Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)
Факультет «Робототехника и комплексная автоматизация»
Кафедра «Системы автоматизированного проектирования»

Домашнее задание №3 по дисциплине
«Теория вероятностей и математическая статистика»

Вариант 14

Выполнил:
студент группы РК6-36Б
Петраков С.А.

Москва
2020

Оглавление

Условие.

Известно, что среднее время между звонками клиентов составляет $T_c = R1 + G1 + B1$ секунд, а среднее время обслуживания $T_s = R2$ секунд. Все потоки случайных событий считать пуассоновскими. Если все операторы заняты, звонок теряется.

R1 = 11	**R2 = 10**	**R3 = 5**
G1 = 10	**G2 = 9**	**G3 = 1**
B1 = 11	**B2 = 10**	**B3 = 1**

Известно, что среднее время между звонками клиентов составляет $T_c = T_{\text{заявки}} = 32$ секунд, а среднее время обслуживания $T_s = T_{\text{обслуживания}} = 10$ секунд. Все потоки случайных событий считать пуассоновскими. Если все операторы заняты, звонок теряется.

1. Рассмотреть систему без очереди. Построить графики от числа операторов: вероятности отказа (вплоть до обеспечения отказов менее 1%); математического ожидания числа занятых операторов; коэффициента загрузки операторов.

Пусть:

- число каналов – n.
- Частоты обслуживания заявок: $\mu = \frac{1}{T_{\text{обслуживания}}} \frac{\text{заявки}}{\text{секунду}}$
- Частота появления новой заявки: $\lambda = \frac{1}{T_{\text{заявки}}} \frac{\text{заявок}}{\text{секунду}}$
- Интенсивность нагрузки системы: $\rho = \frac{\lambda}{\mu} = \frac{T_{\text{обслуживания}}}{T_{\text{заявки}}}$

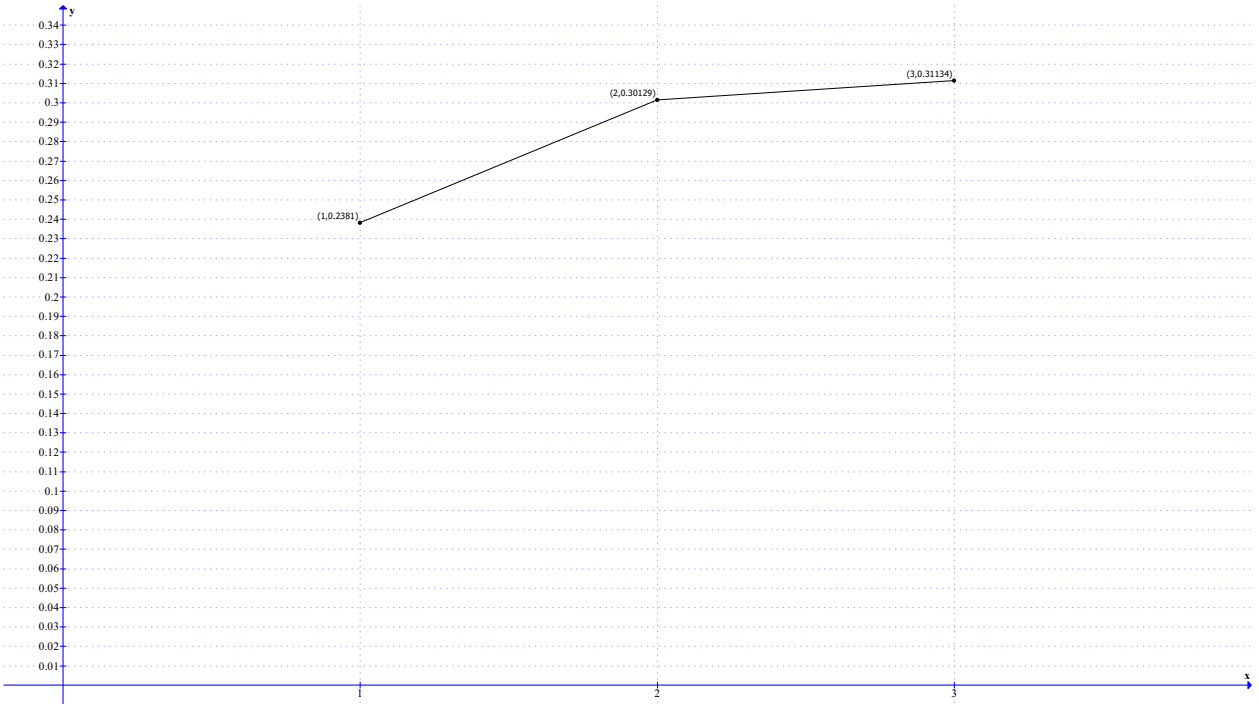
$$P_0 = \left(\sum_{i=0}^{n} \frac{\rho^i}{i!} \right)^{-1}$$

