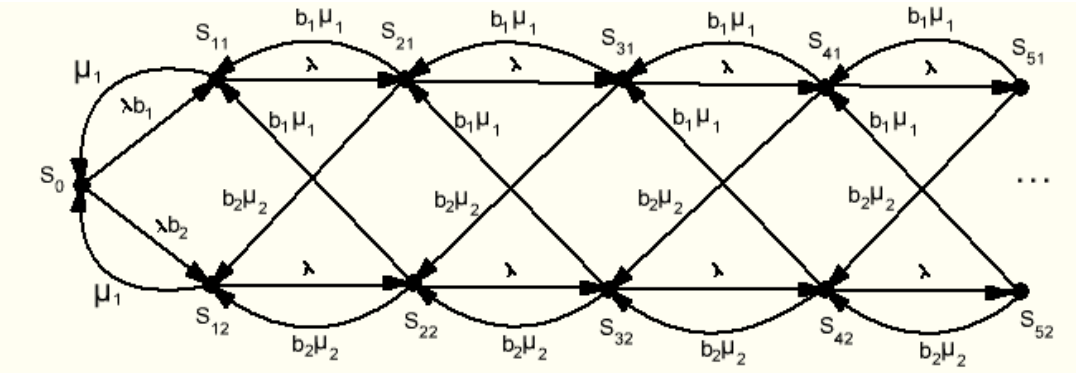


Figure 2.4 – Extended state and transition graph of the M/H2/1 system

3. The input stream is an Erlang *k*-order stream with parameter λ. In this case, a dummy receiver consisting of *k* phases is added to the queueing system. Each request is first sent to this receiver, where it is sequentially delayed at each phase. The delay time at each phase is characterized by an exponential distribution with parameter λ. Only after passing this dummy receiver does the request enter the system for service. The queueing system will be described by a Markov random process s_{ij}, where *j* is the phase number of the incoming request entering the system. The state and transition graph for a single-line queueing system of the form $E_2/M/1$ depicted on Figure 2.5.

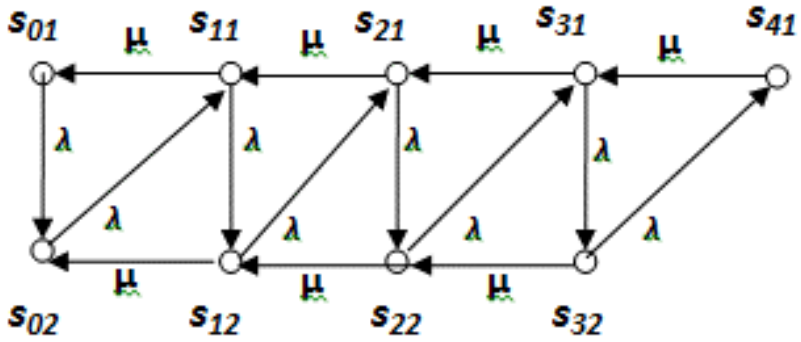


Figure 2.5 – Extended state and transition graph of the $E_2/M/1$ system

4. The input flow is hyperexponential of the k-th order with parameters a_i, λ_i. To transition to a Markov process, a dummy receiver consisting of parallel phases is added to the queueing system. Each service request is first sent to this receiver, where with probability a_i passes the *i*-th phase and then enters service. State and transition graph for a queueing system of the form $H_2/M/1$ shown in Figure 2.6.

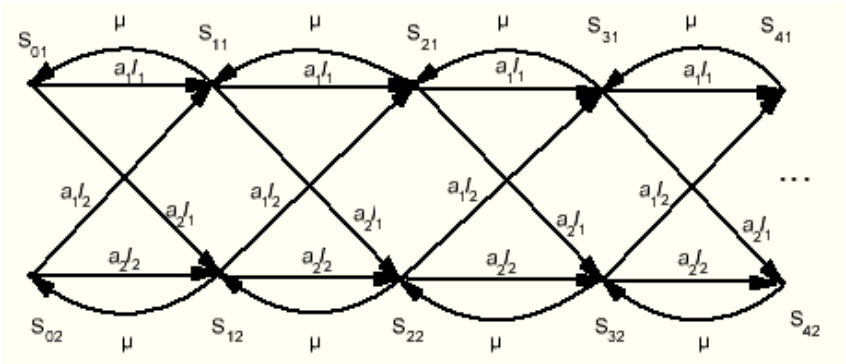


Figure 2.6 – Extended graph of states and transitions of the $H_2/M/1$ system

The main disadvantage of the phase method is the large dimensionality of the region of possible states and the corresponding system of equations, which cannot be solved analytically. The only way to obtain a solution is numerically, using standard subroutines for solving systems of equations on a computer.

To illustrate the use of the phase method, consider the following example. Let us consider a queueing system of the form M/GE/1 where GE is the generalized Erlang distribution symbol containing the Laplace–Stiehljes transform of the distribution function $\beta(s) = \prod_{j=1}^{k} \mu_j / (\mu_j + s)$. It differs from the Erlang distribution in that each stage has its own intensity – μ_j. Denote by $p_n(t)$ the probability of finding *n* requests in the system. Let us introduce fictitious service phases. Then each state can be decomposed into *k* phases, the process of transitions between which is

Markov. Let p_{nj} – stationary probability that the system contains *n* requests, the next request is serviced in the jth phase ($j=\overline{1,k}$). For probabilities, the following system of linear equations is valid, obtained using the method described in subsection 1.2.3:

$$\begin{aligned}
&-\lambda p_0+\mu_1 p_{11}=0,\\
&-(\lambda+\mu_j)p_{1j}+\mu_{j+1}p_{1j+1}=0,\\
&-(\lambda+\mu_k)p_{1k}+\mu_1 p_{21}+\lambda p_0=0,\\
&-(\lambda+\mu_j)p_{nj}+\mu_{j+1}p_{nj+1}+\lambda p_{n-1j}=0,\ n>1,\ 1\le j<k,\\
&-(\lambda+\mu_k)p_{nk}+\mu_1 p_{n+11}+\lambda p_{n-1k}=0,\ n>1.
\end{aligned} \qquad (2.19)$$

The normalization condition is $p_0+\sum_{n=1}^{\infty}\sum_{j=1}^{k}p_{nj}=1.$

To solve the system of equations (2.19), we use the method of generating functions. We multiply each equation of the system (2.19) for p_{nj} $(1\le j\le k)$ on z^n, and for p_0 and p_{nk} – on z^{n+1} and sum the equations with the same index *j* for the same terms. Let us introduce the generating functions $Q_j(z)=\sum_{n=1}^{\infty}p_{nj}z^n$. We obtain a system of equations for $Q_j(z)$:

$$\begin{aligned}
&-(\lambda+\mu_j)Q_j(z)+\lambda zQ_j(z)+\mu_{j+1}Q_{j+1}(z)=0,1\le j<k,\\
&-(\lambda+\mu_k)zQ_k(z)+\lambda z^2Q_k(z)+\mu_1Q_1(z)=\lambda z(1-z)p_0.
\end{aligned}$$

Solving it, we find:

$$Q_j(z)=\frac{\lambda z(1-z)p_0\prod_{i=1}^{j}(\lambda-\lambda z+\mu_i)\coprod_{r=j}^{k}\mu_r}{\left[\prod_{i=1}^{k}\mu_i-z\prod_{i=1}^{k}(\lambda-\lambda z+\mu_i)\right](\lambda-\lambda z+\mu_j)\mu_j}.$$

System boot – $\rho=\lambda\sum_{j=1}^{k}1/\mu_j$ and $p_0=1-\rho.$ Generating function of the stationary probability distribution p_n can be defined as follows:

$Q(z)=p_0+\sum_{j=1}^{k}Q_j(z).$ Then $p_n=\frac{1}{n!}\left[\frac{d^n}{dz^n}Q(z)\right]_{z=0}.$

For example, $p_1 = \frac{\lambda(1-\rho)}{\prod_{j=1}^{k}\mu_j}\sum_{j=1}^{k}\frac{\prod_{i=1}^{k}(\lambda+\mu_i)\prod_{r=j}^{k}\mu_r}{(\lambda+\mu_j)\mu_j}$.

This example, among other things, demonstrates the capabilities of the generating function method, which is widely used to analyze systems of difference equations.

2.3.3 Imbedded Markov chains Method

This method is advisable to use when only one of the event flows in the queueing system (input flow or service flow) is Markov or can be reduced to it by the phase method, and the other is arbitrary. In the method of imbedded Markov chains, a random process in the queueing system is considered not at arbitrary points in time, but at specially designated points – regeneration points (or recovery points), which form a Markov chain [19, p. 195]. If the input flow is Poisson, then the regeneration points should be chosen at times when the next service procedure has recently been completed, and the corresponding requirement is leaving the system. If the service time is distributed according to an exponential law, then the regeneration points should be chosen at times that coincide with the moments of receipt of new requests.

After identifying the regeneration points, the events occurring around these points are specified and the probability distribution of the number of events of the incoming Poisson flow or Poisson service flow over time between neighboring regeneration points is determined.

A transition probability matrix is being constructed.

$$\left[P_{ij}\right] = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \dots & p_{0n} \\ p_{10} & p_{11} & p_{12} & \dots & p_{1n} \\ p_{20} & p_{21} & p_{22} & \dots & p_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ p_{n0} & p_{n1} & p_{n2} & \dots & p_{nn} \end{bmatrix}, \qquad (2.20)$$

where p_{ij} – the probability of transition from state *i* to state *j* during the time between adjacent regeneration points. Using matrix (2.20) and the method of subsection 1.2.2, we can construct a system of linear algebraic equations for the state probabilities:

$$p_j = \sum_{i=0}^{n} p_i \cdot p_{ij},\ j = 0,1,2,...,n, \qquad (2.21)$$

which is supplemented by the normalization condition: $\sum_{j=0}^{n} p_j = 1.$

It is advisable to solve system (2.21) using one of the computer mathematics systems, for example Mathlab.

For a single-channel queueing systems of the form M/G/1 with a Poisson input flow characterized by intensity λ and service time distribution function *B*(t), we obtain $p_{ij} = k_{j-i+1},\ j > 0;\ p_{oj} = k_j$ [20, p. 152], that is, the transition probability matrix (2.20) will become as follows:

$$\left[P_{ij}\right] = \begin{bmatrix} k_0 & k_1 & k_2 & k_3 & \dots \\ k_0 & k_1 & k_2 & k_3 & \dots \\ 0 & k_0 & k_1 & k_2 & \dots \\ 0 & 0 & k_0 & k_1 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix},$$

where $k_m = \int_0^{\infty} e^{-\lambda t} \frac{(\lambda t)^m}{m!} dB(t), m \geq 0$ and $k_m = 0,\ m < 0.$

To illustrate the use of the imbedded Markov chains, method, we consider a queueing systems with group selection of requests (similar to the example from Section 2.3.1), but with the condition that the incoming flow of requests is described by the distribution function *A*(*t*), and the process of selecting requests is described by an exponential distribution with intensity μ [21]. As the state of the imbedded Markov chains, we choose the number of requests in the accumulator immediately before the next request arrives. Denote by p_{ij} the probability of transition in time τ between two regeneration points. During time τ a request for a census of the storage contents may be received, the probability of this is $1 - \exp(-\mu\tau)$. Then the process will transition from the state s_i in the state $s_0, i = \overline{0,k}$ The probability of this event is $p_{i0} = \int_0^{\infty} (1 - e^{-\mu t}) dA(t) = 1 - \alpha(\mu)$, where $\alpha(\mu)$– Laplace-Stiehljes transform of A(t) with respect to the parameter μ.

During time τ, the serving device may not select the contents of the storage device, the probability of this is $\exp(-\mu\tau)$. So the process will go from state i to state $i+1, i = \overline{0,k-1}$. The probability of this event is $p_{i,i+1} = \int_0^{\infty} e^{-\mu t} dA(t) = \alpha(\mu)$.

If the process is in state *k*, then with probability $p_{kk} = \alpha(\mu)$ it will remain in this same state, meaning the request that recently arrived will be lost. The transition probability matrix looks like this:

$$\left\| P_{ij} \right\| = \begin{Vmatrix} 1-\alpha(\mu) & \alpha(\mu) & 0 & ... & 0 \\ 1-\alpha(\mu) & 0 & \alpha(\mu) & ... & 0 \\ 1-\alpha(\mu) & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 1-\alpha(\mu) & 0 & 0 & ... & \alpha(\mu) \end{Vmatrix}.$$

All states of the matrix $\left\| P_{ij} \right\|$ – non-periodic, non-zero and indecomposable. Therefore, for the investigated chain there is a stationary state characterized by the probabilities p_i, $i = \overline{0,k}$ that there was $i = \overline{0,k}$ requests in the storage buffer before the regeneration moment. For these probabilities, the following system of equations is valid:

$$p_j = \sum_{i=0}^{k} p_i \cdot p_{ij}, j = 0,1,2,...,k, \qquad (2.22)$$

and the normalization condition is

$$\sum_{j=0}^{k} p_j = 1. \qquad (2.23)$$

Solving the system of equations (2.22) simultaneously with the normalization condition, we obtain:

$$p_j = [1-\alpha(\mu)][\alpha(\mu)]^j, 0 \le j < k; \; p_k = [\alpha(\mu)]^k.$$

Average number of requests in the drive –

$$L_q = \sum_{j=1}^{k} j p_j = \alpha(\mu)\{1-[\alpha(\mu)]^k\}/[1-\alpha(\mu)].$$

An important characteristic of the operation of a storage-based queueing system is the probability of service failure.

Figure 2.7 shows graphs of the dependence of the probability of failure on service p_k as a function of the storage volume *k* and the reading intensity μ for three types of input streams with their average intensity equal to eight:

a – hyperexponential input flow with distribution function $A(t) = 1-[0,3\exp(-2,79t) + 0,7\exp(-40t)]$ and the coefficient of variation $\nu = 1,996$;

b – Poisson input flow with distribution function $A(t) = 1 - \exp(-8t)$ coefficient of variation $\nu = 1$;

c – Erlang input stream with distribution function $A(t) = 1 - \sum_{i=1}^{3} \frac{(24t)^{i-1}}{(i-1)!} e^{-24t}$ and the coefficient of variation $\nu = 0{,}5773$.

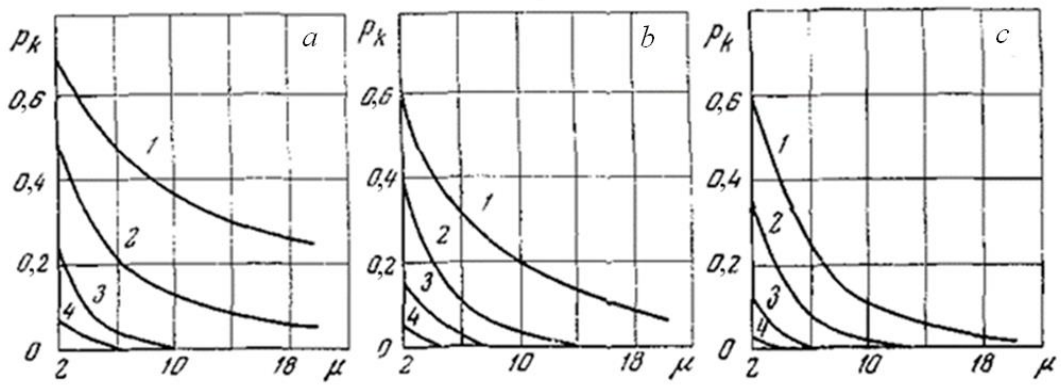


Figure 2.7 – Probability of service failure

It is clear that the coefficient of variation significantly affects the characteristics of the system.

The characteristics obtained by the method of imbedded Markov chains are valid only for regeneration moments and do not fully describe the features of the queueing systems. Only the characteristics for an arbitrary moment of time, connected by certain relations with the characteristics for regeneration points, contain complete information. We will present these relations.

Let $t_n (n \geq 1)$ – moments of receipt of requests, $z(t)$ – the number of requests in the storage at time t, $\lambda_p(i)$ – the intensity of the subsequence of those moments of time for which the number of requests in the storage is equal to *i*. Then

$$p_i = p\{z(t_n - 0) = i\} = \lambda_p(i) / \lambda, \qquad (2.24)$$

where λ is the average intensity of the incoming flow of requests.

Let us denote by $N_i(t)$ number of jumps number of requests in the type storage $i \to i+1$ for the time interval (0, i), and after $M_i(t)$ – number of transitions of type $i+1 \to 0$ (i<k) for the same time interval (0, i). The following equality will be true:

$$\left| N_i(t) - \sum_{j=i}^{k-1} M_j(t) \right| \leq 1. \qquad (2.25)$$

Average (E) number of transitions $i \to i+1$ will be at time t $E[N_i(t)] = \lambda_p(t)t$. Using (2.24), we obtain $E[N_i(t)] = \lambda p_i t$. Average number of type transitions $i+1 \to 0$ for time t

$$E[M_i(t)] = \int_0^t \mu x p_{\xi_{j+1}(t)} dx, \qquad (2.26)$$

Where $p_{\xi_{j+1}(t)}$ – distribution of a random fraction of time in (0, t) during which the storage contained $i+1$ requests. Since $E[\xi_{j+1}(t)] = \int_0^t x p_{\xi_{j+1}(x)} dx = \bar{p}_{j+1} t$, where $\bar{p}_{j+1}$ – probability of being in the storage $j+1$ requests at any point in time, then from (2.26) we obtain: $E[M_i(t)] = \mu \bar{p}_{j+1} t$. From inequality (2.25) it follows: $\lim_{t\to\infty} E[N_i(t)]/t = \lim_{t\to\infty} \sum_{j=i}^{k-1} E[M_j(t)]/t$ or $\sum_{j=1}^{k-1} \bar{p}_{j+1} = (\lambda/\mu) p_i$, $0 \le i < k$.

From here we finally obtain the combination of probabilities of the state of the queueing system at an arbitrary moment in time ($\bar{p}_i$) with state probabilities at regeneration points (p_i):

$$\bar{p}_0 = 1 - \frac{\lambda}{\mu} p_0; \quad \bar{p}_i = \frac{\lambda}{\mu}(p_{i-1} - p_i), 0 < i < k; \; \bar{p}_k = \frac{\lambda}{\mu} p_{k-1}.$$

The expressions for the characteristics of the queueing systems are adjusted accordingly. For example, the probability of service failure is $\bar{P}_{fail} = \frac{\lambda}{\mu}[1 - \alpha(\mu)][\alpha(\mu)]^{k-1}$.

2.3.4 Method of including additional variables

The method of including additional variables is one of the most universal in the study of non-Markovian queueing systems, but its application involves the use of the apparatus of integro-differential equations and the solution is not always possible to obtain in quadratures. The method of including additional variables allows the transformation of the original non-Markovian process into a Markovian one by introducing additional variables to describe the state of the system.

Let $n(t)$ – the state of the original system at time t, $p_n(t)$ – the probability of this state. If the input to the system is Markovian or can be reduced to Markovian, then the state of the system is supplemented by the variable u – the time elapsed since the start of servicing the request contained in the device. If, on the contrary, the servicing process is Markovian, then the state of the system $n(t)$ is supplemented by the variable v – the time elapsed since the last request was received. As a result, the original process is transformed into a Markov process.

Let us consider the use of this method on the example of a queuing system with group selection of requests from the storage, in which the input flow is Poisson with intensity λ, the storage volume is k requests, and the time intervals between adjacent requests to the storage have the distribution $B(t)$. The request is made by the serving device after processing the previous group of requests. Let r(u) denote the instantaneous intensity of reading requests from the storage, which is defined as $r(u) = B'(u)[1 - B(u)]^{-1}$, through μ – the average intensity of reading requests, through $\beta(s) = \int_0^\infty e^{-st} dB(t)$ – Laplace-Stiehljes transform of $B(t)$.

Let [δ] be the state when there are no requests in the system, [n, u] be the state when there are n requests in the storage, the service channel processes the next group of requests, and the time elapsed since the previous access to the storage is equal to u. Denote by $p_\delta(t)$ probability of state [δ], through $p_n(t)$ – the probability that the storage contains n requests, and through $p_n(t,u)$ – the probability density for u that the storage contains n requests, and the time elapsed since the previous access to the storage (service time) is equal to u_0, $u < u_0 < u + \Delta$. Probabilities $p_n(t)$ and $p_n(t,u)$ connected by a ratio $p_n(t) = \int_0^\infty p_n(t,u)du$.

Let us consider the possible transitions of the system from one state to another during the interval $(t, t+\Delta)$. Let's fix the moment in time and determine the probability $p_0(t+\Delta, u+\Delta)\Delta$ – that at time $t+\Delta$ the system will be in the state $[0, u+\Delta u]$. This can happen only under one condition: the system at time t was in the state [0, u], and at time Δ with probability $1 - r(u)\Delta$ the service of a group of requests that were already being processed has not ended (the probability of this by definition $p_n(t,u)\Delta$), and with probability 1 – λΔ no new request has been received by the system. The validity of the second statement follows from the fact that if a request is received during time Δ, it will be immediately selected for service and the system will go from state [δ] at time t to state [0, Δ] at $t+\Delta$, although this contradicts the original statement. Thus,

$$p_0(t+\Delta, u+\Delta)\Delta = p_0(t,u)\Delta\{[1 - r(u)\Delta][1 - \lambda\Delta]\} + o(\Delta). \quad (2.27)$$

Let us determine the probability that at time $t+\Delta$ the system will be in state [n, u + Δ]. This can be achieved in two ways:

– at time t the system was in state [n, u], during time Δ no new request was received and the servicing of the current group of requests was not completed;

At time *t*, the system was in state $[n-1,u]$, and in time Δ with probability λΔ a request was received by the system. So,

$$p_n(t+\Delta,u+\Delta)\Delta = p_n(t,u)\Delta\{[1-r(u)\Delta][1-\lambda\Delta]\} + p_{n-1}(t,u)\Delta\lambda\Delta + o(\Delta). \quad (2.28)$$

Let us define the boundary conditions at the point $u=0$. To do this, we will again fix the time *t* and derive the probability that at the time $t+\Delta$ the system will be in the state $[n,\Delta]$, i.e., from the moment of starting the service of the next group of requests, a time not exceeding Δ has passed. First, we will derive the equation for the state $[0,\Delta]$. The system can go to the state $[0,\Delta]$ in time Δ in two ways:

– at time *t* the system was in state [δ] and during time Δ with probability λΔ a new request was received, which was immediately selected for service;

– at time *t* the system was in state $[n, u]$ and with probability r(u)Δ the servicing of the previous group of requests ended, as a result of which the storsdge was reset. The probability of this event will consist of the probabilities for all possible values of *n* and all possible values of *u*. So,

$$p_0(t+\Delta,\Delta)\Delta = p_\delta(t)\lambda\Delta + \sum_{n=1}^{k}\int_0^\infty p_n(t,u)r(u)du\Delta. \quad (2.29)$$

Since no more than one request can be received by the system during time Δ and the storage device is reset when it is accessed, then

$$p_k(t,\Delta) \equiv 0. \quad (2.30)$$

Let us derive the probability that at time $t+\Delta$ the system will be in state [δ]. This can be achieved in two ways:

– at time *t* the system was in state [δ] and with probability $1-\lambda\Delta$ a new request will not arrive;

– the system at time *t* was in the state [0, u] and in time Δ with probability r(u)Δ the servicing of the next group of requests ended. So,

$$p_\delta(t+\Delta) = p_\delta(t)[1-\lambda\Delta] + \int_0^\infty p_0(t,u)r(u)du\Delta. \quad (2.31)$$

Let us open the brackets in expressions (2.27) and (2.28), transfer $p_n(t,u)$ to the left side, here we add $p_n(t,u+\Delta) - p_n(t,u+\Delta)$, divide everything by Δ and go to the limit at $\Delta \to 0$. We obtain a system of equations

$$\begin{aligned} &\frac{\partial}{\partial t}p_0(t,u) + \frac{\partial}{\partial u}p_0(t,u) = -[\lambda + r(u)]p_0(t,u), \\ &\frac{\partial}{\partial t}p_n(t,u) + \frac{\partial}{\partial u}p_n(t,u) = -[\lambda + r(u)]p_n(t,u) + \lambda p_{n-1}(t,u). \end{aligned} \quad (2.32)$$

Similarly, from expressions (2.29) – (2.31) we derive the boundary conditions

$$
\begin{aligned}
& p_0(t,0)\Delta = \lambda p_\delta(t) + \sum_{n=1}^{k}\int_0^\infty p_n(t,u)r(u)du \\
& \frac{\partial}{\partial t} p_\delta(t) = -\lambda p_\delta(t) + \int_0^\infty p_0(t,u)r(u)du, \\
& p_n(t,0) \equiv 0,\ 0 < n \le k.
\end{aligned}
\tag{2.33}
$$

As initial conditions we choose $p_\delta(0) = 1.$ The joint system of equations (2.32), (2.33) can be solved numerically on a computer, for example, by the grid method, using appropriate software.

Let us consider the stationary regime. In this case, $p_n(t,u) \to p_n(u),\ \dfrac{\partial p_n(u)}{\partial t} = 0.$

As a result, we obtain the following system of equations:

$$
\begin{aligned}
& \frac{\partial}{\partial u} p_0(u) = -[\lambda + r(u)]p_0(u), \\
& \frac{\partial}{\partial u} p_n(u) = -[\lambda + r(u)]p_n(u) + \lambda p_{n-1}(u), 1 \le n \le k
\end{aligned}
\tag{2.34}
$$

and boundary conditions:

$$
\begin{aligned}
& -\lambda p_\delta + \int_0^\infty p_0(u)r(u)du = 0, \\
& p_0(0) = \lambda p_\delta + \sum_{n=1}^{k}\int_0^\infty p_n(u)r(u)du.
\end{aligned}
\tag{2.35}
$$

Solving the system of equations (2.34) step by step, we obtain:

$$
\begin{aligned}
& p_0(u) = [1 - B(u)]p_0(0)e^{-\lambda u}, \\
& p_1(u) = \lambda u[1 - B(u)]p_0(0)e^{-\lambda u}, \\
& p_n(u) = \frac{(\lambda u)^n}{n!}[1 - B(u)]p_0(0)e^{-\lambda u},\ 0 \le n \le k.
\end{aligned}
\tag{2.36}
$$

Substituting expression (2.36) into the first equation of system (2.35), we find $p_\delta = \beta(\lambda)p_0(0)/\lambda.$

Therefore, $p_n = \int_0^\infty p_n(u)du = \dfrac{p_0(0)}{\lambda} - \dfrac{(-1)^n}{n!}\lambda^{n-1}\beta^{(n)}(\lambda),$ where $\beta^{(n)}(\lambda)$ – the value of the *n*-th derivative of $\beta(s)$ at s = λ.

Using the normalization condition $p_\delta + \sum_{n=0}^{k} p_n = 1$, we will get

$$p_0(0) = \lambda \cdot \{k - \sum_{n=1}^{k} \frac{(-1)^n}{n!} \beta^{(n)}(\lambda)\}^{-1}.$$

For the Erlang distribution of order m $\beta^{(n)}(\lambda) = (-1)^n \frac{(m+n-1)!}{(m-1)!} \frac{\mu^m}{(\mu+\lambda)^{m+n}}$.

For the hyperexponential distribution $\beta^{(n)}(\lambda) = (-1)^n n! \sum_{i=1}^{m} \frac{b_i \mu_i}{(\mu_i + \lambda)^{n+1}}$.

Basic characteristics of the system under study:

– probability of service failure – $P_{fail} = p_k$;

– average number of requests in the buffer – $L_q = \sum_{i=1}^{k} i p_i$;

– average time requests stay in the buffer – $\tau_q = \sum_{i=1}^{k} \frac{p_i}{\mu_i}$.

Thus, to use the method of including additional variables, it is required that one of the queueing system event flows be Poisson or reduce to it, and the distribution function of the second one has a finite Laplace–Steltjes transform.

2.3.5 Little's formulas

For queuing systems operating in a stationary mode, the most suitable are Little's formulas, which relate the more important characteristics of the queueing system.

Let the queueing system, which receives a flow of requests with intensity λ, operate in a stationary mode for a sufficiently large time interval *T*. Then, during this time, the system will receive on average λT requests. Each request is in the queue for an average time of τ_q; then the average time W, which will be in the queue all λT requests, is calculated by the formula:

$$W = \lambda \tau_q T. \quad (2.37)$$

On the other hand, if L_q – the average number of requests waiting for service, then the total average time W, which they will be in the queue for time T, is calculated by the formula:

$$W = L_q T. \quad (2.38)$$

Comparing the right-hand sides of expressions (2.37) and (2.38) and reducing T, we obtain

$$L_q = \lambda \tau_q . \tag{2.39}$$

Each request is in the system for an average time τ_s; then the average time V, which are all in the system λT requests, is calculated by the formula:

$$V = \lambda \tau_s T . \tag{2.40}$$

On the other hand, if L_s – the average number of requests in the system, then the average time τ_s, which they will be in the system for time *T*, is calculated by the formula:

$$V = L_s T . \tag{2.41}$$

Comparing the right-hand sides of expressions (2.40) and (2.41) and reducing T, we obtain:

$$L_s = \lambda \tau_s . \tag{2.42}$$

Formulas (2.39) and (2.42) were named Little's formulas in honor of the researcher J. D. K. Little, who substantiated them [3]. These formulas play an important role in the theory of queuing, since under certain conditions they make it possible to eliminate complex transformations when deriving the characteristics of the system.

3 TYPICAL QUEUING SYSTEMS

3.1 Queuing system with failures

The queuing system with rejections consists of *n* service channels, to which the simplest flow of requests with intensity λ arrives, the service time is distributed according to an exponential law with parameter μ, that is, the service flow is also the simplest with parameter μ. There is no queue, that is, a request that arrived when all channels are busy is denied service. This system is one of the simplest and the theory of mass service systems is built on it. It was first considered by the prominent researcher A. K. Erlang. A typical example of the application of the queuing system with rejections is automatic telephone exchanges. Here, the request that arrived in the system is a client's appeal to the telephone exchange. If the required communication line is already busy with a conversation, the subscriber is denied. The automatic telephone exchange gives frequent beeps and the request is lost.

Let us examine it. Let us denote by S_i the state of the system when it contains $i, i = 1,2,...,n$. The state and transition graph for this system at $n = 5$ is shown in Figure 3.1.

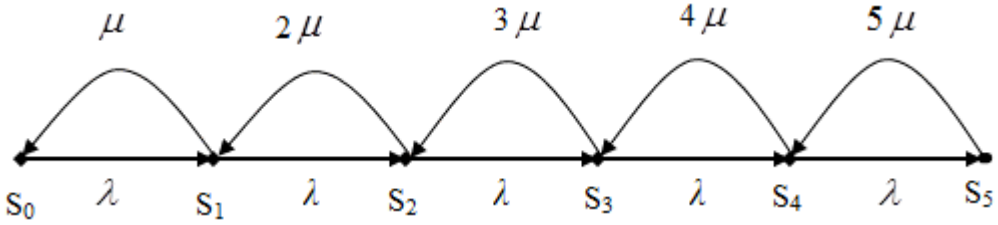


Figure 3.1 – State and transition graph of a queuing system with failures

Let us denote by $p_i(t)$ probability of state S_i. Using the method of subsection 1.2.3, we will compose a system of differential equations for the states $S_i,\ i = 0,1,2,...,n$:

$$\begin{aligned} p_0'(t) &= -\lambda p_0(t) + \mu p_1(t), \\ p_1'(t) &= -(\lambda + \mu) p_1(t) + \lambda p_0(t) + 2\mu p_2(t), \\ p_i'(t) &= -(\lambda + i\mu) p_i(t) + \lambda p_{i-1}(t) + (i+1)\mu p_{i+1}(t), 1 \le i < n, \\ p_n'(t) &= -n\mu p_n(t) + \lambda p_{n-1}(t). \end{aligned} \tag{3.1}$$

Initial conditions –

$$p_0(0) = 1,\ p_i(0) = 0,\ 1 \le i < n. \tag{3.2}$$

If *n* is small, then using the technology of solving systems of linear differential equations [16] for specific values of λ and μ, one can find an analytical

solution of this system. For example, for $n=2;\ \lambda=1;\ \mu=0{,}55$ the solution to system (3.1) looks like this:

$$p_0(t)=0{,}645\cdot e^{-1{,}03t}+0{,}12\cdot e^{-2{,}61t}+0{,}224,$$
$$p_1(t)=-0{,}040\cdot e^{-1{,}03t}-0{,}366\cdot e^{-2{,}61t}+0{,}407,$$
$$p_2(t)=-0{,}605\cdot e^{-1{,}03t}+0{,}25\cdot e^{-2{,}61t}+0{,}370.$$

Graphs of function dependencies $p_0(t), p_1(t), p_2(t)$ over time is shown in Figure 3.2.

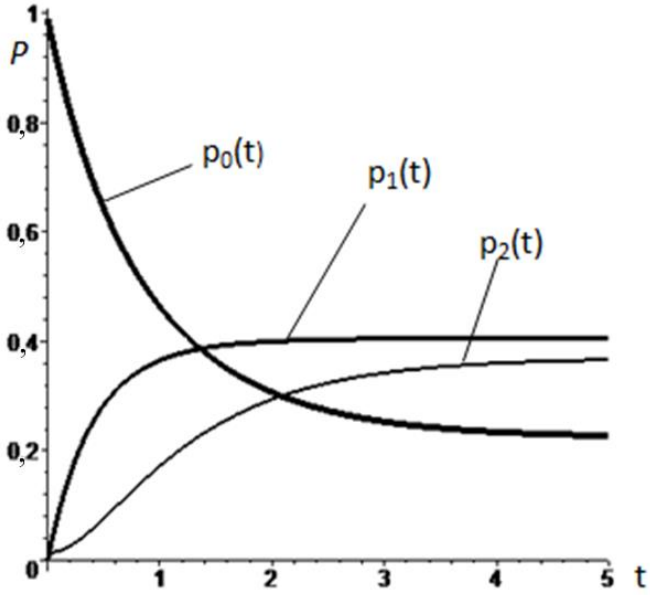


Figure 3.2 – Dependences of the probabilities of the states of a system with failures on time

It is clear that in the case of unlimited growth of t the behavior of the functions $p_i(t)$ ceases to depend on time and the system enters a stationary mode.

That is $\lim\limits_{t\to\infty} p_i(t)=p_i$, $i=0,\ 1,\ 2$, where p_i – constants. From the analysis of expressions for functions $p_i(t)$ it follows that when $t\to\infty$ they are equal to the corresponding constants on the right-hand side, i.e. $p_0=0{,}224;\ p_1=0{,}406;\ p_2=0{,}370$.

The probabilities of the states of the system in the stationary regime can be determined by a simpler method. Given that $\lim\limits_{t\to\infty} p_i'(t)=\lim\limits_{t\to\infty} p_i'=0$, the system of equations (3.1) is transformed as follows:

$$\begin{aligned}
&-\lambda p_0+\mu p_1=0,\\
&-(\lambda+\mu)p_1+\lambda p_0+2\mu p_2=0,\\
&-(\lambda+i\mu)p_i+\lambda p_{i-1}+(i+1)\mu p_{i+1}=0,\ 1\le i<n,\\
&-n\mu p_n+\lambda p_{n-1}=0.
\end{aligned} \qquad (3.3)$$

The normalization condition is

$$\sum_{i=0}^{n} p_i = 1. \tag{3.4}$$

Divide each equation of system (3.3) by μ and introduce the parameter $\rho = \lambda / \mu$ – system load on one channel. The system of equations (3.4) will become as follows:

$$\begin{aligned} &-\rho p_0 + p_1 = 0, \\ &-(\rho+1) p_1 + \rho p_0 + 2 p_2 = 0, \\ &-(\rho+i) p_i + \rho p_{i-1} + (i+1) p_{i+1} = 0, 1 \le i < n, \\ &-n p_n + \rho p_{n-1} = 0. \end{aligned} \tag{3.5}$$

Sequentially solving system (3.5) with respect to p_0, we obtain: $p_i = \frac{\rho^i}{i!} p_0$.