  - Вероятность отказа:

$$P_{\text{отк}} = P_n = \frac{\rho^n}{n!} * P_0$$

- Математическое ожидание числа занятых операторов

$$M = \sum_{i=0}^{n} i * P_i \qquad P_i = \frac{\rho^i}{i!} * P_0$$



- Коэффициент загрузки операторов

$$K_{\text{загрузки операторов}} = \frac{\overline{N_{\text{зан}}}}{n} = \frac{\rho Q}{n} = \frac{\rho(1 - p_{\text{отк}})}{n}$$

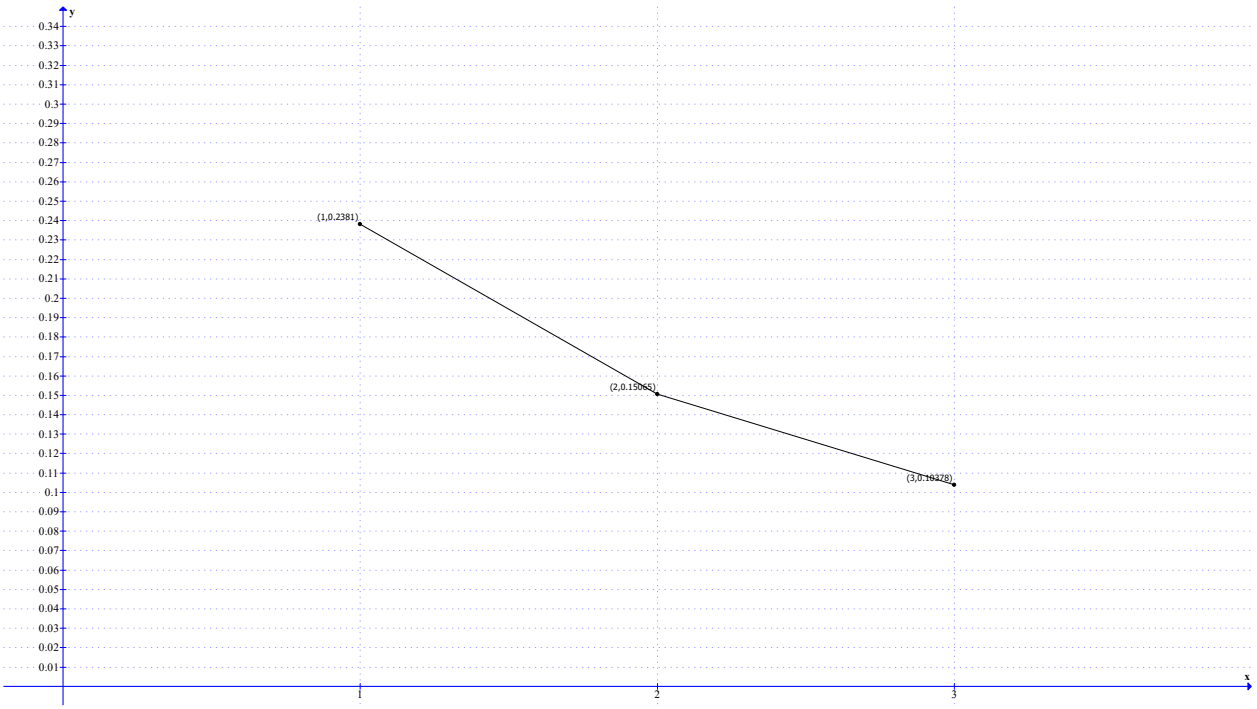


Таблица значений:

Количество каналов	Вероятность отказов	Математическое ожидание	Коэффициент загрузки операторов
1	0,2381	0,2381	0,2381
2	0,03587	0,30129	0,15065
3	0,00372	0,31134	0,10378

2. **Рассмотреть систему с ограниченной очередью. Варьируя число операторов (вплоть до числа каналов, соответствующего 1% отказов в системе без очереди), построить семейства графиков от числа мест в очереди: вероятности отказа; математического ожидания числа занятых операторов; коэффициента загрузки операторов; вероятности существования очереди; математического ожидания длины очереди; коэффициента занятости мест в очереди.**

Пусть:

- число каналов – n.
- число мест в очереди – m.
- Частоты обслуживания заявок: $\mu = \frac{1}{T_{\text{обслуживания}}} \frac{\text{заявки}}{\text{секунду}}$
- Частота появления новой заявки: $\lambda = \frac{1}{T_{\text{заявки}}} \frac{\text{заявок}}{\text{секунду}}$
- Интенсивность нагрузки системы: $\rho = \frac{\lambda}{\mu} = \frac{T_{\text{обслуживания}}}{T_{\text{заявки}}}$

$$P_0 = \left(\sum_{k=0}^{n} \frac{\rho^k}{k!} + \frac{\rho^{n+1}}{n! * (n-\rho)} * \left(1 - \frac{\rho^m}{n}\right) \right)^{-1}$$

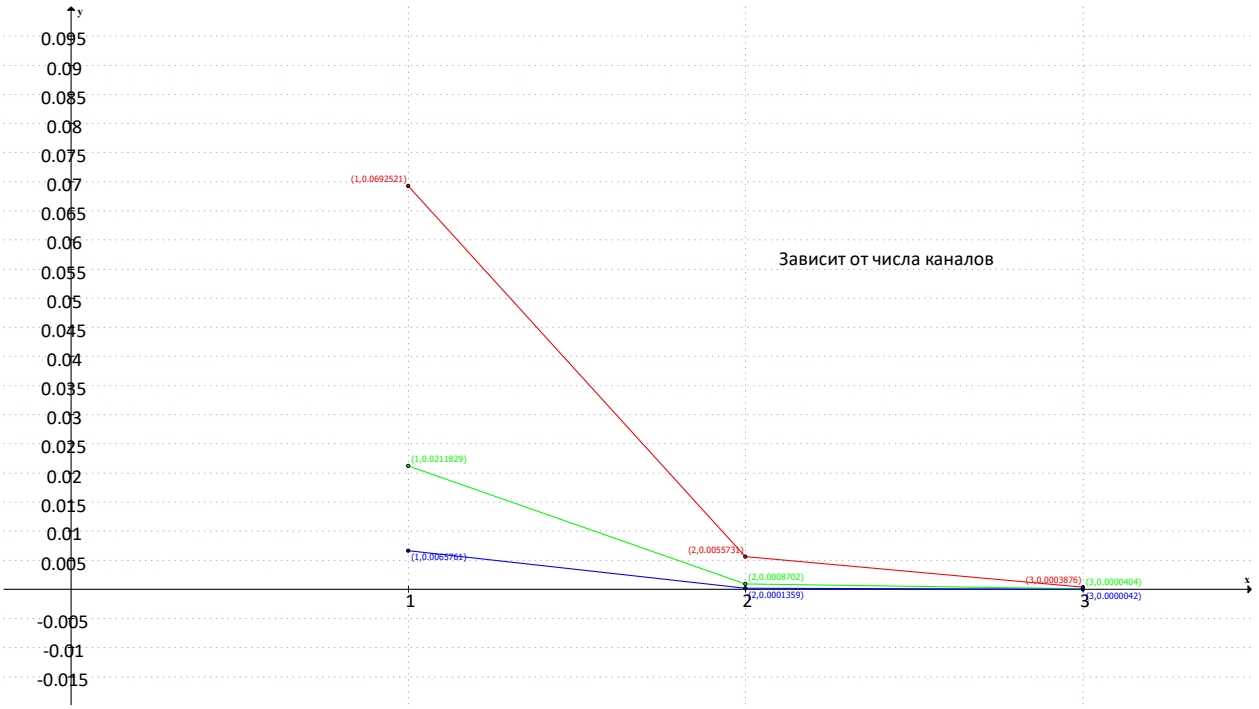
Для графиков:
Красный – 1 канал(место в очереди)