Substituting this expression into (3.4), we define $p_0 = \left(1 + \sum_{i=1}^{n} \frac{\rho^i}{i!}\right)^{-1}$.

Finally, the probability that i requests are served, i.e. the system is in state S_i, is given by the expression

$$p_i = \frac{\rho^i}{i!} \left(\sum_{k=0}^{n} \frac{\rho^k}{k!} \right)^{-1}, 0 \le i \le n. \tag{3.6}$$

This expression was called Erlang's formula. It was proved that this formula is valid for any continuous law of service time distribution on the channel [14].

The basic characteristics of the studied queuing system with failures are as follows:

– probability of service failure $P_{fail} = p_n = \frac{\rho^n}{n!} \left(\sum_{k=0}^{n} \frac{\rho^k}{k!} \right)^{-1}$. This expression means that this request will be denied service if all lines are busy;

– absolute throughput is equal to the average number of requests that the queuing system can serve per unit of time, that is, it is the fraction of the incoming flow of requests that enters the system and is served by it:

$$\mathrm{A} = \lambda Q = \lambda \left[1 - \frac{\rho^n}{n!} \left(\sum_{k=0}^{n} \frac{\rho^k}{k!} \right)^{-1} \right]; \tag{3.7}$$

– the average number of requests in the system, and accordingly the average number of channels occupied by servicing requests, is calculated as the mathematical expectation:

$$L_s = \sum_{i=1}^{n} i p_i = \sum_{i=1}^{n} i \frac{\rho^i}{i!} \left(\sum_{k=0}^{n} \frac{\rho^k}{k!} \right)^{-1}; \quad (3.8)$$

– system load factor $K_z = L_s / n = \sum_{i=1}^{n} i \frac{\rho^i}{ni!} \left(\sum_{k=0}^{n} \frac{\rho^k}{k!} \right)^{-1}$.

It is advisable to know the dependence of the basic characteristics of the system on n and the load factor on all channels $\alpha = \lambda/(n\mu) = \rho/n$. From here $\rho = n\alpha$. Then the basic characteristics will be as follows:

– probability of service failure $P_{fail} = p_n = \frac{(n\alpha)^n}{n!} \left(\sum_{k=0}^{n} \frac{(n\alpha)^k}{k!} \right)^{-1}$;

– system load factor $K_z = L_s / n = \sum_{i=1}^{n} i \frac{(n\alpha)^i}{ni!} \left(\sum_{k=0}^{n} \frac{(n\alpha)^k}{k!} \right)^{-1}$.

Figure 3.3 shows a graph of the dependence of the probability of service failure on the number of service channels at different ρ.

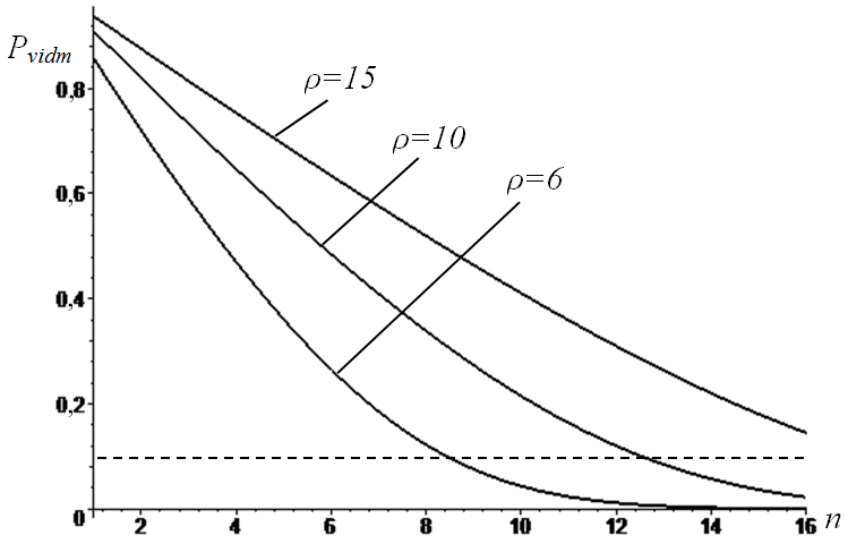


Figure 3.3 – Dependence of the probability of failure on service for a queuing system with failures

Figure 3.4 shows a graph of the system load factor versus the number of service channels at different ρ.

According to these graphs, it is possible to choose the number of service channels at different load factors of one channel ρ depending on the technical tasks for system design. For example, if it is necessary to choose the number of service channels in case of failure probability 0.1 at $\rho = 10$, then from Figure 3.3 it follows that there must be at least 13 units.

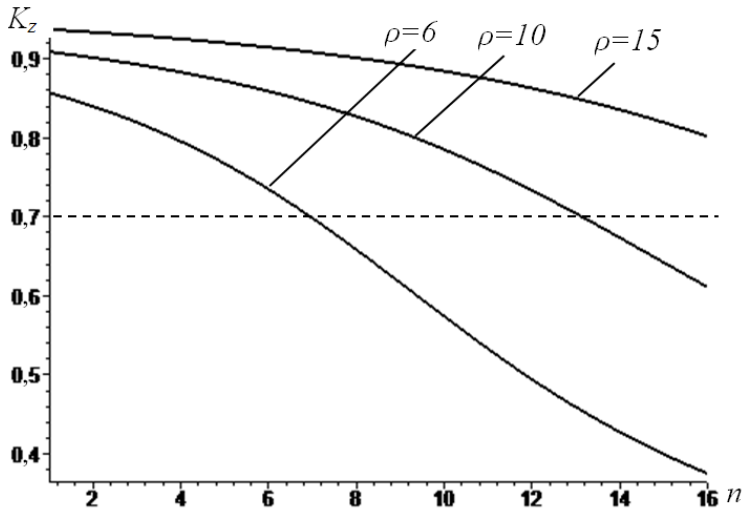


Figure 3.4 – Dependence of the system load factor for a queuing system with failures

If the number of channels is selected depending on the system load factor, then you need to use the graphs shown in Figure 3.4. For example, if you need to select the number of service channels in the case of a system load factor $K_z = 0{,}7$ at $\rho = 10$, then from Figure 3.4 it follows that there should be no more than 14 units.

3.2 System with Poisson input flow and exponential service time (M/M/1)

This is one of the most studied queuing system [4], for which almost all characteristics have been obtained in finite expressions. Let the system be supplied with a Poisson input flow ($p_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$) with intensity λ, the service time is distributed according to the exponential law ($B(t) = 1 - e^{-\mu t}$) with intensity μ. The queue is not bounded. The state and transition graph for this system is shown in Figure 3.5.

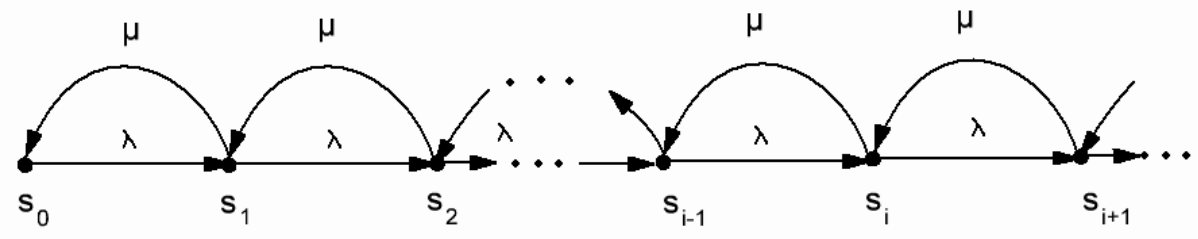


Figure 3.5 – State and transition graph of the M/M/1 queuing system

It follows from the graph that the random process in the studied queuing system belongs to the so-called birth and death processes, which describe changes in the bacterial population.

Let S_i denote the state of the system when it contains i requests $(i = 0, 1, 2, 3, ...)$, the probability of this $p_i(t)$. Using the method of subsection 1.2.3, we will compose a system of differential equations for the states s_i, $i = 0,1,2,...$:

$$
\begin{aligned}
&p_0'(t) = -\lambda p_0(t) + \mu p_1(t),\\
&p_1'(t) = -(\lambda+\mu)\, p_1(t) + \lambda p_0(t) + \mu p_2(t),\\
&p_2'(t) = -(\lambda+\mu)\, p_2(t) + \lambda p_1(t) + \mu p_3(t),\\
&.....\\
&.....\\
&p_i'(t) = -(\lambda+\mu)\, p_i(t) + \lambda p_{i-1}(t) + \mu p_{i+1}(t),\ i \geq 1,\\
&.....
\end{aligned}
\qquad (3.9)
$$

Initial conditions –

$$p_0(0) = 1,\ p_i(0) = 0, i \geq 1. \qquad (3.10)$$

The solution and investigation of system (3.9) using the Laplace transform and the generating function are given in [15]. But the finite expressions are very complicated and inconvenient for practical application. From this work it is known that if $\mu > \lambda$, then for the system there is a stationary regime with a stationary probability distribution of the system states $p_i, i = 0,1,2,...$. Thus, the functioning of the queuing system is decomposed into two regimes: transient and stationary. The stationary regime will be investigated below. The transient regime can be investigated by numerical methods. For this, we rewrite system (3.9) in differences:

$$
\begin{aligned}
&p_0(t+\Delta t) = p_0(t) + \Delta t[-\lambda p_0(t) + \mu p_1(t)],\\
&p_1(t+\Delta t) = p_1(t) + \Delta t[-(\lambda+\mu)\, p_1(t) + \lambda p_0(t) + \mu p_2(t)],\\
&.....\\
&p_i(t+\Delta t) = p_i(t) + \Delta t[-(\lambda+\mu)\, p_i(t) + \lambda p_{i-1}(t) + \mu p_{i+1}(t)],\ i \geq 1,\\
&.....\\
&p_0(0) = 1, p_i(0) = 0, i > 0.
\end{aligned}
\qquad (3.11)
$$

To solve system (3.11), we choose the numerical Euler method for solving systems of differential equations [25]. We denote by p_i^k probability value $p_i(t)$ at the *k*-th step of the calculations. System (3.11) is rewritten in a form ready for iterative calculation:

$$
\begin{aligned}
&p_0^{k+1} = p_0^k + h \cdot (-\lambda p_0^k + \mu p_1^k), k = 0,1,2...,\\
&p_i^{k+1} = p_i^k + h \cdot [-(\lambda+\mu)\, p_i^k + \lambda p_{i-1}^k + \mu p_{i+1}^k], i > 0,\ k = 0,1,2...,\\
&p_0^0 = 1,\ p_i^0 = 0, i \geq 1,
\end{aligned}
\qquad (3.12)
$$

where h is the integration step.

To numerically solve the system of differential equations (3.9) using formulas (3.12), it is necessary to specify specific values of the parameters λ, μ and choose the integration step *h*. Existing software systems cannot provide a solution to a system of equations with an infinite number of equations, so when using formulas (3.12), we will limit ourselves to a certain number of them. It will have to be selected experimentally, and the solution will be approximate.

Figure 3.6 shows the probability graphs of the state of the system M/M/1 for the transition regime at $\lambda = 0{,}6;\ \mu = 1;\ h = 0{,}06$.

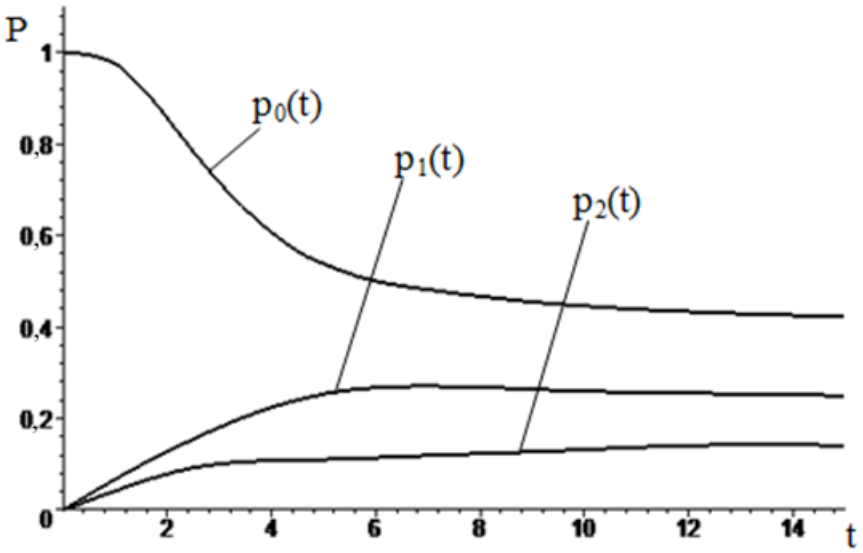


Figure 3.6 – Transition mode of the M/M/1 system

It is clear that from the moment of time $t = 8$ the system enters a steady-state operating mode.

Now let us examine the steady-state mode of operation of the M/M/1 system. In the steady-state mode $p_i(t) \to p_i, p_i' = 0$ and system (3.9) will take the following form:

$$\begin{aligned}
&-\lambda p_0 + \mu p_1 = 0,\\
&-(\lambda+\mu)p_1 + \lambda p_0 + \mu p_2 = 0,\\
&-(\lambda+\mu)p_2 + \lambda p_1 + \mu p_3 = 0,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&-(\lambda+\mu)p_i + \lambda p_{i-1} + \mu p_{i+1} = 0,\ i \geq 1,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned} \qquad (3.13)$$

Normalization condition

$$\sum_{i=0}^{\infty} p_i = 1. \qquad (3.14)$$

Divide each equation of system (3.13) by μ and introduce the parameter $\rho = \lambda/\mu$ system load. The system of equations (3.13) can be rewritten as follows:

$$\begin{aligned} &-\rho p_0 + p_1 = 0, \\ &-(\rho+1)p_1 + \rho p_0 + p_2 = 0, \\ &-(\rho+1)p_2 + \rho p_1 + p_3 = 0, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &-(\rho+1)p_i + \rho p_{i-1} + p_{i+1} = 0,\ i \geq 1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{aligned} \tag{3.15}$$

From the first equation of system (3.15) it follows that $p_1 = \rho p_0$. Let's substitute this expression into the second equation: $-(\rho+1)\rho p_0 + \rho p_0 + p_2 = 0$. After bringing similar terms together, we obtain: $p_2 = \rho^2 p_0$. Continuing the calculation, we get: $p_i = \rho^i p_0,\ i \geq 1$. Let's substitute the obtained expressions for p_i into the normalization condition:

$$p_0 + \sum_{i=1}^{\infty} \rho^i p_0 = 1 \rightarrow p_0 \left(1 + \sum_{i=1}^{\infty} \rho^i\right) = 1.$$

If $\rho < 1$, then the sum $\sum_{i=1}^{\infty} \rho^i$ is a geometric progression that converges to $\frac{\rho}{1-\rho}$.

Then $1 + \frac{\rho}{1-\rho} = \frac{1}{1-\rho}$ from where

$$p_0 = 1-\rho,\ p_i = (1-\rho)\rho^i. \tag{3.16}$$

Let's define the basic characteristics of the system's functioning. To do this, we will need a generating function:

$$P(z) = \sum_{i=0}^{\infty} p_i z^i = (1-\rho)\sum_{i=0}^{\infty} (\rho z)^i = \frac{(1-\rho)}{1-\rho z},\ (\rho z < 1). \tag{3.17}$$

The average number of requests in the system is calculated as the mathematical expectation

$$\begin{aligned} L_s &= \sum_{i=1}^{\infty} i p_i = \sum_{i=1}^{\infty} i\rho^i = \sum_{i=0}^{\infty} \frac{\partial}{\partial z}(\rho z)^i \Big|_{z=1} = \frac{\partial}{\partial z} P(z)\Big|_{z=1} = \\ &= (1-\rho)\frac{\partial}{\partial z}\frac{1}{1-\rho z}\Big|_{z=1} = (1-\rho)\frac{\rho}{(1-\rho z)^2}\Big|_{z=1}. \end{aligned} \tag{3.18}$$

Finally we get:

$$L_s = P'(1) = \frac{\rho}{1-\rho}. \tag{3.19}$$

Average number of requests in the queue –

$$L_q = \sum_{i=1}^{\infty}(i-1)p_i = \sum_{i=1}^{\infty} ip_i - \sum_{i=1}^{\infty} p_i = \frac{\rho}{1-\rho} - (1-p_0) = \frac{\rho}{1-\rho} - (1-1+\rho).$$

Finally we get:

$$L_q = \frac{\rho^2}{1-\rho}. \tag{3.20}$$

Let's determine the average time requests spend in the system τ_s. To do this, we apply Little's formula (2.42) – $L_s = \lambda\tau_s$ from where

$$\tau_s = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu(1-\rho)}. \tag{3.21}$$

Similarly, using formula (2.39), we determine the average time spent in the queue: $\tau_q = \frac{L_q}{\lambda} = \frac{\rho^2}{\lambda(1-\rho)}$. After the transformation we get:

$$\tau_q = \frac{\rho}{\mu(1-\rho)}. \tag{3.22}$$

Figure 3.7 shows the dependence of the average time requests spend in the system on the system load ρ.

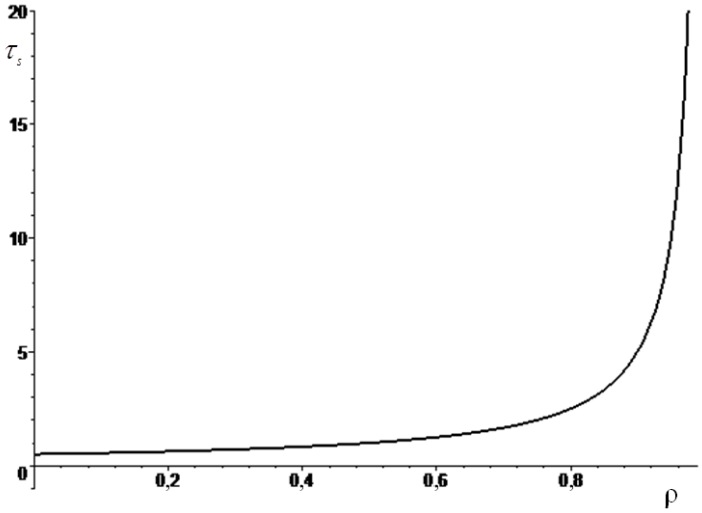


Figure 3.7 – Dependence of the average time of requests in the system depending on the system load ρ

As ρ approaches unity, the time spent by requests in the system increases sharply and approaches infinity.

For several queuing systems with different service disciplines, the average dwell time in the system, according to Little's formula, will be the same. But for queuing systems with different service disciplines, the variance of the dwell time in the system will be different. Let us define the variance of the dwell time of requests in the system M/M/1.

To calculate the variance, we use formula (1.9), which will take the following form:

$$D_s = \sum_{i=1}^{\infty} i^2 p_i - (L_s)^2 , \tag{3.23}$$

Where $\sum_{i=0}^{\infty} i^2 p_i$ the second initial moment.

From (3.17) it follows:

$$P'(z) = \sum_{i=0}^{\infty} p_i \frac{\partial}{\partial z} z^i = \sum_{i=0}^{\infty} p_i i z^{i-1} . \tag{3.24}$$

Multiply both parts of expression (3.24) by z:

$$zP'(z) = \sum_{i=0}^{\infty} p_i i z^i . \tag{3.25}$$

Let us find the derivative with respect to z of (3.25):

$$P'(z) + zP''(z) = \sum_{i=0}^{\infty} p_i i^2 z^{i-1} . \tag{3.26}$$

From here

$$\sum_{i=0}^{\infty} i^2 p_i = \sum_{i=0}^{\infty} p_i i^2 z^{i-1} \Big|_{z=1} = P'(1) + P''(1) . \tag{3.27}$$

Let's find $P''(z)$. From (3.18) it follows:

$$P''(z) = \rho(1-\rho)\left[\frac{1}{(1-\rho z)^2}\right]' = \frac{\rho(1-\rho)\left[2(1-\rho z)\rho\right]}{(1-\rho z)^4} . \tag{3.28}$$

From (3.19) and (3.28) it follows:

$$P'(1) + P''(1) = \frac{\rho}{1-\rho} + \frac{2\rho^2(1-\rho)^2}{(1-\rho)^2} = \frac{\rho+\rho^2}{(1-\rho)^2} . \tag{3.29}$$

Substituting (3.29) into (3.27), and the expression found for $\sum_{i=0}^{\infty} i^2 p_i$ and $(L_s)^2$ in (3.23) we finally obtain:

$$D_s = \frac{\rho}{(1-\rho)^2} . \tag{3.30}$$

From expression (3.30) it follows that with increasing ρ, the variance of the query dwell time in the M/M/1 system also increases, and, when the system load approaches unity, it tends to infinity.

The distribution function of the time spent by requests in the queue is described by the expression $W(t) = 1 - \rho e^{-(\mu-\lambda)t}$ [4], density distribution

$w(t) = \rho(\mu - \lambda)e^{-(\mu-\lambda)t}$. Let us determine the average time requests spend in the queue using formula (1.14):

$$\tau_q = \int_0^\infty t\rho(\mu-\lambda)e^{-(\mu-\lambda)t}dt = \rho(\mu-\lambda)\int_0^\infty \frac{tde^{-(\mu-\lambda)t}}{-(\mu-\lambda)} = -\rho\left[te^{-(\mu-\lambda)t}\Big|_0^\infty - \int_0^\infty e^{-(\mu-\lambda)t}dt\right] =$$

$$= -\rho\left[\frac{e^{-(\mu-\lambda)t}\Big|_0^\infty}{-(\mu-\lambda)}\right] = \frac{\rho}{\mu-\lambda}.$$

When integrating the expression, the method of integration by parts was used. If we take μ out of the brackets, we finally obtain:

$$\tau_q = \frac{\rho}{\mu(1-\rho)}. \qquad (3.31)$$

Expression (3.31), obtained using the classical scheme for calculating the mathematical expectation, completely coincides with expression (3.22), obtained using Little's formula, which confirms its validity.

3.3 Single-line system with an input flow of requests with hyperexponential distribution

This study was carried out by author under the supervision of the remarkable scientist, professor of the Kharkov Aviation Institute, Vyacheslav Alekseevich Popov.

Let the input flow be described by a second-order hyperexponential distribution with distribution function $A(t) = 1 - \sum_{i=1}^{2} a_i e^{-\lambda_i t}$, $\sum_{i=1}^{2} a_i = 1$ and medium intensity $\Lambda = \left(\sum_{i=1}^{2} \frac{a_i}{\lambda_i}\right)$. Service time distribution function $B(t)$, its Laplace-Stieltjes transform is $\beta(s)$, service intensity $r(u) = B'(u)[1 - B(u)]^{-1}$ and the average service intensity is equal to μ. It is assumed that the function $B(t)$ has a derivative.

The process of receipt of requests in the system can be reduced to Markov by introducing a set of states that differ in the number of requirements in the system and the number $i, i = 1, 2$, corresponding to the intensity λ_i input flow at this state (we'll call this number the stage of some fictitious control system). With the arrival of the next request, the system transitions to one of the states corresponding to the changed number of demands in the queueing system, and the stage of the new state is randomly selected by the control system such that the *i*-th stage can be selected with probability $a_i, i = 1, 2$.

The time to complete the *i*-th stage is distributed according to an exponential law with the parameter λ_i. This procedure allows us to define a linear Markov process that describes the functioning of a given system in phase space $\{[0,i],[n,i,u],n\geq 1,i=1,2;u\geq 0\}$. Here is the state $[0,i]$ corresponds to a completely free system when the stage of the regulatory system is equal to *i*, and $[n,i,u]$ corresponds to the state when there are *n* requests in the system, the stage of the control system is equal to *i* and the time during which the next demand is being serviced is equal to *u*.

Let us denote by $p_0^i(t)$ probability of a state $[0,i]$ at time *t*, and after $p_n^i(t,u)$ – probability density of the state $[n,i,u]$. By considering possible changes in the system over an infinitely small period of time and using the methodology of paragraph 2.3.4, one can obtain a system of integro-differential equations describing the functioning of this system. If $\mu>\Lambda$, then for this system there is a stationary regime [17], characterized by probabilities p_0^i and stationary probability densities $p_n^i(u)$. For them, a system of equations can be composed that describes the steady state of the system under consideration:

$$\lambda_i p_0^i=\int_0^\infty p_1^i(u)r(u)du, i=1,2, \tag{3.32}$$

$$\begin{aligned}&\frac{d}{du}p_1^i(u)=-[\lambda+r(u)]p_1^i(u),\ i=1,\ 2,\\&\frac{d}{du}p_n^i(u)=-[\lambda_i+r(u)]p_n^i(u)+a_i\sum_{j=1}^{2}\lambda_i p_{n-1}^j(u)\ ,\ i=1,\ 2,\ n>1\end{aligned} \tag{3.33}$$

and boundary conditions

$$\begin{aligned}&p_1^i(0)=\int_0^\infty p_2^i(u)r(u)du+a_i\sum_{j=1}^{2}\lambda_j p_0^j,\ i=1,2,\\&p_n^i(0)=\int_0^\infty p_{n+1}^i(u)r(u)du,\ \ n>1,\ i=1,2.\end{aligned} \tag{3.34}$$

The stationary probabilities for the case where the time during which the service is performed is not taken into account are

$$p_0^i,\ \ p_n^i=\int_0^\infty p_n^i(u)du,\ i=1,2,$$

with the normalization condition

$$\sum_{i=1}^{2}\sum_{n=0}^{\infty} p_n^i = 1 \ . \qquad (3.35)$$

To solve the systems of equations (3.32) – (3.35), we introduce generating functions

$$Q_i(z,u) = \sum_{n=1}^{\infty} p_n^i(u) z^n, \ \ i = 1, 2 .$$

The system of equations (3.33) corresponds to a system of the form

$$\frac{d}{du} Q_i(z,u) = -[\lambda_i + r(u)] Q_i(z, u) + a_i z \sum_{j=1}^{2} \lambda_i Q_j(z,u) \ , i = 1, 2, \qquad (3.36)$$

and systems (3.32) and (3.34) correspond to a system of the form

$$zQ_i(z,0) = \int_0^{\infty} Q_i(z,u) r(u) du + a_i z^2 \sum_{j=1}^{2} \lambda_j p_0^j - \lambda_i z p_0^i . \qquad (3.37)$$

The solution to system (3.36) has the form

$$Q_1(z,u) = [1 - B(u)] \left\{ Q_1(z,0) \left[ch(uq) + \frac{\varphi_1}{q} sh(uq) \right] + q^{-1} a_1 \lambda_2 Q_2(z,0) sh(uq) \right\} e^{uf} . \ (3.38)$$

$$Q_2(z,u) = [1 - B(u)] \left\{ Q_2(z,0) \left[ch(uq) + \frac{\varphi_2}{q} sh(uq) \right] + q^{-1} a_2 \lambda_1 Q_1(z,0) sh(uq) \right\} e^{uf} , \ (3.39),$$

Where $f \equiv f(z) = 0{,}5 \cdot \left[(a_1\lambda_1 + a_2\lambda_2) z - (\lambda_1 + \lambda_2) \right]$, $\varphi_i \equiv \varphi_i(z) = f + \lambda_i (1 - a_i z)$,

$$q \equiv q(z) = 0{,}5 \cdot \left[(a_1\lambda_1 + a_2\lambda_2)^2 z^2 - 2\left(a_1\lambda_1^2 - \lambda_1\lambda_2 + a_2\lambda_2^2\right) z + (\lambda_1 - \lambda_2)^2 \right]^{0{,}5} .$$

Substituting (3.38) and (3.39) into (3.37) and solving the resulting system with respect to $Q_i(z,0)$, $i = 1, 2,$ we will receive

$$Q_1(z,0) = \frac{\begin{array}{c} D_1 \cdot \left[2qz - (q - \varphi_1)\beta(-g_1) - (q - \varphi_2)\beta(-g_2) \right] + \\ + a_1\lambda_2 z \left[\beta(-g_1) - \beta(-g_2) \right] \cdot D_2 \end{array}}{2q \left[z - \beta(-g_1) \right] \cdot \left[z - \beta(-g_2) \right]} , \qquad (3.40)$$

$$Q_2(z,0) = \frac{\begin{array}{c} D_2 \cdot \left[2qz - (q - \varphi_2)\beta(-g_1) - (q - \varphi_1)\beta(-g_2) \right] + \\ + a_2\lambda_1 z \left[\beta(-g_1) - \beta(-g_2) \right] \cdot D_1 \end{array}}{2q \left[z - \beta(-g_1) \right] \cdot \left[z - \beta(-g_2) \right]} , \qquad (3.41)$$

where $D_i = -\lambda_i z p_0^i + a_i z^2 \sum_{j=1}^{2} \lambda_j p_0^j$, $i = 1, 2, \ \ g_1 = f + q, \ g_2 = f - q$.

On the border of the region $|z| = 1$ we have

$$|\beta(-g_2|=\left|\int_0^\infty \exp\{-(a_1\lambda_2+a_2\lambda_1)t\}dB(t)\right|<\int_0^\infty dB(t)=1.$$

According to Rouché's theorem, the function $[z-\beta(-g_2)]$ has only one zero ξ inside the unit circle, and the function $[z-\beta(-g_1)]$ has no zeros in the same region. Since $Q_i(z,0),\ i=1,2,$ converges in the region $|z|=1|$, the zeros of the numerator and denominator of the right-hand side of expression (3.40) and (3.41) in the region $|z|=1$ must match and therefore

$$p_0^1=\frac{a_1\lambda_2\xi}{q(\xi)+\varphi_2(\xi)}p_0^2. \tag{3.42}$$

Let's find the probability generating functions p_n^i

$$Q_i(z)=\sum_{n=1}^{\infty}p_n^i z^n=\int_0^\infty Q_i(z,u)du,\ i=1,2.$$

After integration and simplifications we obtain

$$Q_1(z)=\frac{[D_1(q+\varphi_1)+a_1\lambda_2 zD_2]\cdot[\beta(-g_1)-1]}{2qg_1[z-\beta(-g_1)]}+ \\ +\frac{[D_1(q+\varphi_2)-a_1\lambda_2 zD_2]\cdot[\beta(-g_2)-1]}{2qg_2[z-\beta(-g_2)]}. \tag{3.43}$$

$$Q_2(z)=\frac{[D_2(q+\varphi_2)+a_2\lambda_1 zD_1]\cdot[\beta(-g_1)-1]}{2qg_1[z-\beta(-g_1)]}+ \\ +\frac{[D_2(q+\varphi_1)-a_2\lambda_1 zD_1]\cdot[\beta(-g_2)-1]}{2qg_2[z-\beta(-g_2)]}. \tag{3.44}$$

Using the normalization condition (3.35), which can be rewritten as $Q_1(z)+Q_2(z)+p_0^1+p_0^2=1$, we find that

$$p_0^1+p_0^2=p_0=1-\rho, \tag{3.45}$$

where $\rho=\Lambda/\mu-$ system loading.

Substituting (3.42) into (3.45), we find

$$p_0^1=\frac{a_1\lambda_2\xi(1-\rho)}{q(\xi)+\varphi_2(\xi)+a_1\lambda_2\xi},\ p_0^2=\frac{[q(\xi)+\varphi_2(\xi)](1-\rho)}{q(\xi)+\varphi_2(\xi)+a_1\lambda_2\xi}. \tag{3.46}$$

Let's determine the average number of requirements in the system

$$L = \sum_{i=1}^{2}\sum_{n=1}^{\infty} n p_n^i = \sum_{i=1}^{2}\left[\frac{d}{dz} Q_i(z)\right]_{z=1} =$$

$$= \rho + \frac{\rho}{2(1-\rho)}\left[\left(C_a^2 - 1\right) + \rho\left(1 + C_b^2\right)\right] - \frac{\Lambda}{\lambda_1 \cdot \lambda_2}\left(\Lambda - \frac{1}{1-\rho}\sum_{i=1}^{2}\lambda_i p_0^i\right)',$$

Where $C_a^2 = 2\Lambda^2 \sum_{i=1}^{2} \frac{a_i}{\lambda_i^2} - 1$ the square of the coefficient of variation of the input flow, and C_b^2 – the square of the coefficient of variation of service time.

Let us define the Laplace-Stieltjes transform of the waiting time distribution function

$$\varphi(s) = p_0 + \int_0^{\infty}\left[\sum_{i=1}^{2}\sum_{n=1}^{\infty} p_n^i(u)\beta^{n-1}(s)\int_0^{\infty} e^{-sv}\frac{B'(u+v)}{1-B(u)}dv\right]du,$$

Where $\int_0^{\infty} e^{-sv}\frac{B'(u+v)}{1-B(u)}dv$ – the Laplace-Stieltjes transform of the distribution function of the time required to complete the servicing of a request that, at the time the next request arrives, has already been serviced for a time u [18]. Using formulas (3.38) – (3.41), after the appropriate transformations we obtain

$$\varphi(s) = (1-\rho) + \frac{\left[1-\beta(s)\right]\cdot\left[\lambda_1\lambda_2(1-\rho) - s\sum_{i=1}^{2}\lambda_i p_0^i\right]}{\lambda_1\lambda_2\left[\beta(s)-1\right] + \left[\lambda_1 + \lambda_2 - (a_1\lambda_1 + a_2\lambda_2)\beta(s)\right]s - s^2}.$$

The average waiting time is

$$\tau_q = \varphi'(s)\big|_{s=0} = \frac{\rho^2\left(C_a^2 + C_b^2\right)}{2\Lambda(1-\rho)} - \frac{\rho}{\lambda_1\lambda_2}\left(\Lambda - \frac{1}{1-\rho}\sum_{i=1}^{2}\lambda_i p_0^i\right). \qquad (3.47)$$

Note that expression (3.47) is a generalization of the fundamental Polyachek-Khinchin formula [14] for the system under study.

The method used can be extended to the case of a *k*-th order hyperexponential input flow. In this case, equations (3.32) – (3.34) will have a similar form, except that *i* will take values from 1 to *k*. However, the possibility of solving them analytically is questionable.

3.4 Multi-channel queuing system with limited queue

Consider a Poisson queueing system with *n* service channels and the number of waiting places *m* (M/M/n/m in Kendall notation). Let {i}, $i = 1, 2, ..., n+m-1, n+m$ denote the state of the system when there are *i* requests in it. If $i \geq n$, then *n* requests will be served, and the rest are in the queue. Let the

probability of state $\{i\}$ be denoted by p_i. The state and transition graph for this system is shown in Figure 3.8.