Зеленый – 2 канала(место в очереди)

Синий – 3 канала(место в очереди)

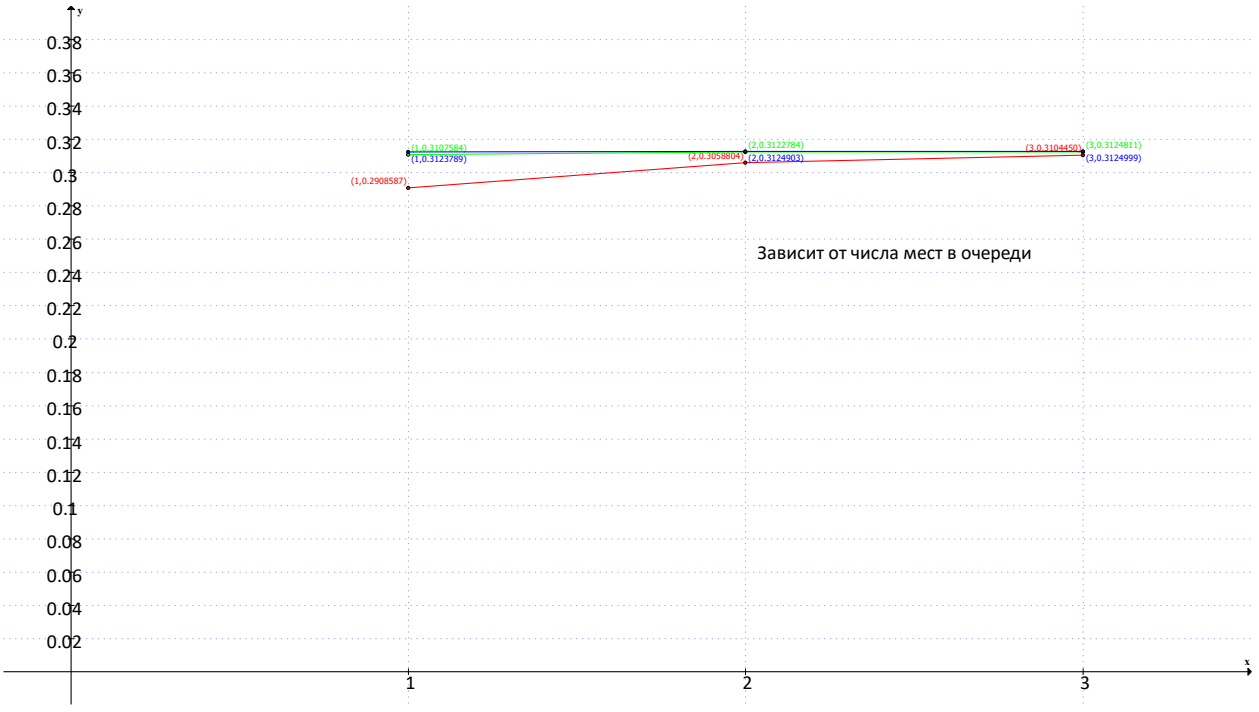