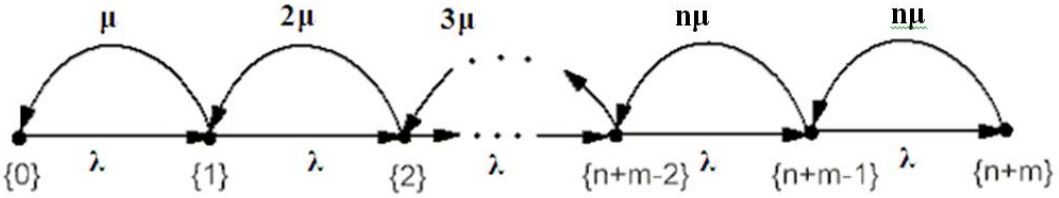


Figure 3.8 – State and transition graph for the M/M/n/m system

Using the method of section 1.2.3, we will construct a system of linear equations for the probabilities $p_i, i = 1,2,...,n+m$:

$$\begin{aligned} &-\lambda p_0 + \mu p_1 = 0, \\ &-(\lambda + i\mu)p_i + \lambda p_{i-1} + (i+1)\mu p_i = 0,\ 1 \le i < n, \\ &-(\lambda + n\mu)p_i + \lambda p_{i-1} + n\mu p_i = 0,\ n \le i < n+m, \\ &-n\mu p_{n+m} + \lambda p_{n+m-1} = 0. \end{aligned} \tag{3.48}$$

System (3.48) is supplemented by the normalization condition

$$p_0 + p_1 + p_2 + ... + p_{n+m-1} + p_{n+m} = 1. \tag{3.49}$$

Divide each equation by $n\mu$ and introduce the notation $\rho = \lambda / n\mu$ – the workload of the entire system. System (3.48) can be rewritten as follows:

$$\begin{aligned} &-\rho p_0 + p_1 / n = 0, \\ &-(\rho + i/n)p_i + \rho p_{i-1} + (i+1)/n \cdot p_i = 0,\ 1 \le i < n, \\ &-(\rho + 1)p_i + \rho p_{i-1} + p_i = 0,\ n \le i < n+m, \\ &-p_{n+m} + \rho p_{n+m-1} = 0. \end{aligned} \tag{3.50}$$

From the first equation of system (3.49) it follows that: $p_1 = n\rho p_0$. Substitute this expression into the second equation of system (3.49). We obtain: $p_2 = (n^2/2!)\rho^2 p_0$. Continuing this operation, we get:

$$\begin{aligned} &p_i = (n^i/i!)\rho^i p_0, 1 \le i \le n, \\ &p_i = (n^n/n!)\rho^i p_0, n < i \le n+m. \end{aligned} \tag{3.51}$$

By substituting p_i in the normalization condition (3.49), we find an expression for the probability of system downtime:

$$P_{downtime} = p_0 = \left[1 + \sum_{i=1}^{n-1} \frac{n^i \rho^i}{i!} + \sum_{i=n}^{n+m} \frac{n^n \rho^i}{n!}\right]^{-1} = \left[\sum_{i=0}^{n-1} \frac{n^i \rho^i}{i!} + \frac{n^n}{n!} \frac{\rho^n (1-\rho^{m+1})}{1-\rho}\right]^{-1}.$$

So, all state probabilities are determined.

Let's define the basic indicators of this system.

Probability of service failure –

$$P_{fail} = p_{m+n} = (n^n / n!)\rho^{n+m} p_0.$$

This expression means that the request will be denied service if all lines and waiting spaces are occupied.

The number of requests that will receive a denial of service during time T – N_{fail} can be calculated using the expression:

$$N_{fail} = \lambda p_{m+n} T = \lambda (n^n / n!)\rho^{n+m} p_0 T.$$

The probability that the request will be served by the system is –

$$P_{service} = 1 - P_{faik} = 1 - (n^n / n!)\rho^{n+m} p_0.$$

In addition, it is the relative throughput of the system, which is equal to the average fraction of requests that enter the system and are served by it.

Absolute throughput A is equal to the average number of requests that the queuing system can serve per unit of time, that is, it is the fraction of the incoming flow of requests that enters the system and is served by it:

$$\text{A} = \lambda P_{service} = \lambda (1 - (n^n / n!)\rho^{n+m} p_0).$$

Let us define the average number of requests waiting for service – L_q. If the queuing system is in the state {n+1}, then there will be one request in the queue, the probability of this is p_{n+1}. If the queuing system is in the state {n+m}, then there will be m requests in the queue, the probability of this is p_{n+m}. The average number of requests in the service queue is equal to the mathematical expectation of the number of requests in the queue, it can be calculated by the formula:

$$L_q = \sum_{i=1}^{m} i \cdot p_{n+i} = \sum_{i=1}^{m} i \cdot (n^n / n!) \cdot \rho^{n+i} p_0 = (n^n / n!)\rho^{n+1}(1 + 2\rho + .. + m\rho^{m-1}) p_0. \quad (3.52)$$

If $\rho = 1$, then the expression in parentheses becomes an arithmetic progression $1 + 2 + 3 + ... + m$. Its sum is equal to $m(m+1)/2$. Then

$$L_q = 0{,}5(n^n / n!)m(m+1) p_0.$$

Let $\rho \neq 1$. Let us simplify the expression in parentheses by expression (3.52):

$$1+2\rho+3\rho^2+...+m\rho^{m-1}=\frac{d}{d\rho}(\rho+\rho^2+\rho^3+...+\rho^m).$$

The expression in parentheses is a finite geometric progression with denominator ρ. The sum of its terms is

$$\rho+\rho^2+\rho^3+...+\rho^m=\frac{\rho(1-\rho^m)}{1-\rho}. \quad (3.53)$$

Let us calculate the derivative of expression (3.53):

$$\frac{d}{d\rho}\left(\frac{\rho(1-\rho^m)}{1-\rho}\right)=\frac{1-\rho^m(1+m-m\rho)}{(1-\rho)^2}. \quad (3.54)$$

Substituting (3.44) into (3.52), we can find L_q at $\rho\neq1$. Finally:

$$L_q=\begin{cases}\dfrac{n^n}{n!}\dfrac{1-\rho^m(1+m-m\rho)}{(1-\rho)^2}\rho^{n+1}p_0, & \rho\neq1,\\ 0{,}5(n^n/n!)m(m+1)p_0, & \rho=1.\end{cases} \quad (3.55)$$

Figure 3.9 shows a graph of the dependence of the average number of requests waiting for service on the load of the entire queuing system ρ for different values of the number of channels n and the same number of waiting places $m=7$.

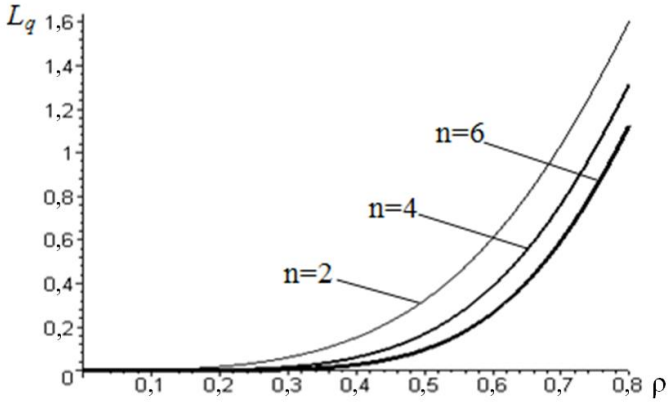


Figure 3.9 – Dependence of L_q on system load ρ at different n

It is clear that when $\rho\geq0{,}5$ the queue length increases sharply; with increasing n and the same value of ρ, L_q also increases sharply, and then the increase slows down.

Figure 3.10 shows a graph of the dependence of the average number of requests waiting for service on the load of the entire queuing system ρ for different values of the number of channel waiting places m and the same number of channels $n=5$.

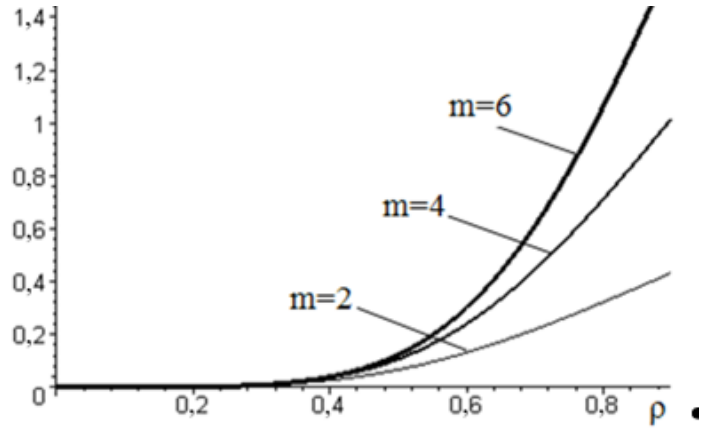


Figure 3.10 – Dependence of L_q on system loading ρ at different m

It is clear that the number of waiting places significantly affects the length of the queue in the system. As m increases, L_q also increases. This is explained by the fact that fewer requests are denied service. In addition, as in the previous case, when $\rho \geq 0{,}5$ the length of the queue increases dramatically.

Let's determine the average number of channels occupied by service – $\overline{n}$, and accordingly, the average number of requests served. Each busy channel serves on average μ requests per unit of time, and the queuing system in general – A requests. By dividing A on μ, we get $\overline{n}$:

$$\overline{n} = \frac{\mathrm{A}}{\mu} = \frac{\lambda(1-(n^n/n!)\cdot\rho^{n+m}p_0)}{\mu} = n\rho(1-(n^n/n!)\rho^{n+m}p_0).$$

This leads to an important indicator – the system channel load factor – z_s:

$$z_s = \overline{n}/n = \rho(1-(n^n/n!)\rho^{n+m}p_0).$$

Average number of service requests and queues (in the system):

$$L_s = \overline{n} + L_q = n\rho(1-(n^n/n!)\cdot\rho^{n+m}p_0) + \frac{n^n}{n!}\frac{1-\rho^m(1+m-m\rho)}{(1-\rho)^2}\rho^{n+1}p_0,\ \rho \neq 1. \quad (3.56)$$

Average request waiting time in the queue τ_q we find by Little's formula (2.39) – $L_q = \lambda\tau_q$:

$$\tau_q = \frac{L_q}{\lambda} = \frac{L_q}{n\mu\rho}. \quad (3.57)$$

Substituting (3.55) into (3.57), we obtain:

$$\tau_q = \begin{cases} \dfrac{n^n}{\lambda n!}\cdot\dfrac{1-\rho^m(1+m-m\rho)}{(1-\rho)^2}\rho^{n+1}p_0, & \rho \neq 1, \\ 0{,}5(n^n/\lambda n!)m(m+1)p_0, & \rho = 1. \end{cases}$$

or

$$\tau_q = \begin{cases} \dfrac{n^{n-1}}{\mu n!} \cdot \dfrac{1-\rho^m(1+m-m\rho)}{(1-\rho)^2}\rho^n p_0, & \rho \neq 1, \\ 0{,}5(n^{n-1}/\mu n!)m(m+1)p_0, & \rho = 1. \end{cases}$$

Average request dwell time in the system τ_s for $\rho \neq 1$ we also find by Little's formula (2.42) – $L_s = \lambda \tau_s$:

$$\tau_s = \frac{L_s}{\lambda} = \frac{L_s}{n\mu\rho}. \qquad (3.58)$$

Substituting (3.56) into (3.58), we obtain:

$$\tau_s = (1-(n^n/n!)\cdot\rho^{n+m}p_0)/\mu + \frac{n^n}{n!}\frac{1-\rho^m(1+m-m\rho)}{\lambda(1-\rho)^2}\rho^{n+1}p_0$$

or

$$\tau_s = (1-(n^n/n!)\cdot\rho^{n+m}p_0)/\mu + \frac{n^{n-1}}{n!}\frac{1-\rho^m(1+m-m\rho)}{\mu(1-\rho)^2}\rho^n p_0$$

If the M/M/n/m queuing system models an economic system in which requests are served, then for its optimization we can use the criterion – the function of the cost of losses in the system for time T – G:

$$E = (C_{downtime}(1-z_s)n + C_q L_q + \lambda C_{fail} P_{fail} + C_e z_s n)T,$$

where $C_{downtime}$ – cost of a unit of channel downtime; C_q – the cost of losses from the request being idle in the queue per unit of time; C_{fail} – the cost of losses from leaving the system of a request that was not serviced; C_e – the cost of operating each channel of the system per unit of time.

The profit from the operation of the SMO – G can also be used as a criterion:

$$G = \lambda C_s P_{service} T - (C_{prost}(1-z_c)n + C_q L_q + \lambda C_{fail} P_{fail} + C_e z_s n)T,$$

where C_s – the cost of servicing each request, i.e. this is the gross profit received when servicing each request.

Optimization can be performed either by the number of service channels n or by the service intensity μ, provided that this parameter can be changed. It is also possible to assign a service cost to each request C_s, so that the profit G is not less than the planned one. It is advisable to carry out the optimization using numerical methods [19].

3.5 Single-channel queuing system with arbitrary event streams

In Kendall's notation, a single-channel system is denoted as G/G/1, i.e. it is a queuing system with an arbitrary input flow described by the distribution function $A(t)$ and an arbitrarily distributed service function – $B(t)$. This system is studied in detail in [3].

Let $\overline{\lambda} = \left[\int_0^\infty t dA(t)\right]^{-1}$ – average intensity of the incoming flow, and $\overline{\mu} = \left[\int_0^\infty t dB(t)\right]^{-1}$ – average intensity of service flow. If $\overline{\mu} > \overline{\lambda}$, then for the steady state of this system there exists a waiting time distribution function $W(t)$, for which we can determine the Lindley integral equation [3]

$$W(t) = \begin{cases} \int_{-\infty}^{t} W(t-x) dC(x), & t \geq 0, \\ 0, & t < 0, \end{cases}$$

Where $C(t) = \int_0^\infty B(x+t) dA(t).$

The solution can be found by factoring the Laplace–Stiehljes transform of the kernel of the integral equation. The corresponding factorization equation will have the following form:

$$\alpha(-s)\beta(s) - 1 = 0,$$

where $\alpha(s)$ and $\beta(s)$ – Laplace–Stiehljes transform of $A(t)$ and $B(t)$ respectively. Let us use Theorem 4 from [18]: if the Laplace–Stiehljes transform of $B(t)$ – $\beta(s)$ is the ratio of two polynomials $\beta(s) = P_m / Q_n$, moreover $n > m$ and $exp(\lambda t)[1 - A(t)]$ has limited variation in $(0, \infty)$, then the transformation Laplace – Stiehljes from $W(t) - \varphi(s)$ will look like this:

$$\varphi(s) = \frac{Q_n(s)}{Q_n(0)\prod_{i=1}^{n}\left(1 - \frac{s}{q_i}\right)} \tag{3.59}$$

where q_i – zeros of the function $1 - \alpha(-s)\beta(s)$ in the left half-plane, where $\operatorname{Re} s < 0$. Choosing the inverse Laplace–Stiehljes transform of $\varphi(s)$, we can obtain an expression for the waiting time distribution function $W(t)$ in a queue.

Example. Let us investigate a system with hyperexponential input flow and Erlang service time distribution, i.e.

$A(t)=1-\sum_{i=1}^{m}a_i\exp(-\lambda_i t),\ B(t)=1-\sum_{i=1}^{n}\frac{(\mu t)^{i-1}}{(i-1)!}e^{-\mu t}$. Accordingly $\overline{\lambda}=\left(\sum_{i=1}^{m}\frac{a_i}{\lambda_i}\right)^{-1}$, $\overline{\mu}=\frac{\mu}{n}$. Laplace–Stiehljes transform from $A(t)$ and $B(t)$ has the following form:

$$\alpha(s)=\sum_{i=1}^{m}\frac{a_i\lambda_i}{s+\lambda_i},\ \beta(s)=\left(\frac{\mu}{s+\mu}\right)^n=\frac{\mu^n}{(s+\mu)^n}=\frac{Q_0(s)}{Q_n(s)}.$$

Then

$$\varphi(s)=\frac{(\mu+s)^n}{\mu^n\prod_{i=1}^{n}\left(1-\frac{s}{q_i}\right)}, \tag{3.60}$$

where q_i – roots of the equation in the left half-plane

$$\left(\sum_{i=1}^{m}\frac{a_i\lambda_i}{\lambda_i-s}\right)\cdot\left(\frac{\mu}{\mu+s}\right)^n-1. \tag{3.61}$$

Let's choose $s=\mu(z-1)$, then equation (3.61) will be as follows:

$$\sum_{i=1}^{m}\frac{a_i\lambda_i}{\lambda_i+\mu(1-z)}-z^n=0. \tag{3.62}$$

Equation (3.62) has $m+n$ zeros, the *n* zeros of this equation are contained in the circle $|z|=1$, taking into account their multiplicity. To prove this fact, we find the values of the moduli of both applications (3.62) on the boundary of the domain $|z|=1-\varepsilon,\ \varepsilon\to+0$, then

$$|-z|^n=(1-\varepsilon)^n. \tag{3.63}$$

If we expand the right-hand side of (3.63) using Taylor's formula and discard the terms of the highest order of smallness with respect to the first, we obtain: $|-z|^n=1-n\varepsilon$. Similarly:

$$\left|\sum_{i=1}^{m}\frac{a_i\lambda_i}{\lambda_i+\mu(1-z)}\right|=1-\mu\varepsilon\sum_{i=1}^{m}\frac{a_i}{\lambda_i};$$

Because $\overline{\mu}>\overline{\lambda}$, then $\varepsilon n<\varepsilon\mu\sum_{i=1}^{m}\frac{a_i}{\lambda_i}\Rightarrow n<\mu\sum_{i=1}^{m}\frac{a_i}{\lambda_i}$, that is on the border of the region $|z|=1-\varepsilon,\varepsilon\to+0$, $\left|-z^n\right|>\left|\sum_{i=1}^{m}\frac{a_i\lambda_i}{\lambda_i+\mu(1-z)}\right|$. According to Rouchet's theorem, inside the domain $|z|=1,\left|-z^n\right|$ and $\sum_{i=1}^{m}\frac{a_i\lambda_i}{\lambda_i+\mu(1-z)}-z^n$ have the same

number of zeros – *n*. We denote them as $z_1, z_1, ..., z$. Using what $s = \mu(z-1)$, we get $q_i = \mu(z_i - 1),\ i = 1,2,...,n$. Finally:

$$\varphi(s) = (\mu + s)^n \prod_{i=1}^{n} \left(\frac{z_i - 1}{\mu(z_i - 1) - s} \right). \tag{3.64}$$

By applying the inverse transformation Laplace – Stiltjes from(3.64), we obtain an expression for the waiting time distribution function in the queue:

$$W(t) = 1 - \sum_{i=1}^{n} \frac{\prod_{e=1}^{n} (z_e - 1) z_i^n}{\sum_{j=1}^{n} \frac{\prod_{e=1}^{n} (z_e - z_i)}{(z_j - z_i)}} e^{-\mu(1-z_i)t}. \tag{3.65}$$

Applying the obtained expressions, one can find the basic indicators of the studied queuing system.

Probability of downtime–

$$P_{downtime} = p_0 = W(0) = \lim_{s \to \infty} \varphi(s) = \prod_{i=1}^{n} (1 - z_i).$$

Average waiting time in line

$$\tau_q = -\varphi'(s)\big|_{s=0} = \sum_{i=1}^{n} \left[\frac{1}{\mu(1-z_i)} - \frac{n}{\mu} \right].$$

The average number of requests in the queue is determined by Little's formula (2.39)

$$L_q = \left(\sum_{i=1}^{m} \frac{a_i}{\lambda_i} \right)^{-1} \sum_{i=1}^{n} \left[\frac{1}{\mu(1-z_i)} - \frac{n}{\mu} \right].$$

Using these expressions, it is possible to evaluate the main characteristics of the system's functioning depending on its parameters.

3.6 Queuing system with a finite number of request sources

A significant number of real-world technical systems that are modeled using queuing systems have a finite number of request sources. A request source does not send a new request for service until the previous request is serviced. An example is an input-output system, the devices of which periodically send requests for the transfer of information arrays. Probabilistic models of such systems are queueing systems with a finite number of request sources, which are also called closed systems.

Let us investigate such a system under the following assumptions:

– system – Markov, number of query sources – *k*, the intensity of initiating requests from a single source – λ, the source does not send a new service request until the previous request is serviced;

– single-line system, intensity of service requests – μ;

– if the service channel is busy, then the initiated request waits for service either in the queue or at the request source itself.

Let us denote by $S_i, i = 0,1,...,k$ the state of the system when *i* requests have been initiated, one is being served, $i-1$ are waiting for service, and $k-i$ sources have not initiated service requests. The probability of state S_i is $p_i(t)$.

Figure 3.11 shows the state and transition graph of such a queuing system when the number of request sources of the closed system is five.

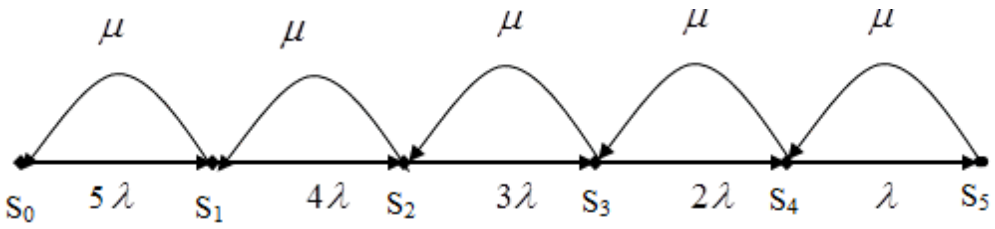


Figure 3.11 – State and transition graph of a closed queuing system

Let us derive a system of differential equations for the state probabilities. Intensity of request initiation from a single source – λ, and from *j* sources – $j\lambda$. If the system is in a state $i, i = 0,1,..,k$, then the total intensity of incoming requests – $(k-i)\lambda$.

Using the method of composing differential equations for state probabilities from subsection 1.2.3 we obtain the following system of linear differential equations:

$$\begin{aligned} p_0(t) &= -k\lambda p_0(t) + \mu p_1(t), \\ p_1(t) &= -[(k-1)\lambda + \mu]p_0(t) + k\lambda p_0(t) + \mu p_2(t), \\ p_i(t) &= -[(k-i)\lambda + \mu]p_i(t) + (k-i+1)\lambda p_{i-1}(t) + \mu p_{i+1}(t), 0 < i < k \\ p_k(t) &= -\mu p_k(t) + \lambda p_{k-1}(t). \end{aligned} \qquad (3.66)$$

Initial conditions – $p_0(0) = 1, p_i(0) = 0, 0 < i \le k.$

To solve the system (3.66), we choose the numerical Euler method for solving systems of differential equations [25]. We denote by p_i^l probability value

$p_i(t)$ at the *l*-th step of the calculations. System (3.66) will be written in a form ready for iterative calculation

$$\begin{aligned}
&p_0^{l+1} = p_0^l + h\cdot(-k\lambda p_0^l + \mu p_1^{lk}), l = 0,1,2...,\\
&p_i^{l+1} = p_i^l + h\cdot[-((k-i)\lambda + \mu)p_i^l + (k-i+1)\lambda p_{i-1}^l + \mu p_{i+1}^l),\ 0 < i < k, l = 0,1,2...,\\
&p_k^{l+1} = p_k^l + h\cdot(-\mu p_k^l + \lambda p_{k-1}^l), l = 0,1,2...,\\
&p_0^0 = 1,\ p_i^0 = 0, i \geq 1,
\end{aligned} \tag{3.67}$$

where *h*–integration step.

To numerically solve the system of differential equations (3.66) using formulas (3.67), it is necessary to choose specific values of the parameters λ, μ and *h*. The integration step *h* will have to be chosen experimentally, and the solution will be approximate.

Figure 3.12 shows the probability graphs of the state of the studied system for the transition regime at $\lambda = 0{,}2;\ \mu = 1;\ k = 5;\ h = 0{,}1$.

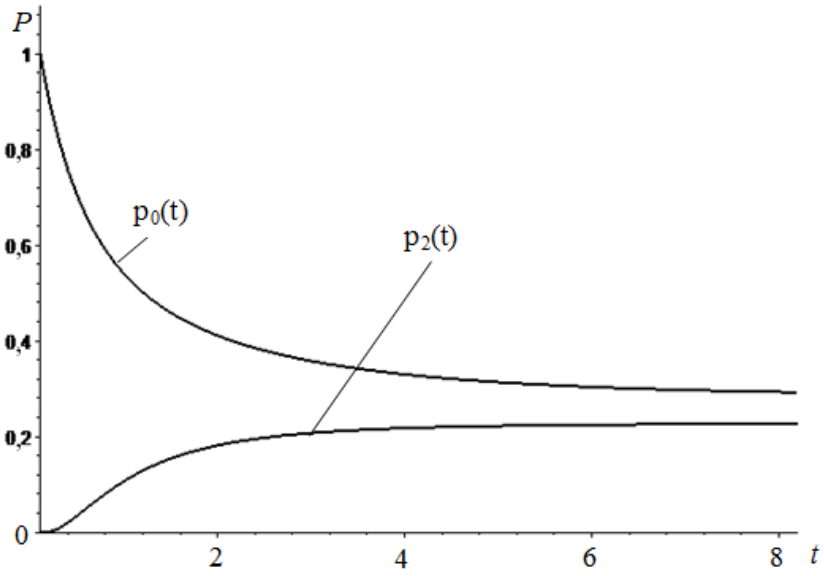


Figure 3.12 – Transitional mode of queuing system with a finite number of query sources

It is clear that from the moment of time $t = 5$ the system enters a steady-state operating mode.

Let us investigate the steady-state mode of operation of the queuing system with a finite number of request sources. For the steady-state mode $p_i(t) \to p_i$, $p_i' = 0$ system (3.66) will take the following form:

$$
\begin{aligned}
&-k\lambda p_0 + \mu p_1 = 0,\\
&-[-(k-1)\lambda + \mu]p_1 + k\lambda p_0 + \mu p_2 = 0,\\
&-[-(k-2)\lambda + \mu]p_2 + (k-1)\lambda p_1 + \mu p_3 = 0,\\
&-[-(k-i)\lambda + \mu]p_i + (k-i+1)\lambda p_{i-1} + \mu p_{i+1} = 0,\ 0 < i < k,\\
&-\mu p_k + \lambda p_{k-1} = 0.
\end{aligned}
\tag{3.68}
$$

Normalization condition:

$$
\sum_{i=0}^{k} p_i = 1. \tag{3.69}
$$

Divide each equation of system (3.68) by μ and introduce the exponent $\rho = \lambda/\mu$ loading the serving channel with one source of requests. System (3.68) will be as follows:

$$
\begin{aligned}
&-k\rho p_0 + p_1 = 0,\\
&-[-(k-1)\rho + 1]p_1 + k\rho p_0 + p_2 = 0,\\
&-[-(k-2)\rho + 1]p_2 + (k-1)\rho p_1 + p_3 = 0,\\
&-[-(k-i)\rho + 1]p_i + (k-i+1)\rho p_{i-1} + p_{i+1} = 0, 0 < i < k,\\
&-p_k + \rho p_{k-1} = 0.
\end{aligned}
\tag{3.70}
$$

By successively solving the equations of the system (3.70) with respect to p_0 we obtain:

$$
p_i = \frac{k!}{(k-i)!}\rho^i p_0. \tag{3.71}
$$

Using the normalization condition (3.69), we find an expression for p_0:

$$
p_0 = \left[\sum_{i=0}^{k} \frac{k!}{(k-i)!}\rho^i\right]^{-1}. \tag{3.72}
$$

Hence, the average number of requests in the system will be as follows: $L_s = \sum_{i=1}^{k} i \cdot p_i$, and the average number of request sources waiting to be serviced is – $L_q = \sum_{i=1}^{k} (i-1) \cdot p_i$. After transformations we get:

$$
L_q = \sum_{i=1}^{k} (i-1) \cdot p_i = \sum_{i=1}^{k} i p_i - \sum_{i=1}^{k} p_i = L_s - (1 - p_0). \tag{3.73}
$$

The average number of sources that have not submitted service requests will be $k - L_s$; the fraction of time when the service channel is busy is equal to $1 - p_0$. Numbers $k - L_s$ and $1 - p_0$ are proportional to the average time that requests spend in the corresponding state, hence

$$\frac{1/\lambda}{k-L_s}=\frac{\tau_q}{L_q}=\frac{1/\mu}{1-p_0}=\frac{\tau_s}{L_s}. \qquad (3.74)$$

Using expression (3.74), we can express the average number of requests in the system as a function of p_0. $\frac{\frac{1}{\lambda}(1-p_0)}{\frac{1}{\mu}}=k-L_s \Rightarrow L_s=k-\frac{1-p_0}{\frac{\lambda}{\mu}}$. Finally:

$$L_s=k-\frac{1-p_0}{\rho}. \qquad (3.75)$$

From expression (3.74) it follows: $\frac{1/\lambda}{k-L_s}=\frac{\tau_s}{L_s}$, hence the average time a request from a source is in the system – $\tau_s=\frac{L_s}{\lambda(k-L_s)}=\frac{k-\frac{1-p_0}{\rho}}{\lambda\left(k-k+\frac{1-p_0}{\rho}\right)}=\frac{k\rho-(1-p_0)}{\lambda(1-p_0)}$. Finally:

$$\tau_s=\frac{1}{\lambda}\left(\frac{k\rho}{1-p_0}-1\right). \qquad (3.76)$$

From expression (3.74) it follows that the average waiting time for a service request is equal to $\tau_q=L_q/[\lambda(k-L_s)]$. Using (3.73) and (3.75), we obtain:

$$\tau_q=\frac{L_q}{\lambda(k-L_s)}=\frac{L_s-(1-p_0)}{\lambda(k-L_s)}=\frac{k-\frac{1-p_0}{\rho}-(1-p_0)}{\lambda\left(k-k-\frac{1-p_0}{\rho}\right)}=\frac{k\rho-(1+\rho)(1-p_0)}{\lambda(1-p_0)}.$$

Finally: $\tau_q=\frac{1}{\lambda}\left(\frac{k\rho}{1-p_0}-(1+\rho)\right)$.

Figure 3.13 shows a graph of the average waiting time for a service request τ_q on the intensity of requests from the source at $\mu=1$ and different numbers of request sources.

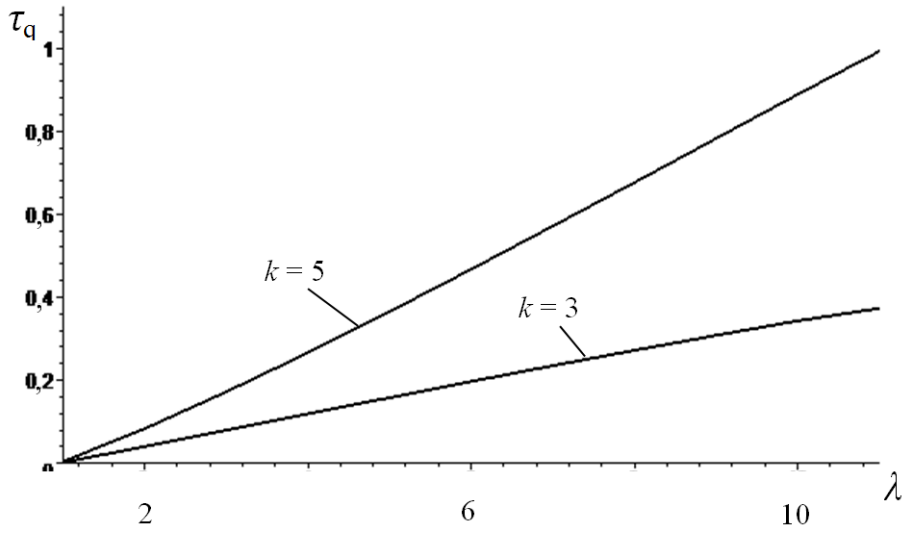


Figure 3.13 – Dependence of service waiting time on parameters λ and k

These dependencies, unlike an open system, are linear.

3.7 Absolute priority queueing system

In computer systems, especially those that operate in real time, there are preferences for servicing some requests over others – priorities. Modern operating systems have up to 16 priority levels. There are relative, absolute, dynamic, situational and mixed priorities [15]. With relative priorities, it is not allowed to interrupt the servicing of requests; only after the channel is released from the queue is the request with the highest priority selected for servicing. With absolute priorities, a request that has recently arrived with a certain priority stops servicing a request with a lower priority. With dynamic priorities, the priority of individual requests in the queue can change depending on changes in some indicators, for example, the waiting time in the queue. The general theory of queuing system with priorities is quite complex and goes beyond the scope of a higher educational institution course.

As an example of a priority queuing system, let us consider a single-channel Poisson system with absolute priority, to which two streams of requests arrive: non-priority and priority. Priority requests are serviced in the order of arrival. If a priority request arrives in the system when it is servicing a non-priority request, then this service is terminated, and the channel begins to service a priority request. A non-priority request, the service of which has been interrupted, is serviced in the future only after the channel is freed from servicing priority requests. The intensity of the priority flow of requests is λ_1, the intensity of the non-priority flow of requests is λ_2. The intensity of service of incoming flows, respectively, μ_1 and μ_2.

Denote by S_{ij} the state of the system when i priority and j non-priority requests have been received for service. If $j > 0$, then this means that at this point in time the priority request is being served. The graph of states and transitions of the studied system with absolute priority is shown in Figure 3.14.

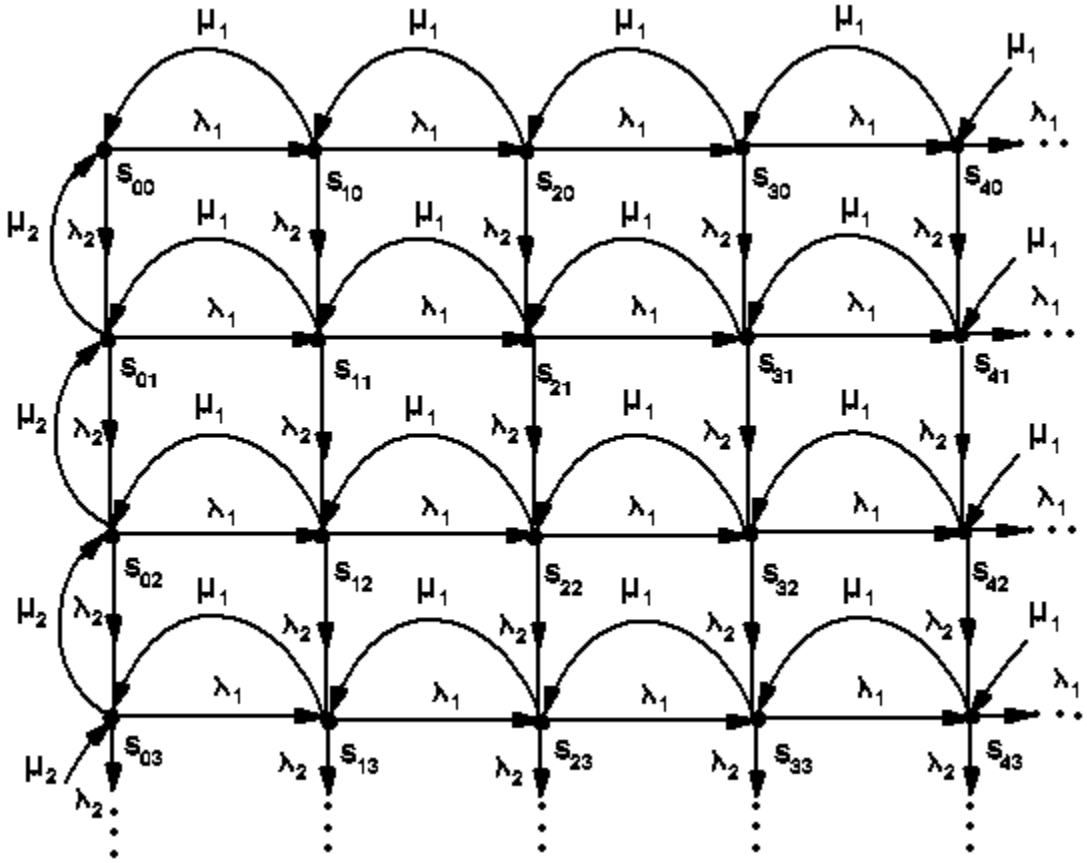


Figure 3.14 – State and transition graph for a queueing system with two input streams and absolute priority

If the number of priority levels is greater than or equal to two, then the graph of states and transitions will take on a three-dimensional form and it will be impossible to study such queuing systems using the apparatus of state probabilities. In view of this, certain difficulties arise when studying priority systems using analytical methods.

We will demonstrate the study of a queuing system with two input streams and absolute priority based on the provisions of [15].

Let's assume that $\rho_1 = \lambda_1 / \mu_1$, $\rho_2 = \lambda_2 / \mu$ – loading of the service channel for each request flow. If $\rho = \rho_1 + \rho_2 < 1$, then there is a stationary mode of operation of the system. Let us denote by $p_{i,j}$ stationary state probability S_{ij}. Using the method of section 1.2.3, we will construct a system of linear equations for the probabilities $p_{i,j}, i = 0,1,2,..., j = 0,1,2,....$:

$$
\begin{aligned}
&-(\lambda_1+\lambda_2)p_{0,0}+\mu_1 p_{1,0}+\mu_2 p_{0,1}=0,\\
&-(\lambda_1+\lambda_2+\mu_2)p_{0,j}+\mu_1 p_{1,j}+\mu_2 p_{0,j+1}+\lambda_2 p_{0,j-1}=0,\ j>0,\\
&-(\lambda_1+\lambda_2+\mu_1)p_{i,0}+\lambda_1 p_{i-1,0}+\mu_1 p_{i+1,0}=0, i>0,\\
&-(\lambda_1+\lambda_2+\mu_1)p_{i,j}+\lambda_1 p_{i-1,j}+\lambda_2 p_{i,j-1}+\mu_1 p_{i+1,j}, i>0,\ j>0.
\end{aligned}
\tag{3.77}
$$

The normalization condition is

$$
\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} p_{i,j}=1. \tag{3.78}
$$

To solve the system of equations (3.77) according to the normalization condition (3.78), we introduce the generating function

$$
G(u,z)=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} p_{i,j}u^i z^j\ . \tag{3.79}
$$

For a queuing system with an infinite number of waiting places, the probability of idleness does not depend on the service discipline and will coincide with the probability of idleness for the M/M/1 system, i.e. $p_0=p_{0,0}=1-\rho=1-\rho_1-\rho_2=G(0,0)$.

At $u=1,\ z=1$ $G(u,z)$ is transformed into a normalization condition and $G(1,1)=1$. Multiplying each equation (3.77) by $u^i z^j$ and summing up for all with the values of *i*, *j*, we get:

$$
\begin{aligned}
&(\lambda_1+\lambda_2+\mu_1)G(u,z)+\sum_{j=0}^{\infty}(\mu_2-\mu_1)p_{0,j}z^j-\mu_2 p_{0,0}=\\
&=\lambda_1\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}p_{i-1,j}u^i z^j+\mu_1\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}p_{i+1,j}u^i z^j++\lambda_2\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}p_{i,j-1}u^i z^j+\mu_2\sum_{j=0}^{\infty}p_{0,j+1}z^j.
\end{aligned}
\tag{3.80}
$$

By definition $G(u,z)$ it follows:

$$
G(u,z)=\frac{[\mu_1 z(u-1)-\mu_2 u(z-1)]G(0,z)+\mu_2 u(z-1)(1-\rho)}{\lambda_1 uz(1-u)+\lambda_2 uz(1-z)+\mu_1 z(u-1)}. \tag{3.81}
$$

The denominator of the expression is a polynomial of the second degree in *u* and has the root u_1 less than zero and root u_2 greater than zero.

$$
u_1=\frac{\lambda_1+\lambda_2(1-z)+\mu_1-\sqrt{[\lambda_1+\lambda_2(1-z)+\mu_1]^2-4\lambda_1\mu_1}}{2\lambda_1}. \tag{3.82}
$$

Since the double series (3.79) is convergent for $u\le 1, z\le 1$, then u_1 must be the root of the numerator as well, whence

$$
G(0,z)=\frac{\mu_2(z-1)u_1(1-\rho)}{\mu_1 z-u_1[\mu_1 z-\mu_2(z-1)]}. \tag{3.83}
$$

Substituting G(0,z) into (3.81), after transformations we obtain:

$$G(u,z)=\frac{(1-\rho)(1-\rho_1 u_1)}{(1-\rho_1 u_1-\rho_2 z)(1-\rho_1 u_1 u)}. \quad (3.84)$$

State probabilities $p_{i,j}$ can be found using partial derivatives of $G(u,z)$:

$$p_{i,j}=\frac{1}{i!\,j!}\left[\frac{\partial^{i+j}}{\partial u^i \partial z^j}G(u,z)\right]_{u=z=0}. \quad (3.85)$$

If we consider only priority requests, then non-priority requests do not affect the process of their servicing, hence the characteristics of servicing priority requests coincide with the characteristics of the M/M/1 type of queuing system with parameters $\lambda=\lambda_1, \mu=\mu_1$. Accordingly $\rho=\rho_1=\lambda_1/\mu_1$. For example, the average number of priority requests in the system is $L_{s1}=1/(1-\rho_1)$, and the average waiting time in the queue is $\tau_{q1}=\rho_1/[\mu_1(1-\rho_1)]$.

Let us denote by $q_i=\sum_{j=0}^{\infty}p_{i,j}$ the unconditional probability that there are *i* priority requests in the system, and because of $g_j=\sum_{i=0}^{\infty}p_{i,j}$ the unconditional probability that there are *j* unpriority requests in the system. The generating function for the unconditional probabilities g_j is defined as

$$G(1,z)=\sum_{j=0}^{\infty}g_j z^j=\frac{1-\rho}{(1-\rho_1 u_1-\rho_2 u)}. \quad (3.86)$$

Using expression (3.86), we can find the average number of non-priority requests in the system:

$$L_{s2}=\sum_{j=0}^{\infty}jg_j=\left.\frac{dG(1,z)}{dz}\right|_{z=1}. \quad (3.87)$$

Let's calculate the derivative of G(1,z) with respect to z:

$$\frac{dG(1,z)}{dz}=(1-\rho)\frac{\rho_1(u_1)'_z+\rho_2}{(1-\rho_1 u_1-\rho_2 z)^2}. \quad (3.88)$$

Let's use the property of derivatives $y'_x=1/x'_y$. Then expression (3.88) will take the following form:

$$\frac{dG(1,z)}{dz}=(1-\rho)\frac{\rho_1\dfrac{1}{(z)'_u}+\rho_2}{(1-\rho_1 u_1-\rho_2 z)^2}. \quad (3.89)$$

Since u_1 is the root of the denominator (3.81), then

$$\lambda_1 u_1 z(1-u_1)+\lambda_2 u_1 z(1-z)+\mu_1 z(u_1-1)=0. \quad (3.90)$$

From here

$$z = \lambda / \lambda_2 - (\lambda_1 / \lambda_2)u_1 + 1 + \mu_1 / \lambda_2 - \mu_1 / (\lambda_2 u_1). \quad (3.91)$$

Let's find the derivative of *z* with respect to u_1:

$$(z)'_{u_1} = \frac{\mu_1}{\lambda_2 u_1^2} - \frac{\lambda_1}{\lambda_2}. \quad (3.92)$$

Expression (3.89) takes the following form:

$$\frac{dG(1,z)}{dz} = (1-\rho)\frac{\rho_1 \dfrac{\lambda_2 u_1^2}{\mu_1 - \lambda_1 u_1^2} + \rho_2}{(1-\rho_1 u_1 - \rho_2 z)^2}. \quad (3.93)$$

Note that $u_1(1) = 1$, hence

$$L_{s2} = \sum_{j=0}^{\infty} j g_j = \left.\frac{dG(1,z)}{dz}\right|_{z=1} = \left.(1-\rho)\frac{\rho_1 \dfrac{\lambda_2 u_1^2}{\mu_1 - \lambda_1 u_1^2} + \rho_2}{(1-\rho_1 u_1 - \rho_2 z)^2}\right|_{z=1}.$$

Finally: $$L_{s2} = \frac{\rho_2}{1-\rho}\left(1 + \frac{\mu_2}{\mu_1} \cdot \frac{\rho_1}{1-\rho_1}\right). \quad (3.94)$$

Using Little's formula (2.42), we find the average dwell time in the system of non-priority requests:

$$\tau_{s2} = \frac{L_{s2}}{\lambda_2} = \frac{1}{\mu_2(1-\rho)}\left(1 + \frac{\mu_2}{\mu_1} \cdot \frac{\rho_1}{1-\rho_1}\right). \quad (3.95)$$

Figure 3.15 shows graphs of the dependence of the time spent by low-priority requests in the system on the service channel load at different values of the channel load by high-priority requests. For comparison, a graph of the dependence of the time spent by requests in the system for the non-priority M/M/1 system with the same parameters is presented.

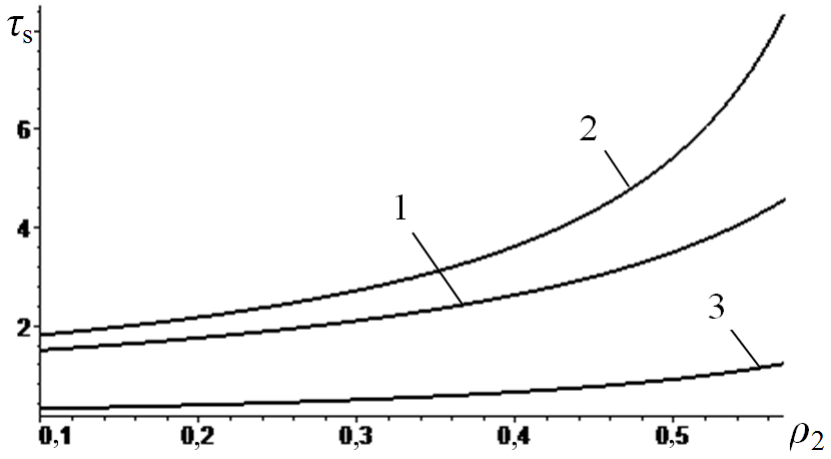


Figure 3.15 – Dependence of the time spent in the system for:
1 – for non-priority requests when $\rho_1 = 0{,}2$;

2 – for non-priority requests when $\rho_1 = 0{,}3$;

3 – for arbitrary requests of the M/M/1 system when $\rho = 0{,}2 + \rho_2$

It is clear that the introduction of a priority stream dramatically increases the time spent in the system by non-priority requests compared to the M/M/1 type of queueing system.

3.8 Queueing systems networks

The current stage of computer technology development is characterized by the mass introduction of computer networks, in which different computers in the network can be used to solve a specific problem. Computer networks are characterized by complex topology, so it is often necessary to pre-model the future network. Through modeling, network characteristics, necessary network equipment, optimal topology, possible reserves for future development are determined. In addition, network modeling ensures the avoidance of costs due to additional network restructuring in the future. As analytical models of computer networks, queue management systems are used.

In general, a queuing system network consists of several individual queuing systems, the outputs of which are connected to the inputs of the queuing system network, as shown in Figure 3.16.

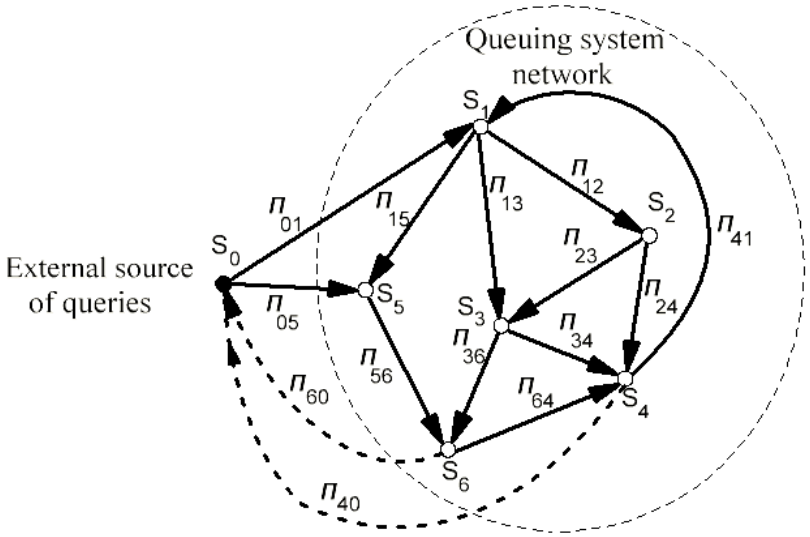


Figure 3.16 – Queuing system network topology

A queuing system network consists of N separate lossless queuing systems S_i. Service requests with probability Π_{0i} arrive at the i-th queuing system of the network ($i = 1,2,...,N$) from an external source S_0. A request that has been serviced

at the *i*-th queuing system with probability Π_{ij} arrives for service at the *j*-th queuing system or leaves the system with probability Π_{i0}.

Let us consider an arbitrary request, which after entering the network sequentially passes through individual queuing systems – $S_1, S_3, S_4, S_1, \ldots$.

Sequence $S_1, S_3, S_4, S_1, \ldots$ forms a certain implementation of the Markov chain, which describes the functioning of a network with a stochastic matrix:

$$[\Pi] = \begin{bmatrix} 0 & \Pi_{01} & \Pi_{02} & \ldots & \Pi_{0N} \\ \Pi_{10} & \Pi_{11} & \Pi_{12} & \ldots & \Pi_{1N} \\ \Pi_{20} & \Pi_{21} & \Pi_{22} & \ldots & \Pi_{2N} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \Pi_{N0} & \Pi_{N1} & \Pi_{N2} & \ldots & \Pi_{NN} \end{bmatrix}, \quad (3.96)$$

Where $0 \le \Pi_{ij} \le 1, i = \overline{1,N}, j = \overline{1,N}$ – the probability of transition from the *i*-th queuing system to the *j*-th. Matrix $\|\Pi\|$ is called the network transmission matrix. The sum of the probabilities of this matrix along the rows is equal to one:

$$\sum_{j=0}^{N} \Pi_{ij} = 1. \quad (3.97)$$

For the network in Figure 3.16

$$[\Pi] = \begin{bmatrix} 0 & \Pi_{01} & 0 & 0 & 0 & \Pi_{05} & 0 \\ 0 & 0 & \Pi_{12} & \Pi_{13} & 0 & \Pi_{15} & 0 \\ 0 & 0 & 0 & \Pi_{23} & \Pi_{24} & 0 & 0 \\ 0 & 0 & 0 & 0 & \Pi_{34} & 0 & \Pi_{36} \\ \Pi_{40} & \Pi_{41} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \Pi_{56} \\ \Pi_{60} & 0 & 0 & 0 & \Pi_{64} & 0 & 0 \end{bmatrix},$$

besides

$$\Pi_{01} + \Pi_{05} = 1, \Pi_{12} + \Pi_{13} + \Pi_{15} = 1, \Pi_{23} + \Pi_{34} = 1, \Pi_{40} + \Pi_{41} = 1, \Pi_{56} = 1, \Pi_{60} + \Pi_{64} = 1.$$

Queuing system networks are very difficult to study. They are usually studied in steady-state conditions and with Poisson event flows.

Let λ_i – the total intensity of the flow of requests to the *i*-th queuing system, i.e. the average number of requests that arrive in this system per unit of time in a stationary mode, and μ_i – the average intensity of the flow of service requests in the *i*-th queuing system of network. The intensity of the flow at the exit from this system is also equal to λ_i (lossless system). Probability Π_{ji} that a request that was

served at the *j*-th queuing system finds its way into system *i* does not depend on the previous path of this request and the state of the network. Therefore, for the stationary regime, based on the total probability formula (1.4), we can write:

$$\lambda_i = \sum_{j=0}^{N} \lambda_j \Pi_{ji}, i = 0,1,2,...,N. \quad (3.98)$$

In expression (3.98) $\lambda_0 = \sum_{j=1}^{N} \lambda_{0j}$ – total intensity of the query source.

Let's write down $\lambda_0, \lambda_1, \lambda_2, ..., \lambda_N$ as a column matrix $[\Lambda] = \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ ... \\ \lambda_N \end{bmatrix}$, then system (3.98) can be written in matrix form:

$$[\Lambda] = [\Pi]^T \cdot [\Lambda], \quad (3.99)$$

Where $[\Pi]^T$ – matrix transposed to matrix $[\Pi]$.

From (3.99) we find

$$\left([\Pi]^T - I\right) \cdot [\Lambda] = 0, \quad (3.100)$$

where *I* is the identity matrix of order $N+1$.

Expressions (3.98) and (3.100) describe a system of $N+1$ linear homogeneous equations with respect to the intensities $\lambda_0, \lambda_1, \lambda_2, ..., \lambda_N$, and the expression $\left(\|\Pi\|^T - I\right)$ is the system matrix of this system of linear equations. For this system to have a nontrivial solution $\lambda_i \neq 0$, it is necessary that the determinant of its system matrix be equal to zero:

$$\left|[\Pi]^T - I\right| = \begin{vmatrix} -1 & \Pi_{10} & \Pi_{20} & ... & \Pi_{N0} \\ \Pi_{01} & \Pi_{11} - 1 & \Pi_{21} & ... & \Pi_{N1} \\ \Pi_{02} & \Pi_{12} & \Pi_{22} - 1 & ... & \Pi_{N2} \\ ... & ... & ... & ... & ... \\ \Pi_{0N} & \Pi_{1N} & \Pi_{2N} & ... & \Pi_{NN} - 1 \end{vmatrix} = 0. \quad (3.101)$$

If we add the sum of the corresponding elements of the other rows to each element of the last row of the determinant (3.101), then this row will become zero:

$$\Pi_{Nj} + \Pi_{0j} + \Pi_{1j} + ... + \Pi_{jj} - 1 + ... + \Pi_{N-1j} = \Pi_{0j} + \Pi_{1j} + ... + \Pi_{jj} + ... + \Pi_{N-1j} + \Pi_{Nj} - 1 = 0.$$

Therefore, the determinant (3.101) will be equal to zero, and the system (3.98) has

many solutions, although all λ_i, $i=1,2,...,N$ can be expressed through λ_0: $\lambda_i = \alpha_i \lambda_0$, $i=1,2,...,N$ by solving the system of inhomogeneous equations:

$$\begin{aligned}
\Pi_{10}\lambda_1 + \quad & \Pi_{20}\lambda_2 + \Pi_{30}\lambda_3 + ... \quad + \Pi_{N0}\lambda_N = \quad \lambda_0, \\
(\Pi_{11}-1)\lambda_1 + \quad & \Pi_{21}\lambda_2 + \Pi_{31}\lambda_3 + ... \quad + \Pi_{N1}\lambda_N = -\lambda_0\Pi_{01}, \\
\Pi_{12}\lambda_1 + & (\Pi_{22}-1)\lambda_2 + \Pi_{32}\lambda_3 + ... \quad + \Pi_{N2}\lambda_N = -\lambda_0\Pi_{02}, \qquad (3.102) \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
\Pi_{1N}\lambda_1 + \quad & \Pi_{2N}\lambda_2 + \Pi_{3N}\lambda_3 + ... + (\Pi_{NN}-1)\lambda_N = -\lambda_0\Pi_{0N}.
\end{aligned}$$

Consider the network shown in Figure 3.17 [20].

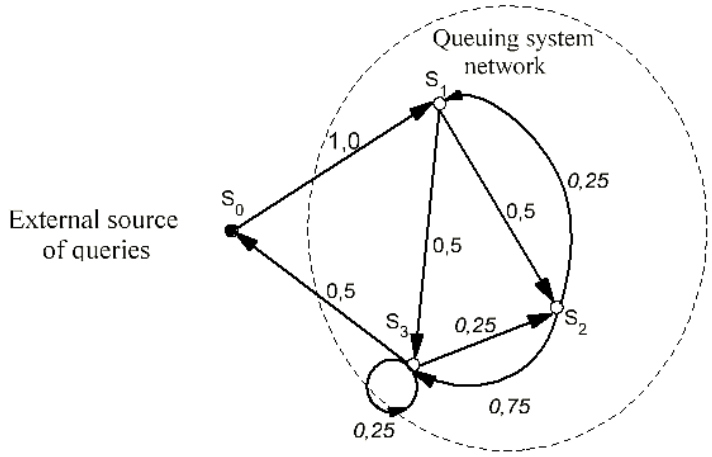


Figure 3.17 – Example queuing system network

It is described by the gear matrix

$$[\Pi] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0,5 & 0,5 \\ 0 & 0,25 & 0 & 0,75 \\ 0,5 & 0 & 0,25 & 0,25 \end{bmatrix}.$$

It corresponds to the following system matrix:

$$[\Pi]^T - I = \begin{bmatrix} -1 & 0 & 0 & 0,5 \\ 1 & -1 & 0,25 & 0 \\ 0 & 0,5 & -1 & 0,25 \\ 0 & 0,5 & 0,75 & -0,75 \end{bmatrix},$$

which corresponds to a system of equations of the form (3.102):

$$\begin{aligned}
& 0,5\lambda_3 = \lambda_0; \\
& -\lambda_1 + 0,25\lambda_2 = -\lambda_0; \\
& 0,5\lambda_1 - \lambda_2 + 0,25\lambda_3 = 0; \\
& 0,5\lambda_1 + 0,75\lambda_2 - 0,75\lambda_3 = 0.
\end{aligned}$$

Let us write its coefficients in the unknowns in matrix form and perform the transformation using the Gauss method. Let us denote the row numbers with Roman numerals:

$$\left.\begin{array}{cccc} 0 & 0 & 0,5 & 1 \\ -1 & 0,25 & 0 & -1 \\ 0,5 & -1 & 0,25 & 0 \\ 0,5 & 0,75 & -0,75 & 0 \end{array}\right\| \Rightarrow II = II + (III + IV) \Rightarrow \left.\begin{array}{cccc} 0 & 0 & 0,5 & 1 \\ 0 & 0 & -0,5 & -1 \\ 0,5 & -1 & 0,25 & 0 \\ 0,5 & 0,75 & -0,75 & 0 \end{array}\right\| \Rightarrow$$

$$\Rightarrow \left.\begin{array}{ccc} 0 & 0 & 0,5 \\ 0,5 & -1 & 0,25 \\ 0,5 & 0,75 & -0,75 \end{array}\right\| \begin{array}{c} 1 \\ 0 \\ 0 \end{array} \Rightarrow \left\{\begin{array}{l} II = II - 0,5I \\ III = III + 0,5I \end{array}\right\} = \left.\begin{array}{ccc} 0 & 0 & 0,5 \\ 0,5 & -1 & 0 \\ 0,5 & 0,75 & 0 \end{array}\right\| \begin{array}{c} 1 \\ -0,5 \\ 1,5 \end{array} \Rightarrow$$

$$\Rightarrow II = II - III \Rightarrow \left.\begin{array}{ccc} 0 & 0 & 0,5 \\ 0 & -1,75 & 0 \\ 0,5 & 0,75 & 0 \end{array}\right\| \begin{array}{c} 1 \\ -2 \\ 1,5 \end{array} \Rightarrow \{III = III + II / 1,75 * 0,75\} =$$

$$= \left.\begin{array}{ccc} 0 & 0 & 0,5 \\ 0 & -1,75 & 0 \\ 0,5 & 0 & 0 \end{array}\right\| \begin{array}{c} 1 \\ -2 \\ 0,643 \end{array}$$

Finally:

$$\lambda_1 = \frac{0,643}{0,5}\lambda_0 = 1,285\lambda_0;\ \lambda_2 = \frac{-2}{-1,75}\lambda_0 = 1,142\lambda_0;\ \lambda_3 = \frac{1}{0,5}\lambda_0 = 2\lambda_0$$

From this example it is clear that the system of linear equations (3.102), which describes the queuing system network, has a unique solution with respect to $\lambda_0 - \lambda_i = \alpha_i\lambda_0,\ i = 1,2,...,\ N$.

For a stationary regime to arise in a network of queuing systems, it is necessary that each queuing system that is part of the network operates in a stationary regime. Since the queuing systems that are part of the network are lossless systems, in order to create a stationary regime, it is necessary to ensure that the following condition is met: $\lambda_i < \mu_i$ for each queuing system. It can be written as $\alpha_i\lambda_0 < \mu$ or $\lambda_0 < \frac{\mu_i}{\alpha_i}, i = 1,2,...,N$. Therefore, for a stationary regime to arise in the queuing system network, the following condition must be met:

$$\lambda_0 < \min_{i=1,2,..,N} \frac{\mu_i}{\alpha_i}. \quad (3.103)$$

When calculating the characteristics of queuing system networks, it is advisable to know the time a request spends in the network. If the average time a request spends in each queuing system network is known (τ_{si}), then the following formula is used [20]

$$T_m = \sum_{i=1}^{N} \alpha_i \tau_{si}. \quad (3.104)$$

Expression (3.104) means that a single request can be served multiple times on the same queuing system of network.

4 APPLICATION OF THE THEORY OF QUEUING SYSTEMS

The material in this section is based on the author's experience in developing queuing models when designing computer systems for various purposes.

4.1 Research and selection of parameters of the module for receiving pulse-number information

In modern automatic and automated control systems, a special input-output processor is usually used to organize a rational connection of the control computing complex with the control object. One of the requirements for the input-output processor is the minimum cost of the central processor (CPU) machine time for information exchange. If the control system includes sensors of active, pulse-number information, where each pulse contains information about a certain increase in the measured value over the corresponding communication channel, the input-output processor must include a special module for receiving pulse-number information (hereinafter referred to as the MRPNI). This module, which includes pulse counters for each channel, allows you to pre-process the information received about the state of the control object, and therefore, reduce the load on the CPU. This is especially important at high frequencies of pulse-number signals.

You can download information from the MRPNI in two modes.

1. On a CPU command, which reads data from one or more MRPNI counters; moreover, after reading the information, the counters are reset.

2. By command of MRPNI in case of filling of counters by a certain value or their overflow; in addition, the processor receives an interrupt signal, after which it is necessary to accept and process the next portion of information. Interrupt and reset information from counters using the processor immediately after the interrupt signal arrives. If the interrupt signal arrives after the counter has been filled to a certain value, the processor may stop retrieving information for a while, in particular if the execution of the current program cannot be interrupted at that moment.

The process of removing information by command of the central processor usually occurs in the control cycle; the process of removing information by command of MRPNI, that is, by requests from counters, is carried out asynchronously with respect to the control cycle, and therefore it is desirable that there be as few of them as possible. Let us denote by F the total frequency of interrupts coming from MRPNI counters. Then F can be chosen as a criterion for the efficiency of the functioning of the module for receiving number-pulse information.

The study of the functioning of the MRPNI is based on the following:

a) the interrupt signal arrives at the CPU when the *i*-th counter is filled ($i=\overline{1,n}$) on q_i pulses; the resolution of the *i*-th counter is m_i, and its capacity is $W_i = 2^{m_i} - 1$ pulses. The probability that the processor will immediately remove information from the *i*-th counter in the event of an interrupt signal a_i;

b) the flow of pulses arriving on each channel is random and is described by an exponential distribution function with parameter f_i, where f_i – average frequency of pulses arriving on the *i*-th channel;

c) the process of retrieving information at the processor's requests in the control cycle (before the interrupt signal arrives) is random, described by an exponential distribution function with parameter μ_i, where μ_i – average polling frequency of the *i*-th counter of number-pulse information in the control cycle. Information removal from the counter after it issues an interrupt signal with a delay of a random time ξ is carried out with probability $1-a_i$ (this probability is small). Random time ξ, which has an average value τ_i, subject to an exponential distribution with parameter $\nu_i = 1/\tau_i$.

Let us first investigate the process of functioning of one *i*-th counter receiving number-pulse information. The graph of a random process of states and transitions for it is shown in Figure 4.1, where under the recording {*j*} ($j=\overline{0,w_i}$) understand the state of the counter when it contains *j* pulses, the stationary probability of this is $p_j\left(j=\overline{0,W_i}\right)$.

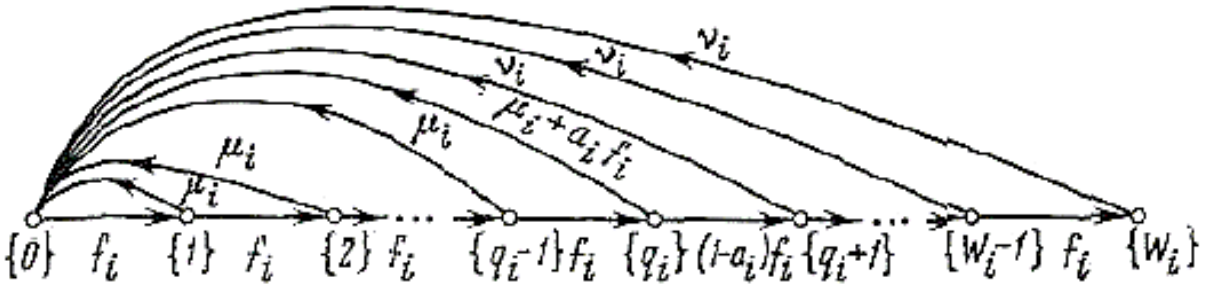


Figure 4.1 – State and transition graph for the *i*-th counter

The system of stationary equations corresponding to this graph has the following form:

$$
\begin{aligned}
&-f_i p_0 + \mu_i \sum_{j=1}^{q_i-1} p_j + (\mu_i + a_i f_i) p_{q_i} + \nu_i \sum_{j=q_i+1}^{W_i} p_j = 0,\\
&-(f_i + \mu_i) p_j + f_i p_{j-1} = 0,\ 0 < j \le q_i,\\
&-(f_i + \nu_i) p_{q_1+1} + (1-a_i) f_i p_{q_i} = 0,\\
&-(f_i + \nu_i) p_j + f_i p_{j-1} = 0,\ q_i + 1 < j < W_i,\\
&-\nu_i p_{W_i} + f_i p_{W_i-1} = 0.
\end{aligned}
\qquad (4.1)
$$

The normalization condition is

$$
p_0 + \sum_{j=1}^{W_i} p_i = 1. \qquad (4.2)
$$

According to the second equation of system (4.1), we consistently find:

$$
p_1 = \left(\frac{f_i}{f_i + \mu_i} \right) p_0,\ p_j = \left(\frac{f_i}{f_i + \mu_i} \right)^j p_0,\ j \le q_i.
$$

By substituting p_{q_i} into the third equation of system (4.1), we obtain:

$$
p_{q_i+1} = \left(\frac{f_i}{f_i + \mu_i} \right)^{q_i} \frac{(1-a_i)}{f_i + \mu_i} p_0 .
$$

From the fourth and fifth equations of system (4.1), we successively find:

$$
p_j = \frac{(1-a_i) f_i^{\,j}}{(f_i + \mu_i)^{q_i} (f_i + \nu_i)^{j-q_i}} p_0, 1 + q_i < j < W_i,
$$

$$
p_{w_I} = \frac{(1-a_i) f_i^{\,w_I}}{\nu_i (f_i + \mu_i)^{q_i} (f_i + \nu_i)^{w_I - q_i - 1}} p_0.
$$

Substituting these probabilities into the normalization condition (4.2), after the appropriate transformations we find the probability p_0:

$$
p_0 = \left\{ 1 + \frac{f_i}{\mu_i} \left[1 - \left(\frac{f_i}{f_i - \mu_i} \right)^{q_i} \right] + \frac{(1-a_i) f_i^{\,q_i+1}}{(f_i + \mu_i)^{q_i} \nu_i} \left[1 - \left(\frac{f_i}{f_i - \mu_i} \right)^{W_i - q_i - 1} \right] + \frac{(1-a_i) f_i^{\,w_i}}{\nu_i (f_i + \mu_i)^{q_i} (f_i + \nu_i)^{w_i - q_i - 1}} \right\}^{-1}.
$$

Probability of information loss p_{W_i} by the *i*-th counter:

$$
p_{W_i} = \frac{(1-a_i) f_i^{\,w_i}}{\nu_i (f_i + \mu_i)^{q_i} (f_i + \nu_i)^{w_i - q_i - 1}} .
$$

According to the last equation, you can choose the counter resolution and its polling frequency so that the probability of information loss is within the specified limits.

An important characteristic of the operation of the *i*-th counter is the average time T_i between issuing an interrupt signal to the CPU. To determine T_i we will

slightly change the graph (Figure 4.1) by introducing a new state $\{r_i\}$ and denoting the intensity of the transition from state $\{r_i\}$ to state {0} by η_i (Figure 4.2).

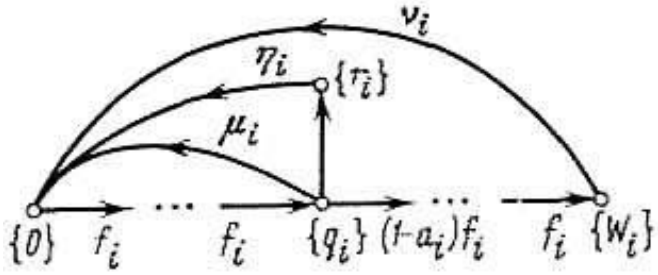


Figure 4.2 – Modified state graph of the system

The graph in Figure 4.2 will be equivalent to the graph in Figure 4.1 if $\eta_i = \infty$.

The state $\{r_i\}$ can occur only upon a counter interrupt signal. Let us denote by T_i^* the average time between interruptions, taking into account the input state $\{r_i\}$. When $\eta_i = \infty$ we have: $T_i^* \to T_i$. Time T_i will consist of the average time L_i – the movement of the random process through the states $\{0\},\{1\},...,\{q_i\}$, the average time N_i – the movement of the random process through the states $\{q_i+1\},\{q_i+2\},...,\{W_i\}$ and the average time R_i the time the random process is in the state $\{r\}_i$, that is $T_i^* = L_i^* + N_i^* + R_i^*$. Let us transform the graph shown in Figure 4.2 into the graph shown in Figure 4.3, that is, we isolate the states $\{r_i\}$ and $\{q_i+1\}$ and make them closed.

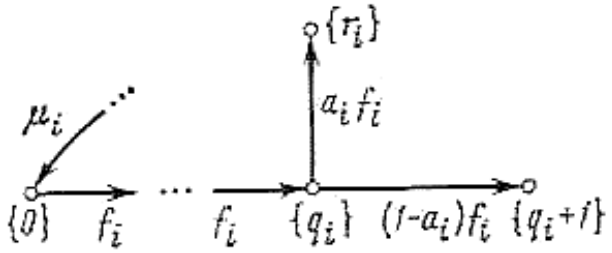


Figure 4.3 – State graph of a system with closed states $\{r_i\}$ and $\{q_i+1\}$

The boundary between the states $\{r_i\}$ and $\{q_i+1\}$ is the state $\{0\}$. The initial conditions are $p_0(0)=1, p_i(0)=0, i\neq 0$.

Probability density U(t) of being in a subset$[\{0\},\{1\},...,\{q_i\}]$ equal to $\frac{d}{dt}p_{r_i}(t)+\frac{d}{dt}p_{q_i+1}(t)$, and $\frac{d}{dt}p_{r_i}(t)+\frac{d}{dt}p_{q_i+1}(t)=f_i p_{q_i}(t)$, that is $U(t)=f_i p_{q_i}(t)$.

To determine $p_{q_i}(t)$ it is necessary to compose a system of differential equations:

$$\begin{aligned}\frac{d}{dt}p_0(t) &= -f_i p_0(t) + \mu_i \sum_{i=1}^{q_i} p_i(t),\\ \frac{d}{dt}p_j(t) &= -(f_i+\mu_i)p_j(t) + f_i p_{j-1}(t),\ j=1,2,...,q_i,\end{aligned} \tag{4.3}$$

with initial conditions $p_0(0)=1, p_i(0)=0, i\neq 0$.