- Вероятность отказа

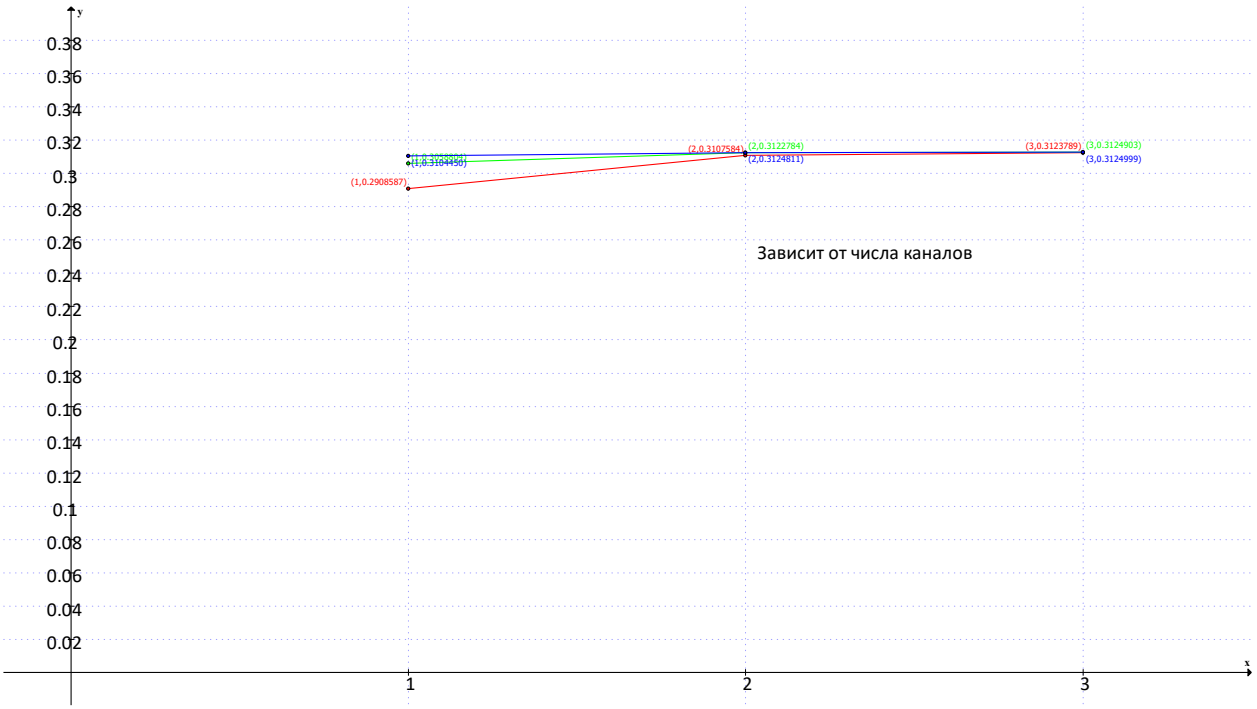
$$P_{\text{отказа}} = P_{n+m} = \frac{\rho^{n+m}}{n^m * n!} * P_0$$