Let us find the solution of system (4.3) using Laplace transforms:

$$p_{q_i}(t) = \frac{(\mu_i+s)\left(\frac{f_i}{f_i+\mu_i+s}\right)^{q_i}}{s(f_i+\mu_i+s)+f_i\mu_i\left(\frac{f_i}{f_i+\mu_i+s}\right)^{q_i}}. \tag{4.4}$$

Applying the known relations, we obtain:

$$L_i^* = \int_0^\infty tU(t)dt = -\frac{d}{ds}U(s)\Big|_{s=0} = -f_i\frac{d}{ds}p_{q_i}(s)\Big|_{s=0} = \frac{(f_i+\mu_i)^{q_i+1}-f_i^{q_i+1}}{\mu_i f_i^{q_i+1}}. \tag{4.5}$$

By performing similar actions, we get:

$$N_i^* = \frac{1-a_i}{\nu_i},\ R_i^* = \frac{a_i}{\eta_i}. \tag{4.6}$$

So,

$$T_i = \lim_{\eta_i\to 0} T_i^* = \frac{\nu_i\left[\left(1+\frac{\mu_i}{\lambda_i}\right)^{q_i+1}-1\right]+\mu_i(1-a_i)}{\mu_i\nu_i}. \tag{4.7}$$

Hence, the average frequency of interrupt signals from the *i*-th counter is

$$F_i = \frac{\mu_i\nu_i}{\nu_i\left[\left(1+\frac{\mu_i}{\lambda_i}\right)^{q_i+1}-1\right]+\mu_i(1-a_i)}. \tag{4.8}$$

The total frequency of interrupt signals from the entire module for receiving pulse-number information is

$$F = \sum_{i=1}^{n}\frac{\mu_i\nu_i}{\nu_i\left[\left(1+\frac{\mu_i}{\lambda_i}\right)^{q_i+1}-1\right]+\mu_i(1-a_i)}. \tag{4.9}$$

Using expression (4.9), it is possible to choose the polling frequencies of the counters for receiving number-pulse information in such a way that, with the same CPU time consumption for exchange, the impact on its operation by the MRPNI

through interrupts is minimal. To do this, it is necessary to minimize the function F in terms of variables $\mu_i, i=\overline{1,n}$ provided that $\sum_{i=1}^{n}\mu_i = const$. This problem can be solved using well-known optimization methods, such as the fastest gradient descent method [19].

In cases where the interrupt signal is given after the MRPNI counters overflow, expression (4.11) will take the following form:

$$F = \sum_{i=1}^{n} \frac{\mu_i}{\left(1+\frac{\mu_i}{\lambda_i}\right)^{2^{m_i}} - 1}. \qquad (4.10)$$

In this case, the solution of two optimization tasks is ensured:

1. With constant bit sizes of individual counters, choose the following polling frequencies μ_i at $\sum_{i=1}^{n}\mu_i = const$ so that the total interruption frequency F is minimal.

2. With constant polling frequencies of individual counters, choose the following bit sizes of individual counters m_i (i.e. the number of flip-flops and other electronic circuits) so that with a total constant number of electronic circuits ($\sum m_i = const$) the total interruption frequency F was minimal.

Example. Let $n=3$, $f_1=1\,000$ Hz, $f_2=5\,000$ Hz, $f_3=8\,000$ Hz. Resolution of individual counters – $m_1=m_2=m_3=8$, $\sum\mu_i=300$ Hz. Interruption is carried out after overflowing of individual MRPNI counters. It is necessary to optimally (with respect to the minimum F) choose the polling frequencies of individual counters.

Using the numerical method of the fastest descent, implemented on a PC, we obtain: $F_{\min}=1{,}92$ Hz at $\mu_1=50{,}03$ Hz, $\mu_2=96{,}25$ Hz, $\mu_3=153{,}76$ Hz. It is worth noting that when $\mu_1=\mu_2=\mu_3=100$ Hz $F=4{,}971$, i.e. the optimal choice of polling frequencies made it possible to reduce the total frequency of interruptions by more than two times compared to a uniform distribution of polling frequencies.

4.2 Probabilistic modelling and assessment of the quality of functioning of information and management systems

Management information systems (hereinafter referred to as MIS), which belong to real-time systems, occupy an important place in modern production and social life. They are complex to implement, are constantly being improved, and are used to solve increasingly complex tasks. The development of MIS is a rather

complex process that requires designers to take into account many factors. To ensure high quality of the MIS at the design stage, it is necessary to use modelling, which will ensure verification of basic design solutions, and will enable the identification of bottlenecks in the system. The most important component of the MIS is software. It provide real-time solution of a wide range of tasks for managing individual subsystems of the MIS, which are characterized by frequency, duration of work, priority, volume of data input and output. Launching of certain tasks is carried out by the MIS dispatching system according to commands periodically prepared by the planner taking into account the functioning of the IMS. To study the dispatching processes in the MIS, it is necessary to use appropriate models that provide modelling of the processing of processes by the system depending on the number of partial tasks, their frequency and taking into account the specified algorithms. Several approaches to modelling the MIS are known, but at the stage of preliminary design it is advisable to use analytical models that take into account the stochasticity of the MIS functioning and allow, albeit relatively, to assess important characteristics of the system being developed.

The simple form of planning of the real time tasks execution is synchronous, when slot between sequential starts of the same task are equal to the selected slot. The flexibility and efficiency of such planning is greatly improved with the introduction of priorities, as well as with the ability to solve the background tasks when the computer system of MIS is free from solving real-time tasks. A variety of characteristics can serve as a criterion of the effectiveness of the dispatching operation of the system. For example the delay of start time of individual tasks on the planned time: the meantime, the probability that the time delay exceeds a certain fixed value and so on. The knowledge of these characteristics can be used for synthesis of the system of planning of MIS software.

Let the entire time interval is divided into separate intervals Δ using a timer. For each interval, a certain number of high-priority and low-priority tasks are planned to be performed in accordance with the MIS operation algorithm. The time free in the interval from solving real-time tasks is used to solve background tasks (for example, test control of equipment). Interrupting high-priority tasks is prohibited, while low-priority real-time tasks is allowed.

Interrupted low-priority tasks complete their service after processing high-priority tasks. If a low-priority task is interrupted in the *i*-th interval, then its continuation is added to the tasks in the $(i+1)-$th time interval. Each task is scheduled to run with a certain period T_i. It is possible to isolate the main cycle of calculations *T*, which is the least common multiple T_i, $i=1,2,...,N$, where *N* is the

total number of individual tasks solved by the MIS software. In general, the execution time of each task can be random, and therefore it is important to know the time of time delays during their solution. This is especially important for low-priority tasks, the start of which can be shifted by high-priority tasks.

To consider the process of solving high-priority problems, let us assume the following. Let ξ_{ij} – execution time of the task of the *i*-th priority in the *j*-th time interval; $\Phi_j = \sum_{i \in I_j} \xi_{ij}, i = 1,2,...,n,n+1,n+2,$ is the time to solve high-priority tasks in the *j*-th time interval, where I_j – scheduled tasks for the *j*-th interval, and $n = T_h / \Delta, T_h$ – the basic cycle of execution of high-priority tasks. Let φ_j – the average time for solving priority tasks in the *j*-th interval. The total service flow in each interval is formed as the sum of several service flows for each task and can be considered the simplest, with the parameter $\mu_j = 1/\varphi_j$. The general service time distribution function is written as a superposition of service flows for each interval – $B(t) = 1 - \frac{1}{m}\sum_{j=1}^{m} e^{-\mu_j t}$, that is, it is described by a hyperexponential distribution. The input flow for such a system is regular, but it can be approximated by an Erlang flow if the coefficient of variation approaches zero. The distribution function of such a flow, in general, looks like this: $A(t) = 1 - \sum_{i=1}^{k} \frac{(\lambda t)^{i-1}}{(i-1)!} e^{-\lambda t}$. So, the entire system reduces to a single-line queueing system of the form E/H/1. Let's examine it.

Let $\overline{\lambda} = \frac{\lambda}{k}$ and $\overline{\mu} = \left[\sum_{i=1}^{m} \frac{1}{m\mu_i}\right]^{-1}$ –average intensity of flows described according to distributions $A(t)$ and $B(t)$, and $\alpha(s)$ and $\beta(s)$ – the Laplace–Stiehljes transform of these distributions, that is, $\alpha(s) = \left(\frac{\lambda}{\lambda + s}\right)^k$ and $\beta(s) = \frac{1}{m}\sum_{i=1}^{m} \frac{\mu_i}{\mu_i + s}$.

If $\overline{\mu} > \overline{\lambda}$, then for the steady state of this system there exists a waiting time distribution function $F(t)$, to find which we can compose the Lindley integral equation [17]:

$$F(t)=\begin{cases}\int\limits_{-\infty}^{t}F(t-x)dC(x), & t\geq 0,\\ 0 & , \ t<0,\end{cases} \tag{4.11}$$

where $C(t)=\int\limits_{0}^{\infty}B(x+t)dA(t)$.

The solution can be found by factoring the Laplace–Stiehljes transform of the kernel of the integral equation. The corresponding factorization equation will have the following form:

$$\gamma(s)=\frac{K_{+}(s)}{K_{-}(s)}=\alpha(-s)\beta(s)-1=\frac{1}{m}\left(\frac{\lambda}{\lambda-s}\right)^{k}\sum_{i=1}^{m}\frac{\mu_i}{\mu_i+s}-1=0. \tag{4.12}$$

Note that the function

$$\beta(s)=\frac{\dfrac{1}{m}\sum\limits_{i=1}^{m}\mu_i\dfrac{\prod\limits_{j=1}^{m}(\mu_j+s)}{\mu_i+s}}{\prod\limits_{i=1}^{m}(\mu_i+s)}=\frac{P_{m-1}(s)}{Q_m(s)} \tag{4.13}$$

is the ratio of two polynomials, and the degree of the numerator is less than the degree of the denominator. Therefore, according to [17], we can write that by the Laplace–Stiehljes transform of $F(t)$ there are

$$\varphi(s)=\frac{Q_m(s)}{Q_m(0)\prod\limits_{i=1}^{m}\left(1-\dfrac{s}{q_i}\right)}, \tag{4.14}$$

where $q_i\,(i=1,2,...,m)$ – roots of the equation

$$\gamma(s)=0\Rightarrow\frac{1}{m}\left(\frac{\lambda}{\lambda-s}\right)^{k}\sum_{i=1}^{m}\frac{\mu_i}{\mu_i+s}-1=0, \tag{4.15}$$

which are contained in the left half-plane $\operatorname{Re}s<0$. Using (4.13), we obtain in the left half-plane $\operatorname{Re}s<0$.

$$\varphi(s)=\frac{Q_m(s)}{\prod\limits_{i=1}^{m}\mu_i\left(1-\dfrac{s}{q_i}\right)}. \tag{4.16}$$

Let $s=\lambda(1-z)$, then equation (4.15) will be as follows:

$$\frac{1}{m}\sum_{i=1}^{m}\frac{\mu_i}{\mu_i+\lambda(1-z)}-z^{k}=0. \tag{4.17}$$

Equation (4.17) has $m+k$ zeros. The *k* zeros of this equation lie within $|z|=1$, taking into account their multiplicity. To prove this, we find the values of the moduli of both applications (4.19) on the boundary of the domain $|z|=1-\varepsilon,\ \varepsilon\to+0$. Then

$$|-z|^k=(1-\varepsilon)^k. \tag{4.18}$$

If we expand the right-hand side of (4.19) using Taylor's formula and discard the terms of higher order of smallness compared to the first, we obtain $|-z|^k=1-k\varepsilon$. Similarly $\left|\frac{1}{m}\sum_{i=1}^{m}\frac{\mu_i}{\mu_i+\lambda(1-z)}\right|=1-\lambda\varepsilon\sum_{i=1}^{m}\frac{1}{m\mu_i}=1-\varepsilon k\frac{\overline{\lambda}}{\overline{\mu}}$.

Because $\overline{\mu}>\overline{\lambda}$, then $\varepsilon k>\varepsilon\frac{k\overline{\lambda}}{\overline{\mu}}\Rightarrow k>\lambda\sum_{i=1}^{m}\frac{1}{m\mu_i}$, that is on the border of the region $|z|=1-\varepsilon,\varepsilon\to+0$ $\left|-z^k\right|>\left|\frac{1}{m}\sum_{i=1}^{m}\frac{\mu_i}{\mu_i+\lambda(1-z)}\right|$. Then, according to Rouchet's theorem, inside the domain $|z|$ z^k and $\left(\frac{1}{m}\sum_{i=1}^{m}\frac{\mu_i}{\mu_i+\lambda(1-z)}-z^k\right)$ have the same number of zeros – *k*. The remaining *m* zeros of equation (4.17) are outside the domain $|z|=1$. Let us denote them as $z_i, i=1,2,...,m.$ Function $\gamma_1(s)=\frac{1}{m}\sum_{i=1}^{m}\frac{\mu_i}{\mu_i+\lambda(1-z)}-z^k$ is monomorphic with poles at the points $1+\frac{\mu_i}{\lambda}, i=1,2,...,m$, between which there is at least one zero. Since there are m poles in total and in the region $|z|>1-\frac{\mu_m}{\lambda}$ there are no zeros, then

$$1<z_1<1+\frac{\mu_1}{\lambda}<z_2<...<1+\frac{\mu_{k-1}}{\lambda}<z_k<1+\frac{\mu_k}{\lambda}<.....<z_m<1+\frac{\mu_m}{\lambda}. \tag{4.19}$$

Let's express q_i through z_i: $q_i=\lambda(1-z_i),\ i=1,2,...,m$. Because $|z_i|>1$, then q_i contained in the left half-plane $\mathrm{Re}\,s<0$.

Finally we get:

$$\varphi(s)=\lambda^m\prod_{i=1}^{m}\frac{(\mu_i+s)(1-z_i)}{\mu_i+[\lambda(1-z_i)-s]}. \tag{4.20}$$

Using the inverse Laplace–Stieltjes transform from (4.20), we obtain an expression for the distribution function of the delay time for servicing high-priority tasks:

$$F(t)=1-\sum_{i=1}^{m}\frac{\prod_{i=1}^{m}(1-z_i)(\mu_i+\lambda-\lambda z_i)}{(z_i-1)\prod_{l=1}^{m}\mu_l\sum_{r=1}^{m}\frac{\prod_{k=1}^{m}(z_i-z_k)}{(z_i-z_r)}}e^{-\lambda(z_i-1)t}. \quad (4.21)$$

Average delay time regarding maintenance high-priority tasks are calculated as follows:

$$\bar{t}_d=-\varphi'(s)\big|_{s=0}=\sum_{i=1}^{m}\left[\frac{1}{\lambda(z_i-1)}-\frac{1}{\mu_i}\right].$$

Let's define the time dispersion delays regarding maintenance high-priority tasks:

$$D=\sigma_d^2=m^2-\bar{t}_d^2, \quad (4.22)$$

Where σ_d – root mean square deviation of the delay time, m_2 – the second initial moment of delay time. Having calculated m_2 and substituting into (4.22) $\bar{t}_d^2$ and m_2 after transformation we get:

$$D=\sigma_d^2=\sum_{i=1}^{m}\left[\frac{1}{\lambda^2(z_i-1)^2}-\frac{1}{\mu_i^2}\right].$$

The probability that time to solve high-priority tasks ξ will exceed the time interval Δ, is determined by the distribution law:

$$P\{\xi>\Delta\}=1-B(\Delta).$$

Real-time conditions have a greater impact on the execution of low-priority tasks for which service interruption is allowed. Let i be the priority level of the analysed low-priority task, and $T_i=N_i\Delta$ is its execution period. Let us denote by u_j the time required to solve this problem in the j-th interval, and through ω_j – the total time for solving all high-priority tasks in this interval, including the time to finish servicing tasks from the $(j-1)-$th interval. Then the time remaining in the *j*-th period is T_i to perform the considered task, is determined by the expression $v_j^i=\max\{0,T_i-w_j\}$, more over v_j^i is a random variable. It can be either more or less than the time required to solve the problem u_j. If $u_j>v_j^i$, then the considered low-priority task will wait for the processor to be freed from solving high-priority tasks. Each interval v_j^i corresponds to the time interval u_j. Thus, the relationship between these random variables is similar to the process of servicing requests in a single-line queuing system of type G/G/1, moreover, the sequence of variables v_j^i

describes the incoming request flow, and u_j – service time. To determine the relationship between the delay time Z during execution low-priority tasks and waiting time η in equivalent single-line queuing system we use the following: if the task are starting to be solved in the planned time interval Δ, then $\eta = z$, otherwise $\eta = z - \Delta \cdot \text{int}(z/\Delta)$, where $\text{int}(x)$ – whole part of an expression x. Therefore, $P\{z > t\} = P^*\{\eta > t^*\} = 1 - F(t - \Delta \cdot \text{int}(t/\Delta)) = 1 - W(t - \Delta \cdot \text{int}(t/\Delta))$, where $F(t)$ – waiting time distribution function for equivalent queuing system, and $W(t)$ – distribution function of delay time Z for low-priority tasks.

Approximating the input stream and the flow service distributions belonging to the family general distributions Erlang. Transformations Laplace – Stiltjes from their distribution functions look like this:

$$f(s) = \prod_{i=1}^{k} \sum_{j=1}^{n_i} \frac{a_{ij}\lambda_{ij}}{s + \lambda_{ij}}, \ \sum_{j=1}^{n_i} a_{ij} = 1, \tag{4.23}$$

they are the ratio of two polynomials. Exponential, hyperexponential and Erlang distributions are special cases of distribution (4.23). Any distribution of the duration of intervals between adjacent events can be approximated with any accuracy by the general Erlang distribution [21], it is only necessary select the appropriate parameters $a_{ij}, \lambda_{IJ}, n_i, k$. It is desirable that the values n_i and k were small.

Consider a single-line queueing system with waiting, which models the service process low-priority tasks, the incoming flow of requests and the service time and for which the generalized Erlang distributions with Laplace-Stiehljes transforms of the distribution functions are given, respectively:

$$\alpha(s) = \prod_{i=1}^{k} \sum_{g=1}^{l_i} \frac{a_{ig}\lambda_{ig}}{\lambda_{ig} + s}, \ \sum_{g=1}^{l_i} a_{ig} = 1, \ i = 1,2,...,k, \tag{4.24}$$

$$\beta(s) = \prod_{j=1}^{n} \sum_{q=1}^{c_j} \frac{b_{jq}\mu_{jq}}{\mu_{jq} + s}, \ \sum_{q=1}^{c_j} b_{jq} = 1, \ j = 1,2,...,n. \tag{4.25}$$

Expression for average intensity of service requests low-priority tasks $\bar{\lambda}$ is calculated as follows:

$$\bar{\lambda} = -\alpha'(0)^{-1} = \left[\sum_{i=1}^{k} \sum_{g=1}^{l_i} \frac{a_{ig}}{\lambda_{ig}} \right]^{-1}, \tag{4.26}$$

and the expression for the average intensity of the service process is low-priority tasks $\bar{\mu}$ will be like this:

$$\overline{\mu} = -\beta'(0)^{-1} = \left[\sum_{i=1}^{n} \sum_{q=1}^{c_i} \frac{b_{iq}}{\mu_{iq}} \right]^{-1}. \qquad (4.27)$$

If $\overline{\mu} > \overline{\lambda}$, then for the steady state of the system there exists a waiting time distribution function $W(t)$, for which we will again use the Lindley integral equation method (4.11). The corresponding factorization equation looks like this:

$$\gamma(s) = \frac{K_+(s)}{K_-(s)} = \alpha(-s)\beta(s) - 1 = 0$$

or

$$\gamma(s) = \prod_{i=1}^{k} \sum_{g=1}^{l_i} \frac{a_{ig}\lambda_{ig}}{\lambda_{ig} + s} \cdot \prod_{j=1}^{n} \sum_{q=1}^{c_j} \frac{b_{jq}\mu_{jq}}{\mu_{jq} + s} - 1 = 0.$$

Note that the function

$$\beta(s) = \prod_{j=1}^{n} \frac{b_{jq}\mu_{jq} \dfrac{\prod_{l=1}^{c_j} (\mu_{jl} + s)}{\mu_{jq} + s}}{\prod_{q=1}^{c_j} (\mu_{jq} + s)} = \frac{P(s)}{Q(s)} \qquad (4.28)$$

is the ratio of two polynomials, and the degree of the numerator is less than the degree of the denominator. Then we can write [17]

$$\varphi(s) = -\frac{Q(s)}{Q(0)\prod_{i=1}^{f} \left(1 - \dfrac{s}{\eta_i}\right)},$$

where $\varphi(s)$ – Laplace-Stiehljes transform of the waiting time distribution function $W(t)$; $\eta_i\,(i = 1, 2, ..., f)$ - roots of equations $\gamma(s) = 0$, lying in the left half-plane $\operatorname{Re} s < 0$, and $f = \sum_{j=1}^{n} c_j$.

Using (4.28), we obtain

$$\varphi(s) = \frac{\prod_{j=1}^{n} \prod_{q=1}^{c_j} (\mu_{jq} + s) \prod_{i=1}^{f} \eta_i}{\prod_{j=1}^{n} \prod_{q=1}^{c_j} \mu_{jq} \prod_{i=1}^{f} (\eta_i - s)}. \qquad (4.29)$$

Applying the inverse Laplace-Stieltjes transform from (4.29), we find $W(t)$. Having chosen $z_i = -\eta_i$, we finally obtain:

$$W(t) = 1 - \sum_{i=1}^{f} \frac{\prod_{j=1}^{n}\prod_{q=1}^{c_j}(\mu_{jq} - z_i)\prod_{i=1}^{f} z_i}{\prod_{j=1}^{n}\prod_{q=1}^{c_j}\mu_{jq}\sum_{l=1}^{f}\frac{\prod_{r=1}^{f}(z_r - z_i)}{z_l - z_i}} \cdot e^{-z_i t}.$$

The average waiting time for starting to service low-priority tasks in the queue

$$t_q = -\varphi'(0) = \sum_{i=1}^{f}\frac{1}{z_i} - \sum_{j=1}^{n}\sum_{q=1}^{c_j}\frac{1}{\mu_{jq}}.$$

Average number of low-priority task requests in the queue

$$L_q = \bar{t}_q \cdot \bar{\lambda} = \frac{\sum_{i=1}^{f}\frac{1}{z_i} - \sum_{j=1}^{n}\sum_{q=1}^{c_j}\frac{1}{\mu_{jq}}}{\sum_{i=1}^{k}\sum_{g=1}^{c_j}\frac{a_{ig}}{\lambda_{ig}}},$$

and the total average number of requests for solving low-priority tasks in the system is

$$L_s = \bar{t}_q \cdot \bar{\mu} = \frac{\sum_{i=1}^{f}\frac{1}{z_i} - \sum_{j=1}^{n}\sum_{q=1}^{c_j}\frac{1}{\mu_{jq}}}{\sum_{i=1}^{k}\sum_{g=1}^{c_j}\frac{b_{ig}}{\lambda_{ig}}}.$$

Based on the results conducted research, it is possible to obtain comprehensive characteristics of the performance of low-priority tasks in real time.

Thus, the analysis of the functioning of the process of solving MIS tasks in real time on the basis of the studied G/G/1 type queuing system makes it possible to plan the procedure for their functioning and evaluate the more important characteristics of the functioning of the entire information and management system.

4.3 Probabilistic analysis and selection of I/O processor parameters for control computers

The use of computers in control systems for various objects implies the need to solve the problems of rational exchange of information between the computer and the control object. As already noted, "output and, especially, input of information is the most vulnerable component of modern computers" [22]. This is especially true for control computers of non-stationary objects due to strict restrictions on dimensions, weight, and power consumption. Progress in

microelectronics, the creation of microprocessors made it possible to implement the input-output device of the control computer in the form of an autonomous unit – an input-output processor, which exchanges information in parallel with the operation of the central processor (here in after referred to as the CPU).

The design of an input-output processor for control computers of non-stationary objects has certain features compared to the design of information exchange systems for universal computers. This is, first of all, the creation of algorithms and programs for information exchange, taking into account the specifics of external subscribers of the control system, the selection of a rational structure and parameters of the input-output processor.

One of the basic modes of operation of the I/O processor is the exchange of word arrays between the random access memory (RAM) of the CPU and the control system subscribers. Since the CPU usually exchanges word arrays of random size with different subscribers, the processing time of one request by the I/O processor is random. The time between the receipt of individual requests for exchange is also random. Thus, the I/O processor operates in the queueing system mode.

To simplify the analysis, we will assume that the information transfer process occurs in one direction – from the CPU to the control system subscribers and the source of exchange requests is the central processor.

The process of transferring one word from the CPU RAM to the control system subscriber occurs in two stages: first, this word is transferred from the CPU RAM to one of the I/O processor registers in parallel code, and then to the interface, with a breakdown into bytes or bits of information depending on the type of communication line.

Thus, if one word is transferred from the CPU RAM to the I/O processor in time τ, then this same word is transferred from the I/O processor to the control system subscriber in time $k\tau$, where k can vary from 1 to m – the bit size of the parallel word.

Taking into account the above, to simplify the analysis of the operation of the input-output processor, it is formally considered that the process of transferring an array of words of size N from the CPU RAM to the control system subscriber consists of two stages: transferring this array from the CPU RAM to the input-output processor in time $N\tau$ and transferring the array of words from the input-output processor to the control system subscriber in time $kN\tau$.

We assume that the flow of exchange requests coming from the central processor to the I/O processor follows an exponential distribution with parameter λ;

the size of the word arrays exchanged between the CPU and the subscriber of the control system also follows an exponential distribution with mean value m. The average intensity of request processing when receiving information into the I/O processor from the CPU RAM is $\mu = 1/m\tau$, and during issuance – $\mu_{out} = \mu / k$. Let us consider a number of schemes for organizing an input-output processor with respect to these assumptions.

Scheme 1. The central processor initiates the exchange of an array of words, the input-output processor sequentially, word by word, transmits this array to the subscriber control system. Until the exchange is completed, the CPU cannot initiate a new request for exchange.

The graph of states and transitions of the specified exchange module scheme is shown in Figure 4.4, a, where [0] is the state when the module is idle, [10] is the state when the input-output processor receives an array of words from the CPU's RAM, and [01] is the state when the input-output processor transmits the next array of words to the subscriber of the control system.

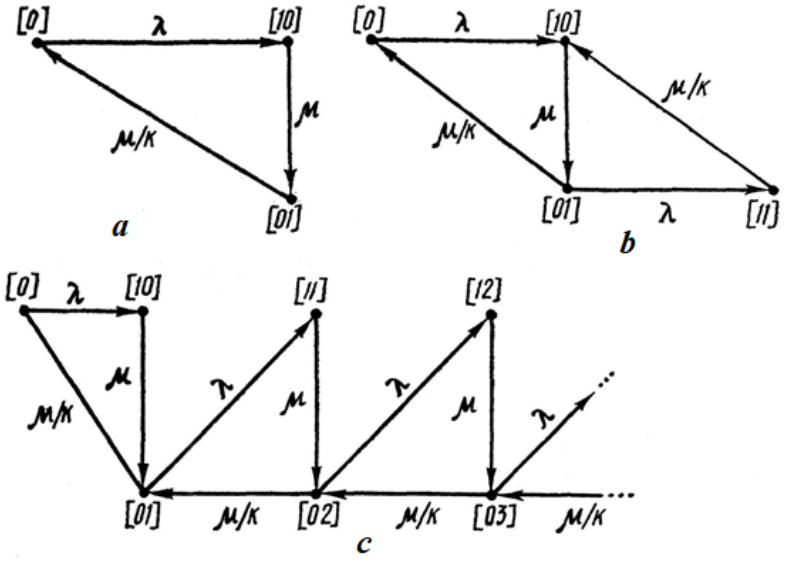


Figure 4.4 – State and transition graphs for I/O processor construction schemes

Let us determine the characteristics of the functioning of an input-output processor built according to this scheme, by compiling a system of equations for the probabilities of states according to this graph of states and transitions:

$$\begin{aligned} -\lambda p_0 + \mu / k\ p_{01} &= 0, \\ -\mu p_{10} + \lambda p_0 &= 0, \\ p_0 + p_{01} + p_{10} &= 1. \end{aligned} \qquad (4.30)$$

Solving the system, we get the following:

– probability of I/O processor downtime – $p_0 = \frac{k}{k + (1+k)\rho}$, where $\rho = \frac{\lambda k}{\mu}$ – loading the input-output processor with the information output process;

– bandwidth $q_1 = \frac{1 - p_0}{\frac{1}{\mu} + \frac{k}{\mu}} = \frac{\lambda k}{k + (1+k)\rho}$;

– probability of CPU RAM blocking – $R_1 = 1 - p_0 = \frac{(1+k)\rho}{k + (1+k)\rho}$.

Scheme 2. A buffer memory device (here in after referred to as BMD) is introduced into the input-output processor, with the dimension of the maximum possible word array. The process of transferring the word array is changed compared to scheme 1, namely: first, the transmitted array is written to the BMD, and then it is transferred to the subscriber of the control system. The state and transition graph for this scheme is the same as for scheme 2. A feature of this scheme is the reduction of the probability of blocking the CPU RAM, namely:

$$R_2 = p_{01} = \frac{\rho}{k + (1+k)\rho}.$$

The blocking time of scheme 2 of the exchange module organization, compared to scheme 1, decreases the more the time of transferring one word from the I/O processor to the subscriber of the control system is longer compared to the time of transfer from the CPU RAM to the I/O processor.

Scheme 3. The module has a BMZ that contains an array of information about one request. The processor can generate a new exchange request when the exchange module transmits the next array of words to the subscriber of the control system. The graph of states and transitions for such a variant of the input-output processor is shown in Figure 4.4, *b*: [11] – state when the input-output processor transmits an array of words to the subscriber, and the processor issues a new exchange request.

The system of equations for the state probabilities for this graph is as follows:

$$\begin{aligned} -\lambda p_0 + \mu / k\ p_{01} &= 0, \\ -\mu p_{10} + \lambda p_0 + \mu / k\ p_{11} &= 0, \\ -(\lambda + \mu / k)\, p_{01} + \mu p_{10} &= 0, \\ p_0 + p_{01} + p_{10} + p_{11} &= 1. \end{aligned} \qquad (4.31)$$

Its solution looks like this:

$$p_0 = \frac{k}{k+(1+\rho)(1+k)\rho}, p_{01} = \frac{k\rho}{k+(1+\rho)(1+k)\rho}, p_{10} = \frac{\rho(1+\rho)}{k[k+(1+\rho)(1+k)\rho]},$$

$$p_{11} = \frac{k\rho^2}{k+(1+\rho)(1+k)\rho}, \ \rho = \frac{\lambda k}{\mu}.$$

The basic characteristics of the I/O processor will be as follows:

– probability of CPU RAM blocking $R_3 = p_{10} + p_{11} = \frac{\rho[1+\rho(1+k)]}{k+(1+\rho)(1+k)\rho}$,

– bandwidth $q_3 = \frac{1-p_0}{\frac{1}{\mu}+\frac{k}{\mu}} = \frac{\lambda k(1+\rho)]}{k+(1+\rho)(1+k)\rho}$.

Let's compare the bandwidth of scheme 3 of the I/O processor organization with scheme 2: $\Delta q_{3,2} = q_3 - q_2 = \frac{\lambda\rho k^2}{[k+(1+k)\rho][k+(1+\rho)(1+k)\rho]}$. Dependency graph $\delta_{3,2} = \Delta_{3,2}/q_2 \cdot 100\%$ in the case of fixed λ is shown in Figure 4.5, *a*.

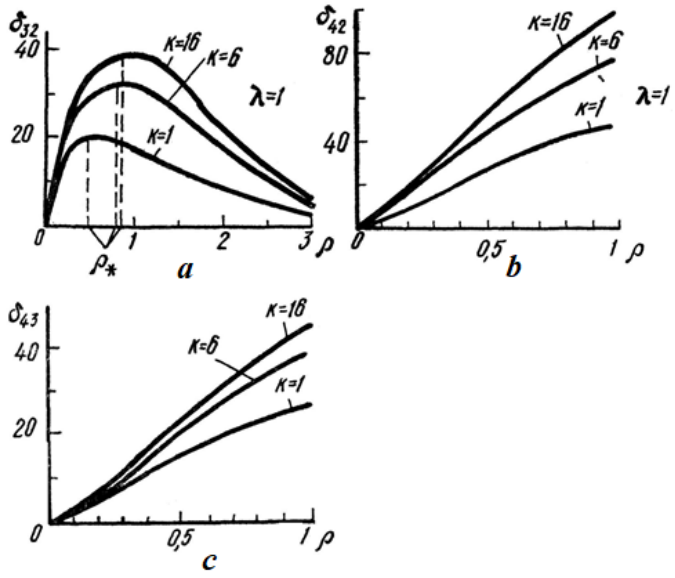


Figure 4.5 – Relative percentage gain of I/O processor design schemes

As is clear from the graph, the function $\delta_{3,2}$ has a maximum at the point ρ_*, which can be determined from the equation $\delta'_{3,2} = 0$.

Thus, in the case of a fixed value of λ, scheme 3 of the organization of the I/O processor in the neighbourhood ρ_* in terms of throughput, it is much higher than schemes 1 and 2 of the exchange organization. At the same time, in the range of small and large values of ρ, scheme 3 of the I/O processor organization has practically no advantages compared to schemes 1 and 2.

Scheme 4. A feature of this scheme of organization of the input-output processor is the algorithm of receiving information arrays and issuing them to the subscribers of the control system. The information array is first received from the CPU RAM to the I/O processor BZP, after which this array is issued from the I/O processor BZU to the subscriber of the SU with the appropriate breakdown of words into bytes or bits of information. If at the time of transmission of the next information array from the I/O processor BZP to the subscriber of the SU a new request for exchange has been received, then the information issuance process is stopped and the exchange module begins to receive a new information array in the BZP. After receipt, the issuance process continues.

The graph of states and transitions for such an organization scheme of the input-output processor is shown in Figure 4.4, c: [1n] – state when there are n word arrays in the BMZ, and the module receives a new array from the CPU RAM; [0n] – state when there are n word arrays in the BMZ of the input-output processor and the module issues the next array to the subscriber of the control system.

It is assumed that the volume of the BMZ is large enough and there is no information loss. Under these assumptions, the system of equations describing the operation of such a variant of the I/O processor in steady-state mode has the following form:

$$\begin{aligned}
&-\lambda p_0 + \mu/k\, p_{01} = 0,\\
&-\mu p_{10} + \lambda p_0 = 0,\\
&-(\lambda+\mu/k)\, p_{01} + \mu p_{10} + \mu/k\, p_{02} = 0,\\
&-\mu p_{1n} + \lambda p_{0n} = 0,\ n>0,\\
&-(\lambda+\mu/k)\, p_{0,n+1} + \mu p_{1n} + \mu/k\, p_{0,n+2} = 0,\ n>0.
\end{aligned} \qquad (4.32)$$

The condition for normalization is $p_0 + \sum_{n=1}^{\infty} p_{0n} + \sum_{n=1}^{\infty} p_{1n} = 1$, the condition for the steady-state mode of operation is the fulfillment of the condition $\rho = k\lambda/\mu < 1$, where ρ – loading the entire I/O processor.

Solving the system of equations (4.32), we obtain:

$$p_0 = \frac{(1-\rho)k}{k+\rho},\ p_{0n} = \frac{1-\rho}{k+\rho} k\rho^n,\ p_{1n} = \frac{1-\rho}{k+\rho}\rho^n. \qquad (4.33)$$

Average number of requests in the I/O processor – $L_s = \frac{(1+k)\rho}{(k+\rho)(1-\rho)}$, bandwidth $-q_4 = \frac{\lambda k}{k+\rho}$. The bandwidth advantage of the I/O processor built

according to scheme 4 compared to schemes 1, 2 and 3 is determined by the following expressions

$$\Delta q_{4,2} = q_4 - q_2 = \frac{\lambda\rho k^2}{(k+\rho)[k+(1+k)\rho]}, \tag{4.34}$$

$$\Delta q_{4,3} = q_4 - q_3 = \frac{\lambda\rho^2 k^2}{(k+\rho)[k+(1+\rho)(1+k)\rho]}. \tag{4.35}$$

Dependency graphs $\delta_{4,2} = \frac{\Delta q_{4,2}}{q_2}100\%$ and $\delta_{4,3} = \frac{\Delta q_{4,3}}{q_3}100\%$ regarding I/O processor usage ρ are presented in Figures 4.5, *b* and 4.5, *c*. It is clear from the graphs that scheme 4 of organizing the input-output processor in the high-load area has significant advantages in terms of throughput and significantly outweighs other schemes of organizing the input processor.

4.4 Probabilistic modeling of the training test subdivision of device production

Control systems for non-stationary objects (cars, airplanes, ships) include a significant number of various devices, such as information sensors, controllers and computers. As a rule, they function in conditions of an aggressive external environment. The temperature can vary from – 120^0C to + 70^0C, devices are affected with vibration during movement, high humidity can lead to loss of dielectric properties of materials. At the same time, high requirements are imposed on the reliability of the operation of control systems for non-stationary objects. It is possible to provide high reliability of such objects both at the design stage and at the stage of production, by including training operations of devices in appropriate conditions on special stands with subsequent monitoring for correct functioning [23]. Thus, the simplified technological scheme of the training tests subdivision of device production can be the same as in Figure 4.6.

Failures of devices during training operations are random. This leads to congestion of individual technological sections and transport conveyors, reducing the capacity and production capacity of the entire system. At the same time, as practice shows, such production systems designed according to well-developed methods of flow-mass production design often can not provide the required productivity and throughput, despite the sufficient capacity of the assembly sections. Possible irregularity of production also render a strong influence on the performance of the system. In this regard, the traditional methods of calculation of the flow-mass production design of devices production must be supplemented by probabilistic methods that allow take into account various random factors affecting the production process.

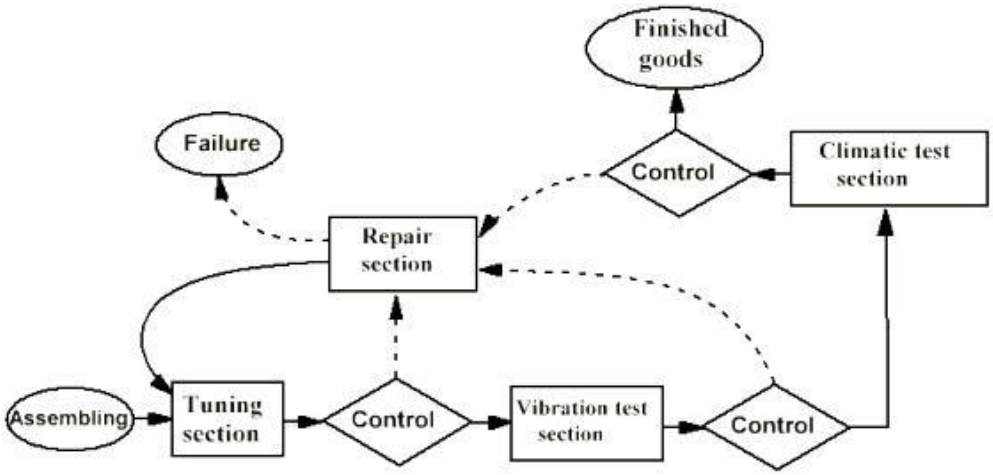


Figure 4.6 – Technological scheme of the training tests subdivision of device production

In a formalized presentation, each technological section of the training test subdivision is a service system consisting of a certain number of service places (these may be the workplaces of the regulators or the seats in the climatic test area), the input flow of devices that act as requests for service, and the place of waiting for requests, in whose role the intermediate buffers can be used. In the whole, the training tests subdivision of device production can be modelling by a queuing network, in which the assembly section is the source of requests (in this case, the assembled devices), separate technological sections are separate service systems.

The probabilistic model of the training tests subdivision of device production will be a queuing network with an appropriate topology. The aim of the work is to determine, based on the queuing network, the load of each production section, the average cycle time of control and training tests, the effect of device failures on the general characteristics of the system.

The structure of the training tests subdivision of device production can be represented by the network transmission graph, shown in Figure 4.7. The node S_0 corresponds to the assembly section, S_1 – to the tuning section, S_2 – to the vibration test section, S_3 – to the climatic test section, S_4 – to the repair section. Solid arcs correspond to the passage of devices along the technological chain without failures; arcs marked with strokes mean the transfer of failed devices to the repair section, or a completely defective product. p_{ij} means the probability of transferring the devices from section i to section j. Accordingly, p_{23} is the probability that the device has passed the vibration test section normally and p_{24} is the probability that

the device require a repair after vibration test, p_{30} is the probability that the device after section of climatic tests has passed all the training tests normally and p_{34} is the probability that the device require a repair after climatic test, p_{40} – this is the probability that the device is not subject to recovery.

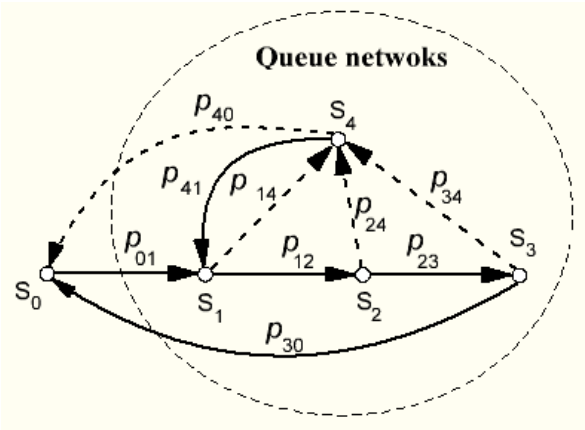


Figure 4.7 – Networks transmission graph

Let us define the transition probability matrix P as consisting of elements p_{ij}, that is

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & p_{12} & 0 & p_{14} \\ 0 & 0 & 0 & p_{23} & p_{24} \\ p_{30} & 0 & 0 & 0 & p_{34} \\ p_{40} & p_{41} & 0 & 0 & 0 \end{bmatrix}. \qquad (4.36)$$

The peculiarity of this matrix is that the sum of the probabilities in the rows is equal to one.

We will investigate the functioning of the system in a stationary state. Let us denote by λ_i the total intensity of the input flow of devices in section i in a stationary mode, τ_i – the average time of the technological operation in the i-th section. For the existence of a stationary state, it is necessary that the inequality $\rho_i = \lambda_i \tau_i < 1$ holds for each section, where ρ_i is a load of i-th section. Then the following system of equations for the stationary state corresponds to the graph in Figure 4.7

$$
\begin{aligned}
&-\lambda_0 + p_{30}\cdot\lambda_3 + p_{40}\cdot\lambda_4 = 0,\\
&\lambda_0 - \lambda_1 + p_{41}\cdot\lambda_4 = 0,\\
&p_{12}\cdot\lambda_1 - \lambda_2 = 0,\\
&p_{23}\cdot\lambda_2 - \lambda_3 = 0,\\
&p_{14}\cdot\lambda_1 + p_{24}\cdot\lambda_2 + p_{34}\cdot\lambda_3 - \lambda_4 = 0.
\end{aligned}
\tag{4.37}
$$

In the matrix form, the system of equations (4.37) is written by

$$
\begin{bmatrix}
-1 & 0 & 0 & p_{30} & p_{40}\\
1 & -1 & 0 & 0 & p_{41}\\
0 & p_{12} & -1 & 0 & 0\\
0 & 0 & p_{30} & -1 & 0\\
0 & p_{14} & p_{24} & p_{34} & -1
\end{bmatrix}
\cdot
\begin{bmatrix}
\lambda_0\\ \lambda_1\\ \lambda_2\\ \lambda_3\\ \lambda_4
\end{bmatrix}
=
\begin{bmatrix}
0\\ 0\\ 0\\ 0\\ 0
\end{bmatrix}.
\tag{4.38}
$$

If we add to the elements of the last row in system matrix corresponding elements of all the remaining rows, we get a row with zero elements. Thus, the determinant of the system matrix of the system of equations (4.38) is equal zero. That is, the system of equations is uncertain. But if we fix the intensity of the arrival of devices from the assembly section λ_0, then the system of equations becomes definite and can be solved relatively λ_0, that can be expressed as

$$
\lambda_i = \alpha_i\lambda_0, i = 1,2,3,4. \tag{4.39}
$$

Let us reformat the system of equations (4.37) as follows

$$
\begin{aligned}
&p_{30}\cdot\lambda_3 + p_{40}\cdot\lambda_4 = \lambda_0,\\
-\lambda_1 &+ p_{41}\cdot\lambda_4 = -\lambda_0,\\
p_{12}\cdot\lambda_1 - \lambda_2 &= 0,\\
p_{23}\cdot\lambda_2 - \lambda_3 &= 0,\\
p_{14}\cdot\lambda_1 + p_{24}\cdot\lambda_2 + p_{34}\cdot\lambda_3 - \lambda_4 &= 0.
\end{aligned}
\tag{4.40}
$$

Reordering the corresponding rows, rewrite the system of equations (4.40) as follows:

$$
\begin{aligned}
-1\cdot\lambda_1 &+ p_{41}\cdot\lambda_4 = -1\cdot\lambda_0,\\
p_{12}\cdot\lambda_1 - 1\cdot\lambda_2 &= 0,\\
p_{14}\cdot\lambda_1 + p_{24}\cdot\lambda_2 + p_{34}\cdot\lambda_3 - 1\cdot\lambda_4 &= 0,\\
p_{23}\cdot\lambda_2 - 1\cdot\lambda_3 &= 0,\\
p_{30}\cdot\lambda_3 + p_{40}\cdot\lambda_4 &= 1\cdot\lambda_0.
\end{aligned}
\tag{4.41}
$$

The solution of the system of linear algebraic equations (4.41) is expediently sought by the Gauss method in a matrix form, which makes it possible to

simultaneously test the system for solvability. We write the extended matrix system (6) in augmented matrix form (only coefficients for unknowns) as follows

$$\left(\begin{array}{cccc|c} -1 & 0 & 0 & p_{41} & -1 \\ p_{12} & -1 & 0 & 0 & 0 \\ p_{14} & p_{24} & p_{34} & -1 & 0 \\ 0 & p_{23} & -1 & 0 & 0 \\ 0 & 0 & p_{30} & p_{40} & 1 \end{array}\right).$$

Let us realize the direct move of the Gauss method. To do this, multiply all the elements of the first row by p_{12} and add them to the corresponding elements of the second row. After that, multiply all elements of the first row by p_{14} and add them to the corresponding elements of the third row. These operations are formally written as follows.

$$\left\{II := II + I \cdot \Pi_{12},\ III := III + I \cdot p_{14}\right\}$$

As a result, we get

$$\left(\begin{array}{cccc|c} -1 & 0 & 0 & p_{41} & -1 \\ 0 & -1 & 0 & p_{41}p_{12} & -p_{12} \\ 0 & p_{24} & p_{34} & p_{41}p_{14} - 1 & p_{14} \\ 0 & p_{23} & -1 & 0 & 0 \\ 0 & 0 & p_{30} & p_{40} & 1 \end{array}\right)$$

Next, we perform the following operations

$$\left\{III := III + II \cdot p_{24},\ IV := IV + II \cdot p_{23}\right\}$$

As a result, we get

$$\left(\begin{array}{ccccc|c} -1 & 0 & 0 & p_{41} & -1 \\ 0 & -1 & 0 & p_{41}p_{12} & -p_{12} \\ 0 & 0 & p_{34} & p_{41}p_{14} + p_{41}p_{12}p_{24} - 1 & -p_{14} - p_{12}p_{24} \\ 0 & 0 & -1 & p_{41}p_{12}p_{23} & -p_{12}p_{23} \\ 0 & 0 & p_{30} & p_{40} & 1 \end{array}\right)$$

Perform the following operation: $\left\{V := V + III\right\}$

As a result, we get

$$\left|\begin{array}{cccc|c} -1 & 0 & 0 & p_{41} & -1 \\ 0 & -1 & 0 & p_{41}p_{12} & -p_{12} \\ 0 & 0 & p_{34} & p_{41}p_{14}+p_{41}p_{12}p_{24}-1 & -p_{14}-p_{12}p_{24} \\ 0 & 0 & -1 & p_{41}p_{12}p_{23} & -p_{12}p_{23} \\ 0 & 0 & p_{30}+p_{34} & p_{40}+p_{41}p_{14}+p_{41}p_{12}p_{24}-1 & 1-p_{14}-p_{12}p_{24} \end{array}\right|$$

Above the elements of the fifth row perform the following transformations:

$$p_{30}+p_{34}=1,$$

$$p_{40}+p_{41}p_{14}+p_{41}p_{12}p_{24}-1=-(1-p_{40})+p_{41}p_{14}+p_{41}p_{12}p_{24}=$$

$$=-p_{41}+p_{41}p_{14}+p_{41}p_{12}p_{24}=-p_{41}(1-p_{14})+p_{41}p_{12}p_{24}=$$

$$=-p_{41}p_{12}+p_{41}p_{12}p_{24}=-p_{41}p_{12}(1-p_{24})=-p_{41}p_{12}p_{23},$$

$$1-p_{14}-p_{12}p_{24}=(1-p_{14})-p_{12}p_{24}=p_{12}-p_{12}p_{24}=$$

$$=p_{12}(1-p_{24})=p_{12}p_{23}.$$

As a result, we get

$$\left|\begin{array}{cccc|c} -1 & 0 & 0 & p_{41} & -1 \\ 0 & -1 & 0 & p_{41}p_{12} & -p_{12} \\ 0 & 0 & p_{34} & p_{41}p_{14}+p_{41}p_{12}p_{24}-1 & -p_{14}-p_{12}p_{24} \\ 0 & 0 & -1 & p_{41}p_{12}p_{23} & -p_{12}p_{23} \\ 0 & 0 & 1 & -p_{41}p_{12}p_{23} & p_{12}p_{23} \end{array}\right|. \qquad (4.42)$$

The fourth and fifth rows of the system (4.42) are equivalent. Throw away the fifth row and swap the third and fourth rows. We get

$$\left|\begin{array}{cccc|c} -1 & 0 & 0 & p_{41} & -1 \\ 0 & -1 & 0 & p_{41}p_{12} & -p_{12} \\ 0 & 0 & -1 & p_{41}p_{12}p_{23} & -p_{12}p_{23} \\ 0 & 0 & p_{34} & p_{41}p_{14}+p_{41}p_{12}p_{24}-1 & -p_{14}-p_{12}p_{24} \end{array}\right|$$

Using the third row, let us perform the following conversion: $\{IV := IV + III \cdot p_{34}\}$

We get

$$\left|\begin{array}{cccc|c} -1 & 0 & 0 & p_{41} & -1 \\ 0 & -1 & 0 & p_{41}p_{12} & -p_{12} \\ 0 & 0 & -1 & p_{41}p_{12}p_{23} & -p_{12}p_{23} \\ 0 & 0 & 0 & p_{41}p_{14}+p_{41}p_{12}p_{24}+p_{41}p_{12}p_{23}p_{34}-1 & -p_{14}-p_{12}p_{24}-p_{12}p_{23}p_{34} \end{array}\right|$$

From the last row follows:

$$\lambda_4=\frac{p_{14}+p_{12}p_{24}+p_{12}p_{23}p_{34}}{1-p_{41}p_{14}-p_{41}p_{12}p_{24}-p_{41}p_{12}p_{23}p_{34}}\lambda_0.$$

Accordingly, from the third row we get:

$$\lambda_3 = p_{12}p_{23} \cdot \lambda_0 + p_{41}p_{12}p_{23}\lambda_4,$$

from the second row we get

$$\lambda_3 = p_{12}p_{23} \cdot \lambda_0 + p_{41}p_{12}p_{23}\lambda_4$$

and from the first row we get

$$\lambda_1 = \lambda_0 + p_{41} \cdot \lambda_4 .$$

So

$$\alpha_4 = \frac{p_{14} + p_{12}p_{24} + p_{12}p_{23}p_{34}}{1 - p_{41}p_{14} - p_{41}p_{12}p_{24} - p_{41}p_{12}p_{23}p_{34}}, \alpha_3 = p_{12}p_{23} + p_{41}p_{12}p_{23}\alpha_4 , \quad (4.43)$$
$$\alpha_2 = p_{12} + p_{41}p_{12}\alpha_4 , \alpha_1 = 1 + p_{41}\alpha_4 .$$

The most important indicators of production will be the intensity of devices that have completely passed the training tests λ_{out} and the intensity of completely defective devices λ_{fail}. From the network transmission graph follows:

$$\lambda_{out} = \lambda_3 p_{30}, \lambda_{fail} = \lambda_4 p_{40} . \quad (4.44)$$

Important characteristic of the production process is the average time of the devices passing through the training and control operations – *T*. It, in turn, depends on the average time spent by devices (customers in terms of queuing theory) in each section $t_i, i = 1,2,3,4$. Then $T = \sum_{i=1}^{4} \alpha_i t_i$ [20].

The execution time of a production operation in the *i*-th section is constant – τ_i. Therefore, a queuing system of the type M/D/1 can be chosen as a model of production section. Then $t_i = \frac{\rho_i}{2(1-\rho_i)} + \frac{1}{\mu_i}$ [17]. After transformations $t_i = \frac{\rho_i}{2(1-\rho_i)} + \frac{1}{\mu_i} = \frac{\lambda_i \tau_i}{2(1-\lambda_i \tau_i)} + \tau_i = \left[\frac{\alpha_i \lambda_0}{2(1-\alpha_i \lambda_0 \tau_i)} + 1 \right] \tau_i$ and

$$T = \sum_{i=1}^{4} \left[\frac{\alpha_i \lambda_0}{2(1-\alpha_i \lambda_0 \tau_i)} + 1 \right] \tau_i \alpha_i . \quad (4.45)$$

By using obtained expressions the performance of the training test sections can be reasonably chosen.

Example. Let the transition probability matrix ***P*** of the network have the form:

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0,8 & 0 & 0,2 \\ 0 & 0 & 0 & 0,9 & 0,1 \\ 0,7 & 0 & 0 & 0 & 0,3 \\ 0,3 & 0,7 & 0 & 0 & 0 \end{bmatrix}$$

Using the expressions (4.43), we get:

$$\lambda_1 = 1,53\lambda_0,\ \lambda_2 = 1,23\lambda_0,\ \lambda_3 = 1,10\lambda_0, \lambda_4 = 0,76\lambda_0.$$

Accordingly $\lambda_{out} = 0,77\lambda_0,\ \lambda_{fail} = 0,76\lambda_0$

Thus, the received expressions allow to estimate loading of each technological section and to choose its productivity.

On the Figure 4.8 depicted the family of curves λ_{out} as a function of the probability of a complete failure of the device p_{40} for different values of the probability of failures in the vibration test section – p_{24}.

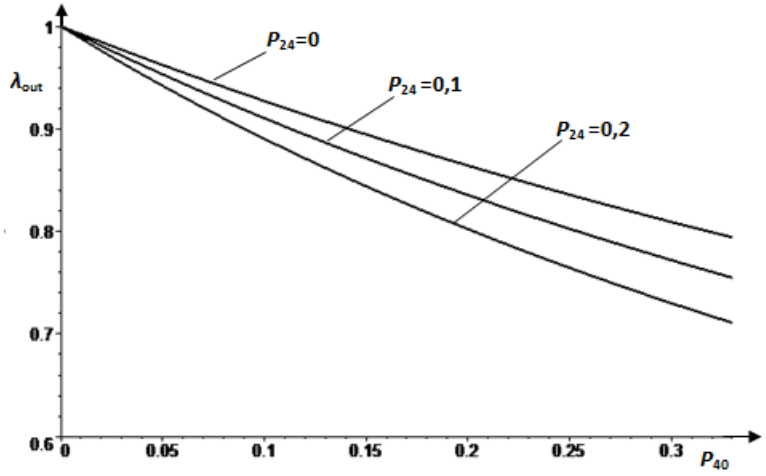


Figure 4.8 – Related intensity of devices that have completely passed the training tests

The dependencies are almost linear, and each section contributes to the decrease in the performance of the entire system of training operations.

4.5 A queuing system with a finite number of sources as a model of a flexible manufacturing system

Small-scale and medium-scale production is most effectively implemented in the form of flexible manufacturing system (FMS). FMS are complex dynamic systems, the creation of which requires large investments. Therefore, in order to achieve their high efficiency and quick payback, it is necessary to ensure a high

utilization rate of the main production equipment, high quality of manufactured products. This imposes serious requirements on the design of the FMS, during which it is necessary to solve many technical and organizational problems, take into account a large number of factors. One of the most important and obligatory stages in the design of the FMS is modeling, which allows you to simulate the practical testing of the system, and the design process itself is combined with scientific research on the corresponding models. An important place in modeling the FMS, taking into account the stochastic nature of their functioning, belongs to queuing systems [24].

One of the types of FMS is a flexible automated cite, which is a set of computer numerical control machine (CNCm) units from 3 to 10 CNC machines, robots and means of equipping them combined with an automated control system and a transport system, which provides for the possibility of changing the sequence of using technological equipment. In this type of FMS the CNCm, having completed processing a part, sends a request to the transport system. In response to this request, the transport robot performs several work cycles. This is especially evident when transporting blanks and parts on pallets, where different pallets are used for blanks and parts, and for each request from the machine, the transport robot performs several cycles.

Let λ – the intensity of requests from CNCm, the number of which is equal to n, τ– the average processing time of one cycle. Then $\mu = 1/\tau$ – intensity of service of one cycle, $M = \mu / c$ – intensity of service of the whole requests from CNCm. Assuming Poisson event flows, we will develop and study a mathematical model of this FMS, which will be a single-line queuing system with a finite number of request sources and multi-stage servicing of each request. We will carry out research for a stationary state. The mathematical apparatus of the developed model is based on the results of the work [**], in the implementation of which the author took an active part.

Denote by $q_0 = p_0$ the probability of idle of transport robot; $p_i, 1 \le i \le n,$ - the probability that i CNCm sent requests on service(one of them is serviced by a transport robot, and the rest are waiting for service); $q_{ij}, 1 \le i \le n, 1 \le j \le c,$ – the probability that the i CNCm have sent service requests and the transport module is working out the j-th cycle.

Similarly to [18], we denote by $q_{ij}(u)du$ the probability of the joint occurrence of two events: 1) at an arbitrary moment of the equilibrium state of the machines, i requests for service were sent, the transport module completes the j-th

service cycle, 2) the time during which the current service cycle continues is contained in the interval $[u, u+du]$. $q_{ij}(u)$ are stationary probability densities of the fact that at an arbitrary moment of the equilibrium state of the machines sent requests for service, the transport module fulfills the *j*-th service cycle, the time during which this service cycle continues is equal to *u* (it is assumed that this value is continuous).

The quantities under consideration are interconnected by the following relationships:

$$q_{ij} = \int_0^\infty q_{ij}(u)du, \quad p_i = \sum_{j=1}^{c} q_{ij}, \; p_0 + \sum_{i=1}^{n} p_i = q_0 + \sum_{i=1}^{n}\sum_{j=1}^{c} q_{ij} = 1. \qquad (4.46)$$

Considering the possible states of the system in an arbitrarily small time interval with subsequent passage to the limit, we obtain the following system of equations for :

$$\begin{aligned} &q'_{1,j}(u) = -[\lambda(n-1)+\mu]q_{1,j}(u), \quad 1 \le j \le c, \\ &q'_{i,j}(u) = -[\lambda(n-i)+\mu]q_{i,j}(u) + \lambda(n-i+1)q_{i-1,j}(u), \; 1 < i < n, \; 1 \le j \le c, \\ &q'_{n,j}(u) = -\mu q_{n,j}(u) + \lambda q_{n-1,j}(u), \quad 1 \le j \le c. \end{aligned} \qquad (4.47)$$

The probabilities of states for the case when the time during which the service continues is not taken into account are q_0 and $q_{ij} = \int_0^\infty q_{ij}(u)du$. The sum of these probabilities is equal to one. The boundary conditions have the form:

$$n\lambda q_0 = \int_0^\infty \mu q_{1c}(u)du = \mu q_{1c}\,. \qquad (4.48)$$

Similarly

$$q_{1,1}(0) = \mu q_{2,c} + n\lambda q_0,, \qquad (4.49)$$

$$q_{i,1}(0) = \mu q_{i+1,c}, \; 1 < i < n, \quad q_{n,1}(0) = 0. \qquad (4.50)$$

$$q_{i,j}(0) = \mu q_{i,j-1}, \; 1 \le i \le n, \; 1 < j \le c, \qquad (4.51)$$

To solve systems of equations (4.47) – (4.51), we use the method of discrete binomial transformations [15], which, along with generating functions, are used to reduce differential-difference equations to simpler ones.

Let us introduce discrete transformations:

$$w_{m,j}(u)=\sum_{i=m}^{n-1}\binom{i}{m}q_{n-i,j}(u),\ w_{m,j}(0)=\sum_{i=m}^{n-1}\binom{i}{m}q_{n-i,j}(0),$$
$$w_{m,j}=\int_0^{\infty}w_{m,j}(u)du=\sum_{i=m}^{n-1}\binom{i}{m}q_{n-i,j},\ 0\le m\le n-1,\ 1\le j\le c, \tag{4.52}$$

where are $\binom{i}{m}=\frac{i!}{(i-m)!m!}$ binomial coefficients equal to zero at $i<m$.

Let us multiply all the terms of the equations of system (4.47) with respect to $q'_{1,j}(u),\ j=\overline{1,c}$ by $\binom{n-1}{0}$, the equations of system (4.47) with respect to $q'_{2,j}(u),\ j=\overline{1,c}$ by $\binom{n-2}{0}$, the equations with respect to $q'_{k,j}(u),\ k=3,4,...,n,\ j=\overline{1,c},$ by $\binom{n-k}{0}$ and summing the resulting expressions, taking into account expression (4.52), we obtain

$$w'_{0,j}(u)=-[0\cdot\lambda+\mu]w_{0,j}(u),\ 1\le j\le c\ . \tag{4.53}$$

For $m=1,2,...,n-1$ we will do the following: multiply all the terms of the equations of system (4.47) with respect to $q'_{1,j}(u),\ j=\overline{1,c}$ by $\binom{n-1}{m}$, the equations with respect to $q'_{2,j}(u),\ j=\overline{1,c}$ by $\binom{n-2}{m}$, all the terms of the equations with respect to $q'_{k,j}(u),\ k=3,4,...,n-1,\ j=\overline{1,c}$ by $\binom{n-k}{m}$ and summing the resulting expressions, taking into account expression (8), we obtain the general expression

$$w'_{m,j}(u)=-[m\lambda+\mu]w_{m,j}(u),\ 0\le m\le n-1,1\le j\le c\,. \tag{4.54}$$

In the same way, from formulas (6) we find

$$w_{m,j}(0)=\mu w_{m,j-1},\ 0\le m\le n-1,\ 1<j\le c\,. \tag{4.55}$$

Let's derive an expression for $w_{0,1}(0)$.

Equation (4.48) can be rewritten as follows:

$$0=\mu q_{1c}-n\lambda p_0\,. \tag{4.56}$$

We write equations (4.50), (4.51) and (4.56) as follows:

$$
\begin{aligned}
&q_{n,1}(0) = 0,\\
&q_{n-1,1}(0) = \mu q_{n,c},\\
&q_{n-2,1}(0) = \mu q_{n-1,c},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&q_{2,1}(0) = \mu q_{3,c},\\
&q_{1,1}(0) = \mu q_{2,c} + n\lambda q_0,\\
&0 = \mu q_{1c} - n\lambda q_0.
\end{aligned}
\tag{4.57}
$$

Let us multiply both parts of the *i*-th equation of system (4.57), $i = 1,2,...,n$ by $\binom{i-1}{0}$, and the last equation of system (4.57) by $\binom{n-1}{0}$.

$$
\begin{aligned}
&\binom{0}{0} q_{n,1}(0) = 0,\\
&\binom{1}{0} q_{n-1,1}(0) = \binom{1}{0} \mu q_{n,c},\\
&\binom{2}{0} q_{n-2,1}(0) = \binom{2}{0} \mu q_{n-1,c},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\binom{n-2}{0} q_{2,1}(0) = \binom{n-2}{0} \mu q_{3,c},\\
&\binom{n-1}{0} q_{1,1}(0) = \binom{n-1}{0} \mu q_{2,c} + \binom{n-1}{0} n\lambda q_0,\\
&0 = \binom{n-1}{0} \mu q_{1,c} - \binom{n-1}{0} n\lambda q_0.
\end{aligned}
\tag{4.58}
$$

Taking into account that $\binom{i}{0} = 1$, for the right parts of the equations of system (4.58) we make the substitution $\binom{i-1}{0} \to \binom{i-2}{0}$, (except for the last equation), and sum the left and right parts of the equations of system (4.58). Using discrete binomial transformations, we obtain

$$w_{0,1}(0) = \mu w_{0,c}. \tag{4.59}$$

For further research, we need the following lemma.

Lemma. By definition

$\binom{n}{m} = \frac{n!}{(n-m)!m!}$, $m \le n$. Then for $1 \le m < n$ $\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$.

Proof.

$$\binom{n-1}{m} + \binom{n-1}{m-1} = \frac{(n-1)!}{m!(n-1-m)!} + \frac{(n-1)!}{(m-1)!(n-1-(m-1))!} =$$

$$= \frac{(n-1)!(n-m)}{m!(n-m)(n-m-1)!} + \frac{(n-1)!m}{m(m-1)!(n-1-(m-1))!} = \frac{(n-1)!(n-m)}{m!(n-m)!} + \frac{(n-1)!m}{m!(n-m)!} =$$

$$= \frac{(n-1)!(n-m+m)}{m!(n-m)!} = \frac{(n-1)!n}{m!(n-m)!} = \frac{n!}{m!(n-m)!} = \binom{n}{m}.$$

The lemma is proven.

Let's derive an expression for $w_{m,1}(0)$, $0 < m < n$.

Let us write equations (4.49), (4.50) included $w_{m,1}(0)$ in the system of equations

$$\begin{aligned}
&q_{n-m,1}(0) = \mu q_{n-m+1,c},\\
&q_{n-m-1,1}(0) = \mu q_{n-m,c},\\
&q_{n-m-2,1}(0) = \mu q_{n-m-1,c},\\
&\text{……..……..……..……..….}\\
&q_{2,1}(0) = \mu q_{3,c},\\
&q_{1,1}(0) = \mu q_{2,c} + \lambda n q_0.
\end{aligned} \tag{4.60}$$

Let us multiply the *k*-th equation of system (4.60) $k = 1,2,\ldots,n-m$ by $\binom{m+k-1}{m}$. We get the following system of equations:

$$\begin{aligned}
&\binom{m}{m} q_{n-m,1}(0) = \binom{m}{m} \mu q_{n-m+1,c},\\
&\binom{m+1}{m} q_{n-m-1,1}(0) = \binom{m+1}{m} \mu q_{n-m,c},\\
&\binom{m+2}{m} q_{n-m-2,1}(0) = \binom{m+2}{m} \mu q_{n-m-1,c},\\
&\text{……..……..……..……..…..}\\
&\binom{n-2}{m} q_{2,1}(0) = \binom{n-2}{m} \mu q_{3,c},\\
&\binom{n-1}{m} q_{1,1}(0) = \binom{n-1}{m} \mu q_{2,c} + \binom{n-1}{m} \lambda n q_0.
\end{aligned} \tag{4.61}$$

Using the fact that $\binom{m}{m} = \binom{m-1}{m-1}$ and Lemma 1, system (4.61) can be rewritten

$$\begin{aligned}
&\binom{m}{m} q_{n-m,1}(0) = && \binom{m-1}{m-1} \mu q_{n-m+1,c}, \\
&\binom{m+1}{m} q_{n-m-1,1}(0) = \binom{m}{m} \mu q_{n-m,c} + && \binom{m}{m-1} \mu q_{n-m,c}, \\
&\binom{m+2}{m} q_{n-m-2,1}(0) = \binom{m+1}{m} \mu q_{n-m-1,c} + \binom{m+1}{m-1} \mu q_{n-m-1,c}, && \qquad (4.62) \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&\binom{n-2}{m} q_{2,1}(0) = \binom{n-3}{m} \mu q_{3,c} + && \binom{n-3}{m-1} \mu q_{3,c}, \\
&\binom{n-1}{m} q_{1,1}(0) = \binom{n-2}{m} \mu q_{2,c} + \binom{n-2}{m-1} \mu q_{2,c} + \lambda n \binom{n-1}{m} q_0 .
\end{aligned}$$

Let's multiply both parts of equation (4.56) by $\binom{n-1}{m}$ and add as (n+1)-th equation to the system of equations (4.62), after that we'll multiply both parts of equation (4.56) by $\binom{n-1}{m-1}$ and add to the same equation. We get the following system of equations:

$$\begin{aligned}
&\binom{m}{m} q_{n-m,1}(0) = && \binom{m-1}{m-1} \mu q_{n-m+1,c}, \\
&\binom{m+1}{m} q_{n-m-1,1}(0) = \binom{m}{m} \mu q_{n-m,c} && + \binom{m}{m-1} \mu q_{n-m,c}, \\
&\binom{m+2}{m} q_{n-m-2,1}(0) = \binom{m+1}{m} \mu q_{n-m-1,c} + \binom{m+1}{m-1} \mu q_{n-m-1,c}, \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&\binom{n-2}{m} q_{2,1}(0) = && \binom{n-3}{m} \mu q_{3,c} + && \binom{n-3}{m-1} \mu q_{3,c}, \\
&\binom{n-1}{m} q_{1,1}(0) = && \binom{n-2}{m} \mu q_{2,c} + && \binom{n-2}{m-1} \mu q_{2,c} + \lambda n \binom{n-1}{m} q_0, \\
&0 = && \binom{n-1}{m} \mu q_{1,c} + && \binom{n-1}{m-1} \mu q_{1,c} - \lambda n \binom{n-1}{m} q_0 - \lambda n \binom{n-1}{m-1} q_0 \qquad (4.63)
\end{aligned}$$

Summing up the left and right parts of all equations of system (4.63), we finally obtain:

$$w_{m,1}(0) = \mu\left(w_{m,c} + w_{m-1,c}\right) - \lambda n\binom{n-1}{m-1} q_0,\ 0 < m < n\,. \qquad (4.64)$$

From the system of equations (4.54) it follows

$$w_{m,j}(u) = w_{m,j}(0)\exp[-(m\lambda+\mu)u], 0 \le m < n, 1 \le j \le c\,.$$

Whence

$$w_{m,j} = \int_0^\infty w_{m,j}(u)du = \frac{w_{m,j}(0)}{m\lambda+\mu}, 0 \le m < n, 1 \le j \le c\,. \qquad (4.65)$$

Substituting expression (4.65) into expression (10), after appropriate transformations, we obtain

$$w_{m,j}(0) = w_{m,c}(0)\cdot(1+m\rho)^{c-j},\ 0 \le m < n, 1 < j \le c, \qquad (4.66)$$

where $\rho = \lambda/\mu$.