- Математическое ожидания числа занятых операторов

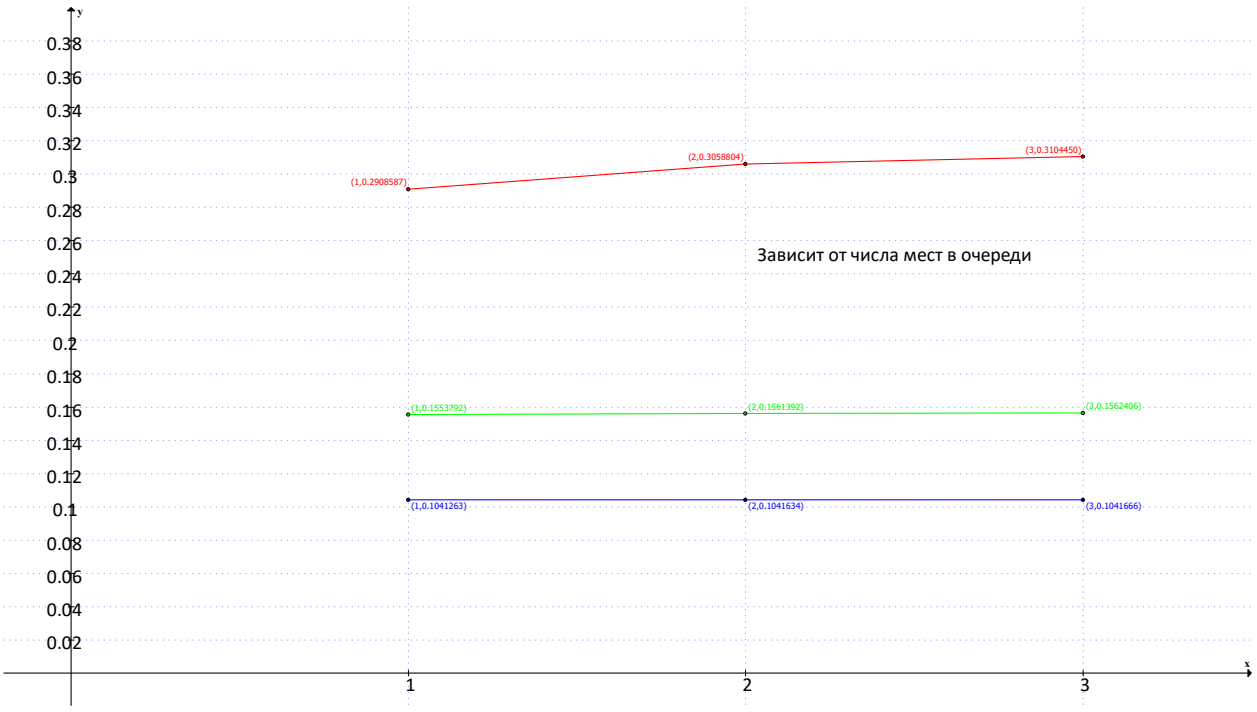
$$M_{\text{загрузки операторов}} = \sum_{i=1}^{n} i * P_i + \sum_{j=1}^{m} n * P_{n+j}$$