Substituting expression (4.66) into equation (4.64), we find

$$w_{m-1,c}(0) = [(1+m\rho)^c - 1]w_{m,c}(0) + \lambda m(1+m\rho)\binom{n}{m} q_0,\quad 0 < m < n. \qquad (4.67)$$

Denote by $R(m,\rho)$ and $G(m,\rho)$ the following expressions:

$$R(m,\rho) = (1+m\rho)^c - 1,\ G(m,\rho) = \lambda m(1+m\rho)\binom{n}{m}. \qquad (4.68)$$

Expression (4.67) will be rewritten:

$$w_{m-1,c}(0) = R(m,\rho)w_{m,c}(0) + G(m,\rho)q_0,\quad 0 < m < n. \qquad (4.69)$$

Using the recurrent properties of expression (4.69), we obtain

$$w_{k,c}(0) = \prod_{i=k+1}^{n-1} R(i,\rho)w_{n-1,c}(0) + \left[G(k+1,\rho) + (1-\delta_{k,n-2})\sum_{j=k+1}^{n-2} G(j+1,\rho)\cdot\prod_{e=k+1}^{j} R(e,\rho)\right]\cdot q_0, 0 \le k \le n-2, \qquad (4.70)$$

where $\delta_{j,i} = \begin{cases}1, & j = i,\\ 0, & j \ne i\end{cases}$ is the Dirac function.

Let's use the fact that $w_{n-1,c}(0)=[(n-1)\lambda+\mu]w_{n-1,c}=\mu[1+(n-1)\rho]w_{n-1,c}$ (from (4.65) and $w_{n-1,c}=n\rho q_0=n\lambda/\mu q_0$ (from (4.52) and (4.48)); then $w_{n-1,c}(0)=\lambda n[1+(n-1)\rho]q_0$ and we rewrite the expression for $w_{m,c}(0)$ in the form

$$w_{m,c}(0)=\lambda\left\{\begin{array}{l} n[1+(n-1)\rho]\prod\limits_{i=m+1}^{n-1}R(i,\rho)+ \\ \left[G(m+1,\rho)+(1-\delta_{k,n-2})\sum\limits_{j=m+1}^{n-2}G(i+1,\rho)\prod\limits_{e=m+1}^{j}R(e,\rho)\right] \end{array}\right\}q_0, 0\le m\le n-2. \quad (4.71)$$

Denoting

$$\Omega(m,\rho)=\left\{\begin{array}{l} n[1+(n-1)\rho]\prod\limits_{i=m+1}^{n-1}R(i,\rho)+ \\ \left[G(m+1,\rho)+(1-\delta_{k,n-2})\sum\limits_{j=m+1}^{n-2}G(i+1,\rho)\cdot\prod\limits_{e=m+1}^{j}R(e,\rho)\right] \end{array}\right\}, 0\le m\le n-2,$$

and using expression (4.66), we finally get:

$$\begin{aligned} &w_{m,j}(0)=\lambda(1+m\rho)^{j-1}\Omega(m,\rho)q_0,\ 0\le m<n-1,\ 1\le j\le c. \\ &w_{n-1,j}(0)=\lambda n[1+(n-1)\rho]^j q_0,\ 1\le j\le c, \end{aligned} \quad (4.72)$$

Using (4.65), we form an expression for discrete transformations of $w_{m,j}$ with respect to q_{ij}

$$\begin{aligned} &w_{m,j}=\rho(1+m\rho)^{j-2}\Omega(m,\rho)q_0,\ 1\le j\le c,\ 0\le m<n-1, \\ &w_{n-1,j}=n\rho[1+(n-1)\rho]^{j-1}q_0,\ 1\le j\le c. \end{aligned} \quad (4.73)$$

Using the inverse discrete transformation [15] with respect to $w_{m,j}$, we obtain expressions for the probabilities of states q_{ij} expressed through $q_0=p_0$:

$$\begin{aligned} &q_{1,j}=n\rho[1+(n-\rho)]^{j-1}q_0,\ 1\le j\le c, \\ &q_{i,j}=\left(\sum_{k=0}^{i-1}\sum_{m=0}^{n-1}\rho(-1)^k\binom{n-i+k}{k}[1+(n-i+k)\rho]^{j-2}\Omega(m-i,k)\rho\right)q_0, \\ &\qquad 1<i\le n,\ 1\le j\le c. \end{aligned} \quad (4.74)$$

Using the normalization condition as applied to the probabilities $q_{i,j}$ ($q_0+\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{c}q_{i,j}=1$), we obtain an expression for the downtime probability q_0 :

$$q_0 = \left\{1 + \rho\sum_{j=1}^{c} n\left[1 + (n-\rho)\right]^{j-1} + \right.$$
$$\left. + \sum_{i=2}^{n}\sum_{k=0}^{i-1}\sum_{m=0}^{n-1}\rho(-1)^k\binom{n-i+k}{k}\left[1+(n-i+k)\rho\right]^{j-2}\Omega(m-i,k)\right]\Bigg\}^{-1}. \quad (4.75)$$

Using expressions (4.74), (4.75) and the transition formula (4.46), we can obtain the final expression for the stationary probabilities p_i .

Now, knowing the probabilities q_{ij} or p_i, it is possible to determine other characteristics of a system with a finite number of sources and a group arrival of service requests.

Average number of requests on service from CNCm in the system

$$L = \sum_{i=1}^{nc} \text{int}((i-1)/c)\, p_i = \sum_{i=1}^{n}\sum_{j=1}^{c} i q_{ij}, \quad (4.76)$$

where $\text{int}(a)$ is the integer part of the expression in brackets.

Average number of requests from CNCm waiting to be serviced,

$$L_q = \sum_{i=1}^{nc}\left[\text{int}((i-1)/c) - 1\right]p_i = \sum_{i=1}^{n}\sum_{j=1}^{c}(i-1)q_{ij} \ . \quad (4.77)$$

The average number of CNCm that did not send requests: $\bar{\nu} = n - L$.

The average number of requests from CNCm in the queue and in service L is proportional to the average waiting time in the queue and in service (time in system – τ_s), and the average number of sources that did not send requests is proportional to the average time the request was in the source $1/\lambda$. Thus, the average waiting time for service plus service time is given by

$$\tau_s = L\left[\lambda(n-L)\right]^{-1} \ . \quad (4.78)$$

Using the obtained expressions, it is possible to find the main technical characteristics of the studied FMS.

Transport robot load factor

$$Z = 1 - q_0 . \quad (4.79)$$

The cycle of work CNCm T consists of the processing time of the next part 1/λ, after which a request to the transport robot is formed, and the time in system τ_s , that is, $T = 1/\lambda + \tau_s$. Then the utilization rate of CNCm:

$$U = \frac{1/\lambda}{T} = \frac{1/\lambda}{\frac{1}{\lambda} + \frac{L}{\lambda(n-L)}} = 1 - \frac{L}{n} \ . \quad (4.80)$$

In addition, the average number of idle machines, the average downtime are determined by expressions (4.76), (4.77).

On Figure 4.9 shows a graph of the dependence of the utilization rate of CNCm on the number of machines in the system and the required number of cycles per request with the following input data: average processing time for one part on a CNCm machine $1/\lambda = 40$ min, time for a transport robot to complete one cycle $\tau = 5$ min.

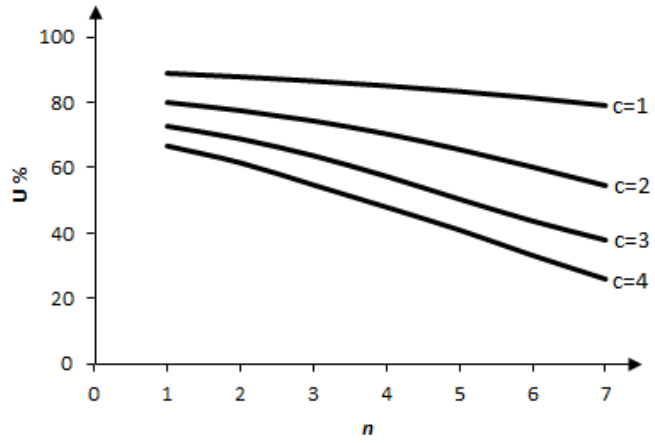


Figure 4.9 – Utilization rate of CNCm

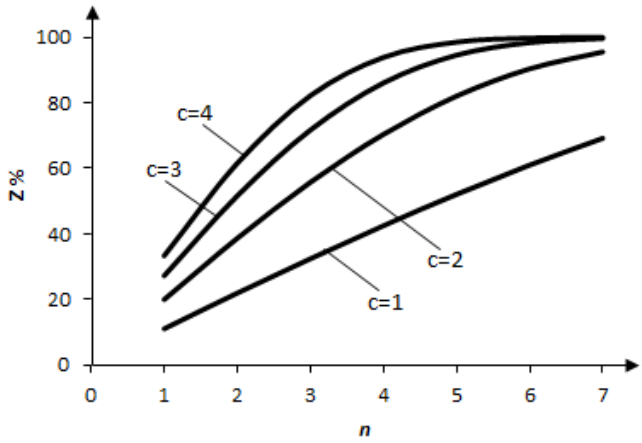


Figure 4.10 – Load factor of a transport robot

It follows from the graphs that with an increase in the number of cycles, the utilization rate of CNCm machines drops sharply. On Figure 4.10 shows a graph of the dependence of the load factor of a transport robot on the number of machines in the system and the required number of cycles per request with the same parameters. From the analysis of the graphs, it follows that for the given parameters, the number of service cycles for one request from CNC machines should not exceed two. With these parameters, the technical characteristics of the FMS remain acceptable.

5 SIMULATION MODELING OF QUEUEING SYSTEMS

5.1 Example and basic concepts of simulation modelling

A network printer on a computer network functions as a classic single-line queue system. If we ignore the printing process itself, the operation of a network printer is mainly determined by two time characteristics: the time between the receipt of print requests from network computers and the time of their service. Since both of these parameters are random, both a queue of print requests and a simple network printer may appear during the operation of a network printer. To simulate such systems, you can use any device that allows you to reproduce time parameters [26]. For example, to simulate the operation of a network printer, you can use two urns, inside which there are chips with numbers on them. The numbers on the chips in the first urn will reproduce (simulate) the time intervals between the receipt of two consecutive print requests, the numbers on the chips in the second urn will reproduce the time of their service. For the purpose of simplification, we will consider all times to be integers, the network printer serves requests one at a time. The time intervals between the receipt of consecutive print requests are in the range from 1 to 15, i.e. the chips in the first box are marked with the numbers 1, 2, …., 15. The service times of individual requests are in the range from 1 to 10, i.e. the chips in the second box are marked with the numbers 1, 2, 3, …., 10. To display the simulation results, we will use the Table 5.1.

Table 5.1 – Blank for displaying simulation results

Request number	ξ_i	η_i	t_i^{a}	t_i^{b}	t_i^{e}	t_i^{w}	t_i^{d}
1	-						
2							
3							
….	….	….	….	….	….	….	….
n-1							
n							
Together (T_n^w, T_n^d)							

The following variables will be used in the model:

- ξ_i – time interval between the arrival of the (*i*-1)-th and *i*-th requests;
- η_i – Service time of the *i*-th request;
- t_i^a – time of arrival of the *i*-th request;
- t_i^b – time of begin of servicing of the *i*-th request;
- t_i^e – time of end of service of the *i*-th request;

- t_i^w – waiting time for the start of service for the *i*-th request;
- t_i^d – network printer downtime caused by the behavior of the *i*-th request.
- T_i^w – total waiting time for requests with numbers from 1 to *i*.
- T_i^d – total downtime of the network printer during maintenance and requests

The time intervals between the arrival of requests, as well as the service times of requests, will be drawn (drawn from boxes). These numbers will be random, and they obey a uniform distribution. The model time starts from zero. All other variables can be calculated using the following formulas:

$$t_i^a = t_{i-1}^a + \xi_i,\ t_i^b = \max\{t_i^a,\ t_{i-1}^e\},\ t_i^e = t_i^b + \eta_i,\ \ t_i^w = t_i^b - t_i^a,\ t_i^d = t_i^b - t_{i-1}^e,$$
$$T_i^w = T_{i-1}^w + t_i^w,\ T_i^d = T_{i-1}^d + t_i^d \qquad (5.1)$$

The variables T_i^w and T_i^d " are used to accumulate, respectively, the total waiting time of requests in the print queue and the total idle time of the network printer. The initial values of all these variables are as follows:

$$\xi_1 = 0,\ t_i^a = 0,\ t_i^b = 0,\ t_i^w = 0,\ t_i^d = 0, T_i^w = 0,\ T_i^d = 0$$

The simulation process will proceed in model time, which starts from zero. The arrival time of the *i*-th request, the start time of the *i*-th request, and the end time of the *i*-th request are absolute in the model time scale. Let us limit ourselves to the simulation process of the first ten requests.

The process of simulating the operation of a network printer is as follows: for the first request, the arrival time and the start time of service are assumed to be zero. From the second box, its service time ($\eta_1 = 8$) is selected, it is entered in the third column, the service end time on the time scale will be 8, the waiting time and idle time are zero. Thus, the first row of the table is filled (Table 5.2).

Table 5.2 – Table state after the first request arrives

Request number	ξ_i	η_i	t_i^a	t_i^b	t_i^e	t_i^w	t_i^d
1	-	8	0	0	8	0	0
2							
3							
….	….	….	….	….	….	….	….
n-1							
n							
Together (T_n^w, T_n^d)						0	0

For the second request, the table row is filled in as follows: the time interval between the arrival of the first and second requests (3) and the time of its service (6) are selected from the urns, respectively. They are entered into the corresponding columns. The time of arrival of the second request is 3. The time of the start of its service is 8, since the previous request finished its service at a time equal to 8. The time of the end of its service is 14 (8 + 6). The service waiting time is 5 (8 – 3), the network printer was working, the idle time is zero ($t_i^d = 0$).Thus, the second row of the table is filled (Table 5.3).

Table 5.3 – Table state after the second request arrives

Request number	ξ_i	η_i	t_i^{a}	t_i^{b}	t_i^{e}	t_i^{w}	t_i^{d}
1	-	8	0	0	8	0	0
2	3	6	3+0=3	8	8+6=14	8-3=5	0
3							
….	….	….	….	….	….	….	….
n-1							
n							
Together (T_n^w, T_n^d)						5	0

For the third request, the table row is filled in as follows: the time interval between the arrival of the second and third requests (12) and the time of its service (1) are selected from the urns, respectively. They are entered in the corresponding columns. The time of arrival of the third request is 15, since the previous request arrived at 3, and the next one after 12 minutes. The time of the start of its service is 15, because the previous request left the system at a time equal to 14. The time of the end of its service is 16 (15 +1). The waiting time for the start of service is 0, but the printer was idle from time 14 to 15. The idle time of the network printer is one (Table 5.4).

Table 5.4 – Table state after the third request arrives

Request number	ξ_i	η_i	t_i^{a}	t_i^{b}	t_i^{e}	t_i^{w}	t_i^{d}
1	-	8	0	0	8	0	0
2	3	6	3+0=3	8	8+6=14	8-3=5	0
3	12	1	3+12=15	max {14,15} = 15	15+1=16	15-15=0	15-14=1
….	….	….	….	….	….	….	….
n-1							
n							
Together (T_n^w, T_n^d)						5	1

For the fourth request, the table row is filled in as follows: the time interval between the arrival of the third and fourth requests (10) and the time of its service (9) are selected from the urns, respectively. They are entered in the corresponding columns. The time of arrival of the fourth request is 25 because the previous request arrived at 15, and the next one after 10 minutes. The time of the start of its service is 25, because the previous request left the system at a time equal to 16. The time of the end of its service is 34 (25 +9). The waiting time for the start of service is 0, but the printer was idle from time 14 to 25. The idle time of the network printer is 9 (Table 5.5)..

Table 5.5 – Table state after the fourth request arrives

Request number	ξ_i	η_i	t_i^{a}	t_i^{b}	t_i^{e}	t_i^{w}	t_i^{d}
1	-	8	0	0	8	0	0
2	3	6	3+0=3	8	8+6=14	8-3=5	0
3	12	1	3+12=15	max {14,15} = 15	15+1=16	15-15=0	15-14=1
4	10	9	15+10=25	max {25,16} = 25	25 +9 = 34	25-25=0	25-16=9
….	….	….	….	….	….	….	….
n-1							
n							
Together (T_n^w, T_n^d)						5	10

Thus, the remaining rows of the table are filled in and the totals are calculated for the waiting time of all print requests and the downtime of the network printer serviced for the time of servicing all 10 requests (Table 5.6).

Table 5.6 – Table state after the tenth request arrives

Request number	ξ_i	η_i	t_i^{a}	t_i^{b}	t_i^{e}	t_i^{w}	t_i^{d}
1	-	8	0	0	8	0	0
2	3	6	3	8	14	5	0
3	12	1	15	15	16	0	1
4	10	9	25	25	34	0	9
5	2	7	27	34	41	7	0
6	13	7	40	41	48	1	0
7	11	2	51	51	53	0	3
8	2	6	53	53	59	0	0
9	5	3	58	59	62	1	0
10	13	7	71	71	78	0	9
Together (T_{10}^w, T_{10}^d)						14	22

The final stage of simulating the printer's operation will be calculating the final characteristics of its operation. The average waiting time for a request to start service can be determined by dividing the total waiting time of all requests by their total number: $\bar{t}_w = T_{10}^w / 10 = 14/10 = 1{,}4$.

The network printer load factor Z can be determined by subtracting the total idle time of the network printer from the time the last request was serviced, dividing the difference by the time the last request was serviced, and multiplying the result by 100%. That is:

$$Z_w = \frac{t_{10}^e - T_{10}^d}{t_{10}^e} \cdot 100\% = \frac{78-22}{78} \cdot 100\% = 71{,}79\%$$

After analysing the procedures carried out above, the following conclusions can be drawn:

1. The completed Table 5.6 is truly a model of the network printer's operation, as it reproduces its operation over time, and the process of filling in Table 5,6 and calculating the final indicators is a simulation.

2. The simulation results can be trusted with a certain degree of caution, since the resulting model directly reproduces the process of network printer operation in time.

3. The simulation results are approximate, since the choice of time parameters is not clearly defined, instead of continuous time intervals, discrete ones are used, the intensity of the print request flow varies over time, and a limited sample is selected for summarization. As a result, the performance indicators of the network printer $\bar{t}_w$ and Z_w are random, and will vary from implementation to implementation.

4. The simulation results do not depend on the method of obtaining random variables ξ_i, η_i and the speed of their obtaining, but depend on the number of requests served and the characteristics of these random variables/

The above example shows a characteristic feature of simulation modeling: to obtain the necessary information, it is necessary to "run" the simulation model, and not to solve it. Simulation models are not able to form their own solution in the form that occurs in analytical models, that is, in the form of dependencies, systems of equations.

We can give the following definition of simulation modelling. Simulation modelling refers to the process of directly reproducing the behaviour of the

modelled system by any means with artificial imitation of the quantities on which the behaviour of the system depends.

From the last property of the above example it follows that for simulation modelling can use the most universal programmable means of performing mathematical and logical calculations invented by mankind – a computer. In the modern interpretation, the reproduction of a system modelled on a computer with artificial imitation of the quantities on which the behaviour of the system depends is called simulation modelling. And the corresponding program is called a simulation model. In general, there are many definitions of simulation modelling, which can be found in the works [26, 27].

The algorithm diagram that simulates the operation of a network printer can be as shown in Figure 5.1.

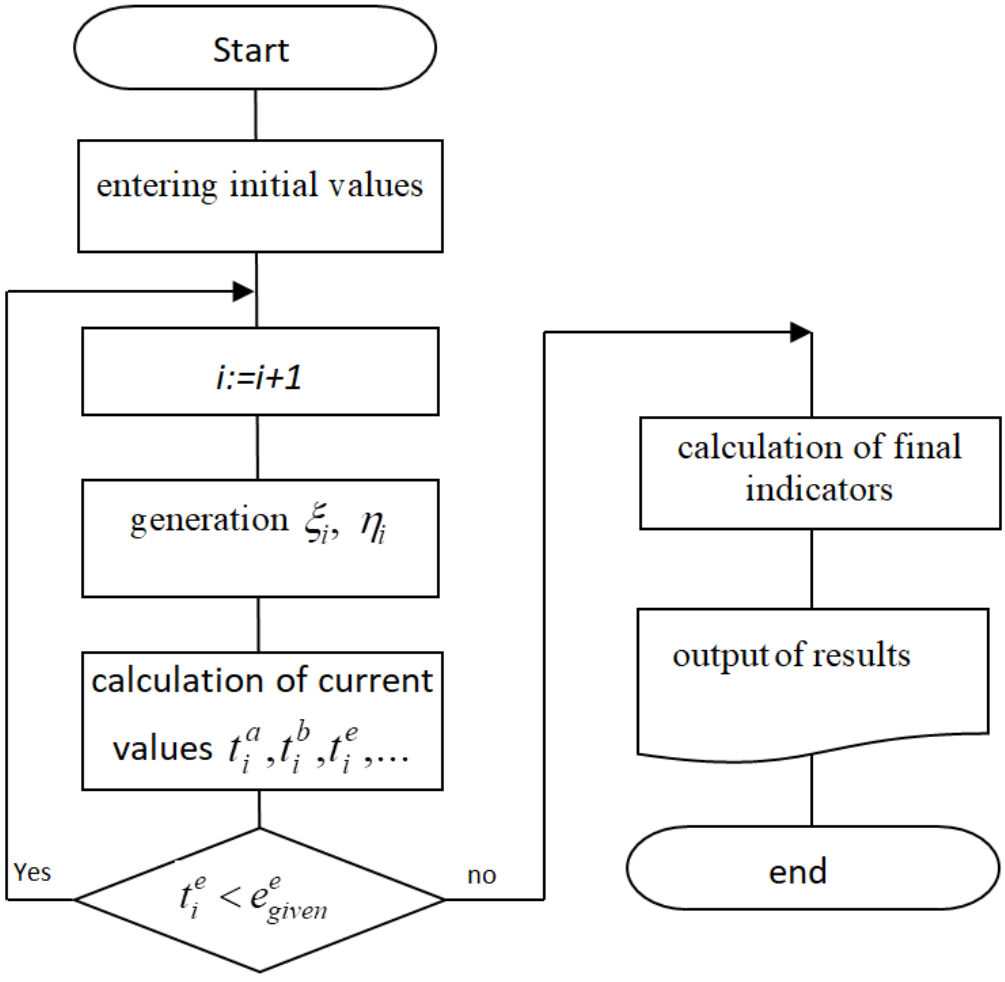


Figure 5.1 – Scheme of the algorithm of the simulation model of a network printer

As you can see from the above scheme, the simulation time is limited by the value t_{given}^e. The results are presented as one value for each indicator. If it is

necessary to analyze the operation of a network printer with variable parameters of either the incoming flow of requests or the service time, then one or two external loops will be added to the algorithm that change the settings of the random number generators. Also, the procedure for processing the final indicators and presenting them to researchers will be significantly complicated.

Using Figure 5.1, you can create a simulation model program in one of the programming systems (C++, Python) and run numerical experiments on it. In principle, simulation models can be developed in any general-purpose programming language. However, this is a very labor-intensive process. Therefore, specialized simulation modeling systems such as AnyLogic, Arena, and MATLAB/Simulink are used to develop simulation models of complex systems.

5.2 Random Number Generation

The quality of a simulation model largely depends on the random numbers with the appropriate distribution laws used in it. The procedure for obtaining random numbers with a given distribution law is called generation, and the program itself for obtaining random numbers with a given distribution law is called a random number generator. As a rule, a random number generator produces so-called pseudorandom numbers with a very large repetition period.

Generating random numbers programmatically is usually done in two stages.

1. Obtaining random numbers τ, uniformly distributed in the range (0, 1).

2. Converting these numbers into random numbers with a given distribution law – $\xi = \Phi(\tau)$.

2.1. Obtaining random numbers with a uniform distribution law in the range (a, b): $\xi = (b-a)\tau + a$.

2.2. Obtaining random numbers distributed according to the exponential law with the parameter λ is performed by the inverse transformation method. According to this method, if α is a random variable uniformly distributed on the interval [0, 1], then the quantity $\xi = F^{-1}(\alpha)$ will have a distribution with the distribution function $F(\xi)$. Let us equate the distribution function to the uniformly distributed quantity α: $1-\exp(-\lambda\xi) = \alpha$. Let us express ξ in terms of α: $\exp(-\lambda\xi) = 1-\alpha$. Hence $-\lambda\xi = \ln(1-\alpha)$, $\xi = -1/\lambda \cdot \ln(1-\alpha)$. Since the quantity α is uniformly distributed on the interval [0, 1], then the quantity $\tau = 1-\alpha$ is also uniformly distributed on the same interval (0, 1). Hence we obtain the final working formula:

$$\xi = -1/\lambda \cdot \ln(\tau).$$

2.3. To obtain random numbers distributed according to the normal law with a mathematical expectation of 0 and a variance of 1, the $\xi = \sum_{i=1}^{12} \tau_i - 12$ formula is used, which is justified by the central limit theorem.

To implement a uniform distribution in the range (0, 1) in programming, the linear congruence method (LCM) is most often used. This is the simplest and most classic algorithm, which formed the basis of most early generators. The algorithm builds a sequence of integers X_{n+1} by the recurrence formula:

$$X_{n+1} = (a \cdot X_n + c) \bmod m,$$

where X_0 – initial number, a – multiplier, c – increment, m – modulus (determines the maximum range). Operation $G \bmod m$ (remainder of division) returns the difference between the dividend and the nearest integer multiple of m, which does not exceed G. To obtain a number in the range (0, 1), the result is divided by m

$$\tau_{n+1} = \frac{X_{n+1}}{m} .$$

Every programming language has tools for generating random numbers with a given distribution law. In Python, the implementation of the laws (distributions) of random number generation depends on the library used. The main tools are the standard **random** module – a standard library for generating pseudo-random numbers and a specialized package for scientific computing SciPy.stats.

The standard random module is suitable for simple tasks and simulation modeling. It supports the main continuous distributions:

Uniform: random.uniform(a, b) for real numbers in the range (a, b) and random.randint(a, b) for integers.

Normal (Gaussian): random.normalvariate(mu, sigma) with mean value mu and standard deviation sigma.

- Lognormal: random.lognormvariate(mu, sigma).
- Exponential: random.expovariate(lambd) with parameter lambd.
- Beta distribution: random.betavariate(alpha, beta).
- Gamma distribution: random.gammavariate(alpha, beta).
- Pareto distribution: random.paretovariate(alpha).

For serious research, it is recommended to use the SciPy.stats library – the most comprehensive tool, containing more than 100 continuous and 20 discrete distributions.

5.3 A Brief Overview of the SimPy Framework

SimPy is a specialized library for Python designed for creating models of discrete-event processes. With this tool, you can model the functioning of complex discrete systems in detail: from network switches to flexible manufacturing systems and logistics. This helps to identify weaknesses in the system and optimize its operation, avoiding the costs of real experiments. It is perfectly suited for modeling queueing processes. The first public version (0.1.2) was presented by developers Klaus Müller and Tony Vignaux on September 17, 2002 on the SourceForge.net platform. SimPy is based on the classic Simula modeling language. The project's documentation is available on the official website: https://simpy.readthedocs.io/en/latest/. SimPy is released as an open source project under the MIT license.

Key Concepts of SimPy

Environment: Manages simulation time and event queue.

Events: Triggers that initiate changes in the system.

Processes: Describe the behavior of active components (clients, patients with doctor's appointments) through yield generator functions..

Resources: Objects with limited capacity (switches, servers) that processes can "capture" and "release".

Before using SimPy, it must be connected to Python (for example, with the pip install simpy command on the operating system) and to a specific program (import simpy).

Processes, resources, and events are described by separate functions that are used to create the corresponding objects of the simulation environment, which is created by the operator env = simpy.environment().This operator performs three critical functions:

- manages time: keeps track of the current simulation time (env.now).
- plans events: maintains a queue of events and determines what should happen next and when.
- runs a simulation: handles processes and advances time from one event to another.

Using the created env variable, you get access to the main modeling tools:

- env.process(func) – registers the generator function as an active process in the system.
- env.timeout(delay) — creates a delay (pause) for a specific process.

- env.run(until=*X*) – starts the execution of the entire model until time point *X* is reached
- env.now — returns the current time value in the model (number).

Let us consider a simulation model of the process of arrival of Poisson flows of requests from two sources with intensities, respectively, $\lambda_1=1$ and $\lambda_2=2$. The model program has the following form[1]:

```
#Simulation model of the sum of two flows
import simpy   # connecting libraries
import numpy as np

def stream(env, name, lamb): # description of the "stream of events" process
  while True:  #process actions:      print:
    print(name, round(env.now,3)) # "process information; request arrival time"
    # stopping the process for a random period of time, distributed according to an exponential law
    yield env.timeout(np.random.exponential(1/lamb))

env = simpy.Environment() # creating a modeling environment
env.process(stream(env, 'stream №1, λ=1, t=', 1)) #creation of an instance of the stream process with parameter λ=1
env.process(stream(env, 'stream №2, λ=2, t=', 0.5)) #creation of an instance of the stream process with parameter λ=2
env.run(until=8) # run the simulation until the model time is 8
```

As in a regular Python program, the first block is the connection of the necessary libraries, primarily SimPy.

The stream function defines a process that, when started, performs the following actions:

- print “process information; request arrival time” for a specific process instance;

- stopping the process for a random period of time, distributed according to an exponential law, formation on the time axis of the event of the next process launch

The stream function contains the following arguments, which are passed from the process instance:

- env – process instance;

[1] The programs in this section are developed with partial use of generative models of artificial intelligence

- name – information about the process;

- lamb – delay parameter.

Operator env = simpy.Environment() – creating a modeling environment.

Operator env.process(stream(env, 'stream №1, λ=1, t=', 1)) – creation of an instance of the stream process with parameter λ=1.

Operator env.process(stream(env, 'stream №2, λ=2, t=', 0.5)) – creation of an instance of the stream process with parameter λ=2.

Operator env.run(until=8) – run the simulation until the model time is 6.

In fact, the entire program consists of several functional blocks:

- connecting Python libraries;

- a block containing functions that describe processes and devices in general;

- constant block (not present in this program);

- a control unit in which the simulation environment is formed, instances of processes, devices, and queues are created, the simulation model is launched, and the progress of the simulation is controlled.

Figure 5.2 shows the results of the simulation model.

```
stream №1, λ=1, t= 0
stream №2, λ=2, t= 0
stream №2, λ=2, t= 0.141
stream №1, λ=1, t= 0.364
stream №2, λ=2, t= 0.707
stream №1, λ=1, t= 0.931
stream №2, λ=2, t= 2.139
stream №2, λ=2, t= 2.318
stream №2, λ=2, t= 2.398
stream №1, λ=1, t= 2.424
stream №2, λ=2, t= 2.556
stream №2, λ=2, t= 2.57
stream №2, λ=2, t= 2.647
stream №1, λ=1, t= 2.844
stream №2, λ=2, t= 3.758
stream №1, λ=1, t= 4.807
stream №2, λ=2, t= 5.261
stream №2, λ=2, t= 5.459
stream №2, λ=2, t= 5.973
```

Figure 5.2 – Joint request flow on the time axis

Indeed, events caused by requests arriving via the second stream occur approximately twice as often as via the first.

The model should not only record specific events but also, depending on the research objectives, generate general characteristics of the object under study. We require the model to output to the console the time intervals between the arrival of adjacent requests for the total flow, and also ensure that the intensity of the total

request flow is equal to the sum of the intensities of the individual flows. The modified model program has the following form.

```
# Modified flow sum model
import simpy   # connecting libraries
import numpy as np

# global variables
request_id = 0 # Global counter for unique request numbers
T_is = 0      # sum of intervals between requests
t_pr = 0      # the moment of receipt of the previous request

def stream(env, name, lamb): # description of the "stream of events" process
   global request_id; global T_is; global t_pr
   while True:  #process actions:      print:
      print(name, f"{env.now:<3.3f} |",f"t_i – t_(i-1) = {env.now – t_pr:<5.3f} ") # "process information; request arrival time"
      T_is += env.now – t_pr; t_pr = env.now; t_i = env.now;      request_id += 1
      # stopping the process for a random period of time, distributed according to an exponential law
      yield env.timeout(np.random.exponential(1/lamb))

env = simpy.Environment() # Creating a modeling environment
env.process(stream(env, 'stream №1, λ=1, t=', 1)) #creation of an instance of the stream process with parameter λ=1
env.process(stream(env, 'stream №2, λ=2, t=', 2)) #creation of an instance of the stream process with parameter λ=2
env.run(until=100) # run the simulation until the model time is 100

T_average = T_is/request_id # middle interval between requests
lambd = 1/T_average        # intensity of the total flow of requests
print()
print(f"number of requests: {request_id:<3} |", f"average interval: {T_average:<7.3f} |", f"λ=: {lambd:<6.3f}")
```

Three global variables are introduced into the program:

- request_id – the number of the incoming request (request counter);
- t_pr – the time of the previous request;
- T_is – a variable for calculating the sum of intervals.

The time intervals between adjacent requests are calculated as the difference between the current request arrival time and the time of the request before it

(env.now – t_pr). Each time the stream process description function is called, the sum of the intervals between requests is adjusted (T_is += env.now – t_pr), the event generation time is recorded (t_pr = env.now), and the request counter is adjusted (request_id += 1).

At the end of the simulation, the average time interval between consecutive requests (T_average = T_is/request_id) and the total request flow rate are calculated. The request counter, average interval between requests, and the total flow rate are displayed on the screen.

Figure 5.3 shows the results of the modified simulation model (the last 8 units of model time out of 100)

```
stream №1, λ=1, t= 92.061 | t_i - t_(i-1) = 0.284
stream №2, λ=2, t= 92.086 | t_i - t_(i-1) = 0.025
stream №2, λ=2, t= 92.201 | t_i - t_(i-1) = 0.115
stream №2, λ=2, t= 92.376 | t_i - t_(i-1) = 0.175
stream №2, λ=2, t= 92.756 | t_i - t_(i-1) = 0.381
stream №2, λ=2, t= 92.934 | t_i - t_(i-1) = 0.178
stream №1, λ=1, t= 92.947 | t_i - t_(i-1) = 0.013
stream №1, λ=1, t= 93.653 | t_i - t_(i-1) = 0.706
stream №2, λ=2, t= 94.215 | t_i - t_(i-1) = 0.562
stream №2, λ=2, t= 94.699 | t_i - t_(i-1) = 0.484
stream №2, λ=2, t= 94.765 | t_i - t_(i-1) = 0.066
stream №2, λ=2, t= 94.965 | t_i - t_(i-1) = 0.200
stream №1, λ=1, t= 95.259 | t_i - t_(i-1) = 0.293
stream №2, λ=2, t= 95.282 | t_i - t_(i-1) = 0.023
stream №1, λ=1, t= 95.453 | t_i - t_(i-1) = 0.171
stream №1, λ=1, t= 95.678 | t_i - t_(i-1) = 0.225
stream №1, λ=1, t= 97.049 | t_i - t_(i-1) = 1.371
stream №2, λ=2, t= 97.236 | t_i - t_(i-1) = 0.187
stream №2, λ=2, t= 97.680 | t_i - t_(i-1) = 0.444
stream №2, λ=2, t= 99.652 | t_i - t_(i-1) = 1.972

number of requests: 303 | average interval: 0.329   | λ=: 3.041
```

Figure 5.3 – Results of the modified simulation model (the last 8 units of model time out of 100)

As can be seen, the experimental results coincide with the theory, while at the same time the actual flow differs from the ideal. This is explained by the stochastic nature of time interval generation. It should be noted that the results of experiments on the model vary from run to run, as Python in this example changes the initial settings of the random number generators from run to run.

Events are the primary mechanism for managing time and synchronizing processes when using SimPy. Processes in SimPy are generator functions that use the yield keyword to pause their execution and "wait" for a specific event. Once the

event occurs, the environment resumes execution of the process. SimPy uses the following event types:

- Timeout: The most common event. It puts the process to sleep for a specified amount of simulated time (simulating the duration of an operation). We used this in the previous programs.

- Resource events: Requests to acquire or release shared resources (e.g., waiting for a processor to become available).

- Basic Event (user events): Used for manual synchronization between different processes. You can create a pure env.event() in one process, pass it to another process, and have the second process wait until the first calls .succeed().

- Logical combinations (& and |): Wait for multiple events to complete, or at least one of them.

Each event passes through exactly three states once:

1. Not triggered – the event has been created in memory, but is not yet bound to a specific point in time.

2. Triggered – the event has reached its trigger time and is added to the SimPy queue.

3. Processed – the event has occurred, and all processes waiting for it have been successfully resumed.

In the following example, we'll consider an example of using a custom event. We'll modify the simulation model of Poisson request flows from two sources as follows. We'll create an event in Stream 1 (current time > 3) that causes the second stream to double the request rate. Stream 2 needs to implement a wait for this event to activate. The modified model program has the following form.

```
# Sum of Streams Plus Event Model
import simpy
import numpy as np

def stream_1(env, name, lamb, alarm_event):
   while True:
      print(f"{name}|",f"λ = {lamb:<2.2f} |",f"t = {round(env.now, 3)}")
      yield env.timeout(np.random.exponential(1/lamb))

       # Trigger condition: if time is past 2 and the event hasn't happened yet
      if env.now > 3 and not alarm_event.triggered:
         print(f"!!! Stream №1 triggers the EVENT at t={round(env.now, 3)} !!!")
         alarm_event.succeed()  # Activate the shared event
```

```
def stream_2(env, name, lamb, alarm_event):
    # Phase 1: Normal operation until the event is triggered
    while not alarm_event.triggered:
        timeout_choice = env.timeout(np.random.exponential(1/lamb))

        # Wait for either the timeout to expire OR the event to fire (OR operator '|')
        result = yield timeout_choice | alarm_event

        # If the event fired during our timeout wait, break the normal loop
        if alarm_event in result:
            print(f"{name} detected the EVENT at t={round(env.now, 3)} and ACCELERATES!")
            break

        print(f"{name}|",f"λ = {lamb:<2.2f} |",f"t = {round(env.now, 3)}")

    # Phase 2: Post-event operation (intensity lambda is doubled)
    lamb = lamb * 2
    while True:
        print(f"{name}|",f"λ = {lamb:<2.2f} |",f"t = {round(env.now, 3)}")
        yield env.timeout(np.random.exponential(1/lamb))

# Initialize the SimPy simulation environment
env = simpy.Environment()

# Create a shared event object to connect the two streams
shared_event = env.event()

# Register and start both processes
env.process(stream_1(env, 'Stream №1', 1, shared_event)) # λ = 1
env.process(stream_2(env, 'Stream №2', 1.5, shared_event)) # λ = 1.5, initial value

# Run the simulation for 6 time units
env.run(until=6)
```

Compared to the base program:

- a trigger for interaction has been created (shared_event =env.event());
- processes have been separated into stream_1 and stream_2, as they now have different logic;

- the | (OR) operator has been introduced, allowing Thread №2 to wait for whichever comes first – the next time step or a signal from Thread №1
(yield timeout_choice | alarm_event);

- the alarm_event.succeed() instruction has been introduced, according to which Thread №1 "enables" the event for the entire environment.

Instructions related to the alarm_event event are highlighted in bold in the program.

Figure 5.4 shows the results of running the simulation model with an embedded event.

```
Stream №1| λ = 1.00 | t = 0
Stream №1| λ = 1.00 | t = 0.079
Stream №2| λ = 1.50 | t = 0.568
Stream №2| λ = 1.50 | t = 0.677
Stream №2| λ = 1.50 | t = 1.43
Stream №1| λ = 1.00 | t = 2.103
Stream №1| λ = 1.00 | t = 2.278
Stream №2| λ = 1.50 | t = 3.289
!!! Stream №1 triggers the EVENT at t=3.303 !!!
Stream №1| λ = 1.00 | t = 3.303
Stream №2 detected the EVENT at t=3.303 and ACCELERATES!
Stream №2| λ = 3.00 | t = 3.303
Stream №1| λ = 1.00 | t = 3.332
Stream №1| λ = 1.00 | t = 3.443
Stream №1| λ = 1.00 | t = 3.874
Stream №2| λ = 3.00 | t = 4.278
Stream №2| λ = 3.00 | t = 4.73
Stream №2| λ = 3.00 | t = 5.406
Stream №1| λ = 1.00 | t = 5.604
```

Figure 5.4 – Results of the simulation model with a built-in event

Indeed, after a model time of 3, the frequency of requests received on the second stream doubled.

5.4 Simulation of queueing systems using SimPy

Let's complicate the model. We'll retain a single input request flow with a service rate of $\lambda = 2$ and add two service devices with a service rate of $\mu = 1.1$ per device. The event flows are Poisson. We received a classic M/M/2 queuing system. During the model run, the screen displays the request number, the time the request arrived, the service start time, and the service delay (waiting time). Upon completion, the model should display the average service wait time. The model program looks like this:

```
# Simulation model of the M/M/2 system
import simpy
import numpy as np
```

```
# global variables
request_id = 0   # Global counter for unique request numbers
T_queue =0       # total waiting time for service requests

def service_process(env, req_num, arrival_time, resource, mu):
    """Process representing the servicing of a request in one of the two servers."""
    global T_queue
    # Request enters the queue for a server
    with resource.request() as request:
        yield request
        start_service_time = env.now
        # Output: request ID, arrival time, service start time and waiting time
        print(f"Request #{req_num:<3} | "
              f"Arrived at: {arrival_time:<7.3f} | "
              f"Service started at: {start_service_time:<7.3f} | "
              f"Waiting time {start_service_time – arrival_time:<6.3f}")
        # Adding a waiting time for the next request
        T_queue +=round(start_service_time-arrival_time, 3)
        # Servicing the request according to an exponential law
        yield env.timeout(np.random.exponential(1 / mu))

def stream(env, label, lamb, resource, mu):
    """Generator process for the incoming stream of requests."""
    global request_id
    while True:
        # Waiting for a random interval between incoming requests
        yield env.timeout(np.random.exponential(1 / lamb))
        # adjusting the application number counter
        request_id += 1
        arrival_time = env.now
        # Spawn an asynchronous servicing process for each new request
        env.process(service_process(env, request_id, arrival_time, resource, mu))

# Creating the SimPy simulation environment
env = simpy.Environment()

# Creating 2 parallel servicing devices (servers)
servers = simpy.Resource(env, capacity=2)

# System configuration parameters
lambda_param = 2  # Arrival rate of the incoming stream
mu_param = 1.1    # Service rate of each individual server

# Initiating the incoming stream of requests
```

```
env.process(stream(env, 't=', lambda_param, servers, mu_param))

# Run the simulation until the model time reaches 10
env.run(until=2000)
print()
print("                Average waiting time ", round(T_queue/request_id, 3))
```

Let's analyze the operation of the model by operators.

Library Imports

- import simpy – imports the SimPy discrete-event simulation library to manage time, events, and resources.
- import numpy as np – imports the NumPy library to generate random numbers (specifically for exponential distributions).

Global Variables

- request_id = 0 – Initializes a global counter variable to assign a unique sequential ID number to each incoming request.
- T_queue =0 – • T_queue =0 – variable for accumulating the total waiting time for all requests

The service_process Function (Request Lifecycle)

- def service_process(env, req_num, arrival_time, resource, mu): – declares a generator function defining how a single request interacts with the service channels.
- with resource.request() as request: – Uses a context manager to request access to one of the available service channels (servers).
- yield request – Pauses this specific process until a service channel becomes free (manages waiting in line/queue).
- start_service_time = env.now – Records the current simulation clock time as the exact moment the service begins.
- print(f"Request #{req_num:<3} | " ...") – Prints a log message to the console, rounding all simulation timestamps to 3 decimal places.
- T_queue + = round(start_...) – Adding a waiting time for the next request
- yield env.timeout(np.random.exponential(1 / mu)) – Freezes the process for a random duration representing service time, calculated exponentially with a mean of 1/mu. The resource is automatically released afterward by the with block.

The stream Function (Request Generator)

- def stream(env, label, lamb, resource, mu): – declares a generator function that continuously generates new requests over time.
- global request_id – grants the function write access to modify the global request_id counter variable.

- while True: – creates an infinite loop to generate requests nonstop for the entire duration of the simulation run.
- yield env.timeout(np.random.exponential(1 / lamb)) – delays the generator for a random inter-arrival time interval, calculated exponentially with a mean of 1/lambda.
- request_id += 1 – increments the global counter by 1 to assign a brand new ID for the newly arrived request.
- arrival_time = env.now – captures the exact simulation timestamp when this specific request entered the system.
- env.process(service_process(…)) – registers and triggers a new independent, asynchronous service_process instance for the current request.

Environment Setup and Execution

- env = simpy.Environment() – instantiates the main simulation environment object which controls the virtual clock and event loop.
- servers = simpy.Resource(env, capacity=2) – creates a shared resource with a capacity of 2, representing two parallel, independent service stations.
- lambda_param = 2 – sets the arrival rate parameter (lambda = 2), representing the average number of incoming requests per time unit.
- mu_param = 1.1 – sets the service rate parameter (mu = 1.1), representing the average number of requests a single server can process per time unit.
- env.process(stream(env, 't=', lambda_param, servers, mu_param)) – registers the primary stream generator process within the simulation environment scheduler.
- env.run(until=2000) – starts the simulation engine execution loop, running all scheduled events from time 0 up to time 2000.
- print()
- print(" Average waiting time ", ...) – output of the average waiting time for all requests

Figure 5.5 shows the results of the simulation model for the first 11 units of time, with a total simulation time of 2000.

```
Request #1   | Arrived at: 0.192   | Service started at: 0.192   | Waiting time 0.000
Request #2   | Arrived at: 0.790   | Service started at: 0.790   | Waiting time 0.000
Request #3   | Arrived at: 0.832   | Service started at: 1.908   | Waiting time 1.076
Request #4   | Arrived at: 0.955   | Service started at: 2.809   | Waiting time 1.854
Request #5   | Arrived at: 2.728   | Service started at: 2.854   | Waiting time 0.126
Request #6   | Arrived at: 3.001   | Service started at: 3.078   | Waiting time 0.077
Request #7   | Arrived at: 3.240   | Service started at: 3.508   | Waiting time 0.268
Request #8   | Arrived at: 3.826   | Service started at: 4.640   | Waiting time 0.815
Request #9   | Arrived at: 4.242   | Service started at: 4.691   | Waiting time 0.448
Request #10  | Arrived at: 4.324   | Service started at: 5.176   | Waiting time 0.852
Request #11  | Arrived at: 4.943   | Service started at: 6.484   | Waiting time 1.541
Request #12  | Arrived at: 5.211   | Service started at: 6.734   | Waiting time 1.523
Request #13  | Arrived at: 5.323   | Service started at: 6.808   | Waiting time 1.486
Request #14  | Arrived at: 5.636   | Service started at: 6.845   | Waiting time 1.209
Request #15  | Arrived at: 5.927   | Service started at: 8.867   | Waiting time 2.940
Request #16  | Arrived at: 7.192   | Service started at: 10.103  | Waiting time 2.911
Request #17  | Arrived at: 8.776   | Service started at: 10.178  | Waiting time 1.402
Request #18  | Arrived at: 8.816   | Service started at: 10.338  | Waiting time 1.522
Request #19  | Arrived at: 8.870   | Service started at: 10.656  | Waiting time 1.786
Request #20  | Arrived at: 8.887   | Service started at: 10.917  | Waiting time 2.029
```

Figure 5.5 – Results of the simulation model for the first 11 units of model time

The average waiting time in the queue $\widetilde{T}_q = 1.193$. Figure 5.6 shows the results of the simulation model for the last 10 units of time, with a total simulation time of 2000.

```
Request #4024 | Arrived at: 1984.933 | Service started at: 1991.563 | Waiting time 6.630
Request #4025 | Arrived at: 1984.996 | Service started at: 1991.647 | Waiting time 6.651
Request #4026 | Arrived at: 1985.484 | Service started at: 1993.205 | Waiting time 7.721
Request #4027 | Arrived at: 1985.488 | Service started at: 1993.418 | Waiting time 7.930
Request #4028 | Arrived at: 1985.733 | Service started at: 1993.902 | Waiting time 8.169
Request #4029 | Arrived at: 1986.370 | Service started at: 1994.009 | Waiting time 7.639
Request #4030 | Arrived at: 1986.598 | Service started at: 1995.032 | Waiting time 8.434
Request #4031 | Arrived at: 1986.976 | Service started at: 1995.202 | Waiting time 8.225
Request #4032 | Arrived at: 1987.532 | Service started at: 1995.378 | Waiting time 7.846
Request #4033 | Arrived at: 1988.476 | Service started at: 1995.435 | Waiting time 6.960
Request #4034 | Arrived at: 1989.321 | Service started at: 1995.458 | Waiting time 6.137
Request #4035 | Arrived at: 1989.762 | Service started at: 1996.056 | Waiting time 6.294
Request #4036 | Arrived at: 1990.137 | Service started at: 1996.585 | Waiting time 6.448
Request #4037 | Arrived at: 1990.329 | Service started at: 1996.671 | Waiting time 6.342
Request #4038 | Arrived at: 1990.808 | Service started at: 1996.753 | Waiting time 5.945
Request #4039 | Arrived at: 1990.886 | Service started at: 1997.432 | Waiting time 6.546
Request #4040 | Arrived at: 1992.337 | Service started at: 1997.483 | Waiting time 5.146
Request #4041 | Arrived at: 1992.398 | Service started at: 1998.643 | Waiting time 6.245
Request #4042 | Arrived at: 1992.947 | Service started at: 1999.751 | Waiting time 6.804

                    Average waiting time  4.523
```

Figure 5.6 – Results of the simulation model for the last 11 units of model time

The average queue wait time (4.523) is approximately the same as the average queue wait time obtained using formula (3.57) for the steady-state mode (4.359). Thus, Figure 5.3 shows the non-stationary mode of the system's operation, while Figure 5.4 shows the stationary mode.

We'll modify the model once again, making the queue bounded. We'll obtain a model of the M/M/2/m type. This can serve as a model for the operation of

network switches. For them, packet loss due to buffer overflow is a key characteristic. Using the M/M/2/m simulation model, we'll examine the dependence of the probability of request loss on system load and buffer size. The full text of the simulation model program, with detailed comments for each instruction, is provided in Appendix C.

Compared to the M/M/2 model, the queue has a finite size. Before creating the service process, the condition current_users < (2 + max_queue) is checked. If the queue is full, the stats['lost'] counter is incremented, and the request is ignored. All counters are moved inside the run_simulation function to prevent simulations from interfering with each other during the loop. Instead of fixed parameters, the arrival rate is calculated using a variable load factor $\rho = \lambda / n\mu$. The model uses the stats dictionary to collect request loss data for plotting with matplotlib for visual comparisons across different queue sizes.

Figure 5.7 shows a graph of the probability of request loss depending on the system load for different values of the queue size.

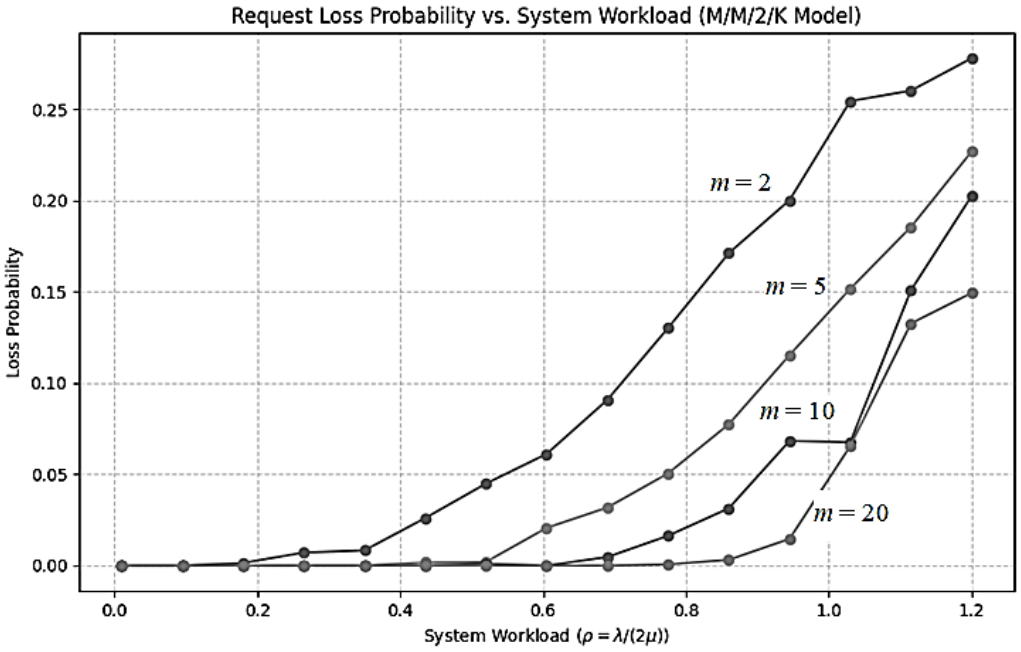


Figure 5.7 – Graph of the dependence $P_{loss} = f(\rho, m)$

Using such graphs allows you to select the parameters of the modeled object to ensure a given QoS value.

REFERENCES

1. Левченко Л. О. Операційні системи [Електронний ресурс]: навч. посіб. / Л. О. Левченко, Ю. А. Тарнавський ; Київ : КПІ ім. Ігоря Сікорського, 2023. – 256 с.
2. Stanislav Szabo. Probabilistic model for airport runway safety areas / Stanislav Szabo, Peter Vittek, Jakub Kraus and others // Transport problems. – Vol. 12, Iss. 2, 2017. – P. 89–96.
3. Kleinrock L. Queueing Systems, Vol. 2: Computer Applications / Leonard Kleinrock ; New York : John Wiley and Sons, 1976. – 537 p.
4. Fundamentals of Queueing Theory / J. F. Shortle, J. M. Thompson, D. Gross, C. M. Harris. – 5th ed. Hoboken : John Wiley & Sons, 2018. – 592 p.
5. Литвинов А. Л. Розробка та дослідження ймовірнісних моделей оцінювання якості інформаційно-управляючих систем / А. Л. Литвинов // Комун. госп-во міст : наук.-техн. зб. / Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2016. – Вип. 126. – С. 22–26. – (Серія: Технічні науки та архітектура).
6. Litvinov A. L. Probabilistic Modeling of the Training Test Subdivision of Device Production / A. L. Litvinov // Science and Education a New Dimension. Natural and Technical Scieces, IX(31). – Issue 250, 2021. – Pp. 42–45.
7. Гнєденко Б. В. Курс теорії ймовірностей : підручник / Б. В. Гнєденко ; Київ : ВПЦ «Київський університет», 2010. – 464 с.
8. Дороговцев А. Я. Теорія ймовірностей та математична статистика : підручник / А. Я. Дороговцев ; Київ : Либідь, 1994. – 384 с.
9. Литвинов А. Л. Вища та прикладна математика з елементами інформаційних технологій (теорія ймовірностей, математична статистика, математичне програмування, управління запасами) : навч. посіб. / А. Л. Литвинов ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2019. – 228 с.
10. Донченко В. С. Теорія ймовірностей та математична статистика для соціальних наук: навчальний посібник / В. С. Донченко, М. В.-С. Сидоров ; Київ : ВПС Київський університет, 2015. – 400 с.
11. Поняття процесу [Електрон. ресурс] – Електрон. текст. дані. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81, вільний (дата звернення: 15.01.2026). – Назва з екрана.

12. Karlin S. A first course in stochastic processes / S. Karlin ; New York and London : Academic press, 1968. – 536 p.

13. Литвинов А. Л. Теорія систем масового обслуговування : навч. посібник / А. Л. Литвинов ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 141 с.

14. Khintchine A. Y. Mathematical Methods in the Theory of Queueing / A. Y. Khintchine ; London : Charles Griffin & Co., Ltd, 1960. – 120 p.

15. Jaiswal N. K. Priority Queues / N. K. Jaiswal ; York and London : Academic press, 1968. – 280 p.

16. Самойленко А. М. Диференціальні рівняння : підручник / А. М. Самойленко, М. О. Перестюк, І. О. Парасюк ; Київ : Либідь, 2003. – 600 с.

17. Klimov G. P. Stochastic Theory of Service Systems / G. P. Klimov ; Oxford / New York : Pergamon Press, 1967. – 243 p.

18. Riordan J. Stochastic Service Systems / J. Riordan ; York and London : John Wiley and Sons, 1962. – 152 p.

19. Яровий А. Т. Методи оптимізації [Електронний ресурс] : навч. посіб. для здобувачів першого (бакалавр.) рівня вищ. освіти спец. 111 Математика, 113 Прикладна математика, 123 Комп'ютерна інженерія / А. Т. Яровий, Є. М. Страхов, О. Б. Васильєв. – Електрон. текст. дані (1 файл : 1,5 МБ). – Одеса : Одес. нац. ун-т ім. І. І. Мечникова, 2025. – 152 с. – Режим доступу: https://dspace.onu.edu.ua/items/89fea292-ad03-4b6c-8067-df5053270439, вільний (дата звернення: 29.05.2026). – Назва з екрана.

20. Kaufmann A. Les files d'attente: gestion des files d'attente / A. Kaufmann, R. Cruon ; Paris : Dunod, 1961. – 304 p.

21. Cox D. R. Renewal Theory / D. R. Cox ; New York : John Wiley & Sons Inc., 1967. – 142.

22. Shuangbao P. W. Computer Architecture and Security: Fundamentals of Designing Secure Computer Systems / P. W. Shuangbao, R. S. Ledley ; New York : John Wiley and Sons, 2013. – 360 p.

23. Jiang R. Introduction to Quality and Reliability Engineering / R. Jiang ; Berlin : Springer, 2015. – 618 p.

24. David D. Y. Stochastic Modeling and Analysis of Manufacturing Systems / D. Y. David ; New York : Springer, 2013. – 360 p.

25. Чисельні методи: навч. посіб. / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков ; Вінницький національний аграрний університет. – Вінниця : ВНАУ, 2020. – 322 с.

26. Shannon R. E. Systems simulation: the art and science / R. E. Shannon ; Englewood Cliffs, NJ : Prentice-Hall, 1975. – 387 p.

27. Зінов'єва, О. Г. Імітаційне моделювання та моделювання систем [Текст] : навч. посіб. / О. Г. Зінов'єва, С. В. Шаров, І. П. Паламарчук ; Запоріжжя : ФОП Однорог Т. В., 2025. – 203 с.

ANNEX A

Table of values of the Laplace function $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \exp(-z^2/2)dz$

X	*Φ(X)*	*X*	*Φ(X)*	*X*	*Φ(X)*	*X*	*Φ(X)*	*X*	*Φ(X)*	*X*	*Φ(X)*	*X*	*Φ(X)*
0,00	0,0000	0,39	0,1517	0,78	0,2823	1,17	0,3790	1,56	0,4406	1,95	0,4744	2,38	0,4913
0,01	0,0040	0,40	0,1554	0,79	0,2852	1,18	0,3810	1,57	0,4418	1,96	0,4750	2,40	0,4918
0,02	0,0080	0,41	0,1591	0,80	0,2881	1,19	0,3830	1,58	0,4429	1,97	0,4756	2,42	0,4922
0,03	0,0120	0,42	0,1628	0,81	0,2910	1,20	0,3849	1,59	0,4441	1,98	0,4761	2,44	0,4927
0,04	0,0160	0,43	0,1664	0,82	0,2939	1,21	0,3869	1,60	0,4452	1,99	0,4767	2,46	0,4931
0,05	0,0199	0,44	0,1700	0,83	0,2967	1,22	0,3883	1,61	0,4463	2,00	0,4772	2,48	0,4934
0,06	0,0239	0,45	0,1736	0,84	0,2995	1,23	0,3907	1,62	0,4474	2,01	0,4772	2,50	0,4938
0,07	0,0279	0,46	0,1772	0,85	0,3023	1,24	0,3925	1,63	0,4484	2,02	0,4783	2,52	0,4941
0,08	0,0319	0,47	0,1808	0,86	0,3051	1,25	0,3944	1,64	0,4495	2,03	0,4788	2,54	0,4945
0,09	0,0359	0,48	0,1844	0,87	0,3078	1,26	0,3962	1,65	0,4505	2,04	0,4793	2,56	0,4948
0,10	0,0398	0,49	0,1879	0,88	0,3106	1,27	0,3980	1,66	0,4515	2,05	0,4798	2,58	0,4951
0,11	0,0438	0,50	0,1915	0,89	0,3133	1,28	0,3997	1,67	0,4525	2,06	0,4803	2,60	0,4953
0,12	0,0478	0,51	0,1950	0,90	0,3159	1,29	0,4015	1,68	0,4535	2,07	0,4807	2,62	0,4956
0,13	0,0517	0,52	0,1985	0,91	0,3186	1,30	0,4032	1,69	0,4545	2,08	0,4812	2,64	0,4959
0,14	0,0557	0,53	0,2019	0,92	0,3212	1,31	0,4049	1,70	0,4554	2,09	0,4817	2,66	0,4961
0,15	0,0596	0,54	0,2054	0,93	0,3238	1,32	0,4066	1,71	0,4554	2,10	0,4821	2,68	0,4963
0,16	0,0636	0,55	0,2088	0,94	0,3264	1,33	0,4082	1,72	0,4573	2,11	0,4826	2,70	0,4965
0,17	0,0675	0,56	0,2123	0,95	0,3289	1,34	0,4099	1,73	0,4582	2,12	0,4830	2,72	0,4967
0,18	0,0714	0,57	0,2157	0,96	0,3315	1,35	0,4115	1,74	0,4591	2,13	0,4834	2,74	0,4969
0,19	0,0753	0,58	0,2190	0,97	0,3340	1,36	0,41311	1,75	0,4599	2,14	0,4838	2,76	0,4971
0,20	0,0793	0,59	0,2224	0,98	0,3365	1,37	0,4147	1,76	0,4608	2,15	0,4842	2,78	0,4973
0,21	0,0832	0,60	0,2257	0,99	0,3389	1,38	0,4162	1,77	0,4616	2,16	0,4846	2,80	0,4974
0,22	0,0871	0,61	0,2291	1,00	0,3413	1,39	0,4177	1,78	0,4525	2,17	0,4850	2,82	0,4976
0,23	0,0910	0,62	0,2324	1,01	0,3438	1,40	0,4192	1,79	0,4633	2,18	0,4854	2,84	0,4977
0,24	0,0948	0,63	0,2357	1,02	0,3401	1,41	0,4207	1,80	0,4641	2,19	0,4857	2,86	0,4979
0,25	0,0987	0,64	0,2389	1,03	0,3485	1,42	0,4222	1,81	0,4649	2,20	0,4861	2,88	0,4980
0,26	0,1026	0,65	0,2422	1,04	0,3508	1,43	0,4236	1,82	0,4656	2,21	0,4864	2,90	0,4981
0,27	0,1064	0,66	0,2454	1,05	0,3531	1,44	0,4251	1,83	0,4664	2,22	0,4868	2,92	0,4982
0,28	0,1103	0,67	0,2486	1,06	0,3554	1,45	0,4265	1,84	0,467!	2,23	0,4871	2,94	0,4984
0,29	0,1141	0,68	0,2517	1,07	0,3577	1,46	0,4279	1,85	0,4678	2,24	0,4875	2,96	0,4985
0,30	0,1179	0,69	0,2549	1,08	0,3599	1,47	0,4292	1,86	0,4686	2,25	0,4878	2,98	0,4985
0,31	0,1217	0,70	0,2580	1,09	0,3621	1,48	0,4305	1,87	0,4693	2,26	0,4881	3,00	0,49865
0,32	0,1255	0,71	0,2611	1,10	0,3643	1,49	0,4319	1,88	0,4699	2,27	0,4884	3,20	0,49931
0,33	0,1293	0,72	0,2642	1,11	0,3665	1,50	0,4332	1,89	0,4706	2,28	0,4887	3,40	0,49966
0,34	0,1331	0,73	0,2673	1,12	0,3686	1,51	0,4345	1,90	0,4713	2,29	0,4890	3,60	0,49984
0,35	0,1368	0,74	0,2703	1,13	0,3708	1,52	0,4357	1,91	0,4719	2,30	0,4893	3,80	0,49992
0,36	0,1406	0,75	0,2734	1,14	0,3729	1,53	0,4370	1,92	0,4726	2,32	0,4898	4,00	0,49996
0,37	0,1443	0,76	0,2764	1,15	0,3749	1,54	0,4382	1,93	0,4732	2,34	0,4904	4,50	0,49999
0,38	0,1480	0,77	0,2794	1,16	0,3770	1,55	0,4394	1,94	0,4738	2,36	0,4909	5,00	0,49999

ANNEX B

Critical distribution points

Number of degrees of freedom	Level of significance					
	0,01	0,025	0,05	0,95	0,97	0,89
1	6,6	5,0	3,8	0,0039	0,00098	0,00016
2	9,2	7,4	6,0	0,103	0,051	0,020
3	11,3	9,4	7,8	0,352	0,216	0,115
4	13,3	11,1	9,5	0,711	0,484	0,297
5	15,1	12,8	9,5	1,15	0,831	0,554
6	16,8	14,4	12,6	1,64	1,24	0,872
7	18,5	16,0	14,1	2,17	1,69	1,24
8	20,1	17,5	15,5	2,73	2,18	1,65
9	21,7	19,0	16,9	3,33	2,70	2,09
10	23,2	20,5	18,3	3,94	3,25	2,56
11	24,7	21,9	19,7	4,57	3,82	3,5
12	26,2	23,3	21,0	5,23	4,40	3,57
13	27,7	24,7	22,4	5,89	5,01	4,11
14	29,1	26,1	23,7	6,57	5,63	4,66
15	30,6	27,5	25,0	7,26	6,26	5,23
16	32,0	28,8	26,3	7,96	6,91	5,81
17	33,4	30,2	27,6	8,67	7,56	6,41
18	34,8	31,4	28,9	9,39	8,23	7,01
19	36,2	32,9	30,1	10,1	8,91	7,63
20	37,6	34,2	31,4	10,9	9,59	8,26
21	38,9	35,5	32,7	11,6	10,3	8,90
22	40,3	36,8	33,9	12,3	11,0	9,54
23	41,6	38,1	35,2	13,1	11,7	10,2
24	43,0	39,4	36,4	13,8	12,4	10,9
25	44,3	40,6	37,7	14,6	13,1	11,5
26	45,6	41,9	38,9	15,4	13,8	12,2
27	47,0	43,2	40,1	16,2	14,6	12,9
28	48,3	44,5	41,3	16,9	15,3	13,6
29	49,6	45,7	42,6	17,7	16,0	14,3
30	50,9	47,0	43,8	18,5	16,8	15,0

ANNEX C

Simulation model of the M/M/2/m queuing system[2]

```
# Import the simpy library to handle discrete-event simulation processes
import simpy
# Import the numpy library to generate random numbers and handle math arrays
import numpy as np
# Import the matplotlib pyplot module to create charts and visualize data
import matplotlib.pyplot as plt

# Define a function to execute a single simulation with specific parameters
def run_simulation(lamb, mu, max_queue, sim_time=1000):
    """
    Runs a single simulation run to calculate the percentage of lost requests.
    """
    # Create an instance of the SimPy simulation environment
    env = simpy.Environment()
    # Create a resource representing 2 parallel processing servers
    servers = simpy.Resource(env, capacity=2)

    # Initialize a dictionary to count total incoming and lost requests
    stats = {'total': 0, 'lost': 0}

    # Define an inner function representing the lifecycle of request service
    def service_process(env, resource, mu):
        # Request access to the server resource using a context manager
        with resource.request() as request:
            # Pause execution until one of the two servers becomes available
            yield request
            # Simulate service duration using an exponential distribution
            yield env.timeout(np.random.exponential(1 / mu))

    # Define an inner function that acts as a continuous stream of requests
    def stream(env, lamb, resource, mu, max_queue):
        # Start an infinite loop to constantly generate incoming requests
        while True:
```

[2] This program was developed with partial use of generative models of artificial intelligence

```
            # Wait for a random interval before the next request arrives
            yield env.timeout(np.random.exponential(1 / lamb))
            # Increment the total counter for requests that reached the system
            stats['total'] += 1

            # Calculate total requests currently in service and in queue
            current_users = resource.count + len(resource.queue)
            # Define total system capacity as 2 servers plus maximum queue size
            system_capacity = 2 + max_queue

            # Check if the number of current requests is below system capacity
            if current_users < system_capacity:
                # Spawn a new parallel asynchronous service process for this request
                env.process(service_process(env, resource, mu))
            # Execute this block if the system is completely full
            else:
                # Increment the counter for dropped (lost) requests
                stats['lost'] += 1

    # Register the request stream generator process in the environment
    env.process(stream(env, lamb, servers, mu, max_queue))
    # Execute the simulation until the specified simulation time limit
    env.run(until=sim_time)

    # Return calculated loss probability if total requests are above zero
    return stats['lost'] / stats['total'] if stats['total'] > 0 else 0

# Set a fixed service rate parameter for an individual server
mu_param = 1.0
# Create an array of different maximum queue sizes to be tested
queue_sizes = [0, 2, 5, 10]
# Generate 15 linear points between 0.2 and 1.5 for system workload (rho)
rho_values = np.linspace(0.2, 1.5, 15)

# Initialize a new figure window with a width of 10 and height of 6 inches
plt.figure(figsize=(10, 6))

# Begin a loop to iterate through each configured queue size limit
```

```
for q_size in queue_sizes:
    # Initialize an empty list to store loss results for the current queue size
    loss_probabilities = []
    # Begin a nested loop to iterate through each system workload value
    for rho in rho_values:
        # Calculate arrival rate lambda dynamically based on rho and mu parameters
        lamb_param = rho * 2 * mu_param

        # Call the simulation function and save the returned loss probability
        loss_prob = run_simulation(lamb_param, mu_param, max_queue=q_size,
sim_time=2000)
        # Append the calculated probability to the results list for this line
        loss_probabilities.append(loss_prob)

    # Plot a line chart of loss probabilities vs workload for this queue size
    plt.plot(rho_values, loss_probabilities, marker='o', label=f'Queue capacity =
{q_size}')

# Set the main title text at the top of the generated figure window
plt.title('Request Loss Probability vs. System Workload (M/M/2/K Model)')
# Set the text label for the horizontal axis using LaTeX notation for rho
plt.xlabel('System Workload ($\\rho = \\lambda / (2\\mu)$)')
# Set the text label for the vertical axis of the chart
plt.ylabel('Loss Probability')
# Enable background grid lines with a dashed style and partial transparency
plt.grid(True, linestyle='--', alpha=0.7)
# Display the color-coded legend box mapped to the plotted data lines
plt.legend()
# Render and display the final chart onto the user screen
plt.show()
```

Електронне наукове видання

ЛИТВИНОВ Анатолій Леонідович

ТЕОРІЯ СИСТЕМ МАСОВОГО ОБСЛУГОВУВАННЯ. ТЕОРІЯ ТА ЗАСТОСУВАННЯ

МОНОГРАФІЯ

(Англ. мовою)

Відповідальний за випуск *М. М. Булаєнко*

Редактор *О. В. Михаленко*

Комп’ютерне верстання *А. Л. Литвинов, Є. Г. Панова*

Підп. до друку 29.05.2026. Формат 60×84/16.
Ум. друк. арк. 10,3.

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова,
вул. Чорноглазівська, 17, Харків, 61002.
Електронна адреса: office@kname.edu.ua
Свідоцтво суб’єкта видавничої справи:
ДК № 8386 від 14.07.2025.

Anatoliy Leonidovich LITVINOV – Doctor of Engineering Sciences, Professor. His specialization is modeling, computer technology and information systems. He has published approximately 200 scientific papers, inventions, textbooks and monographs.