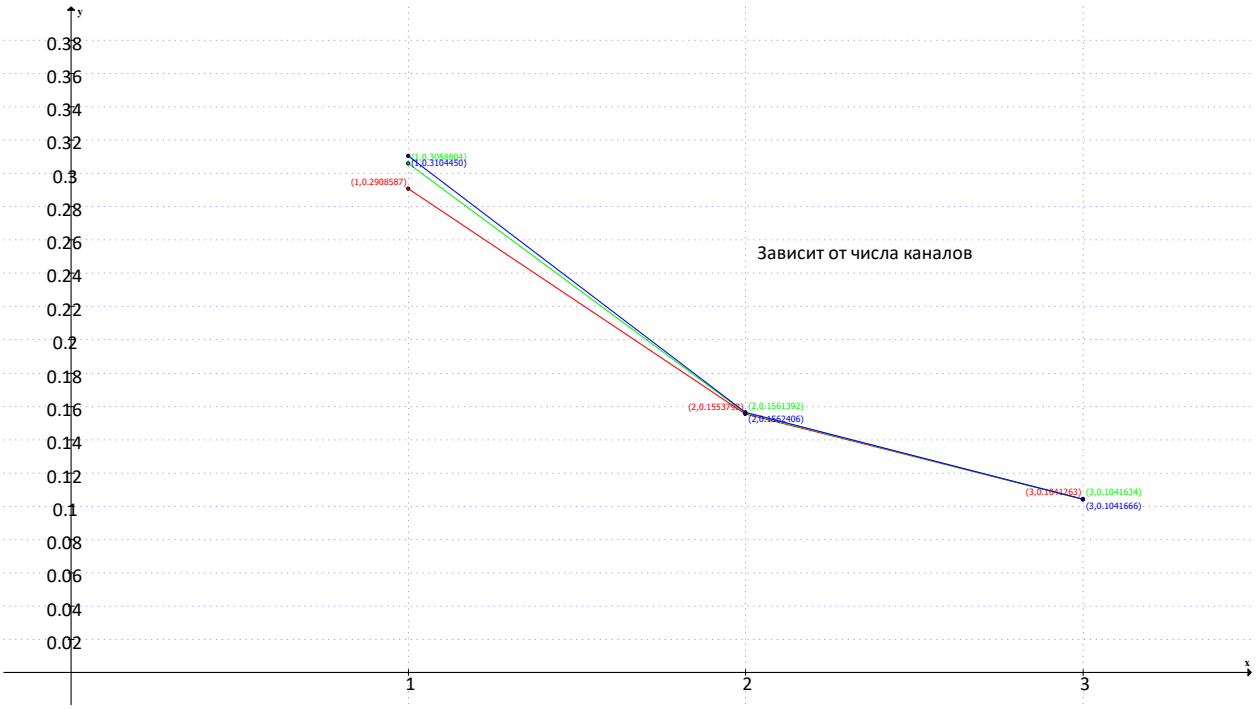




- Коэффициент загрузки операторов

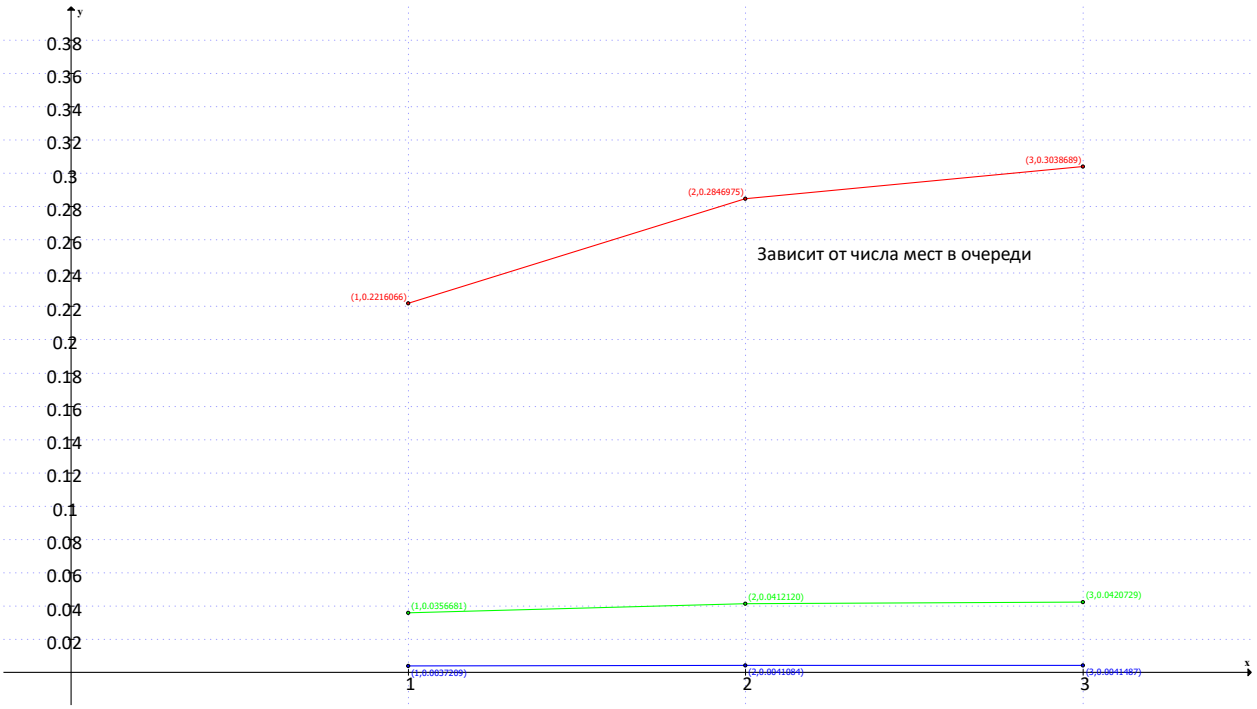
$$K_{\text{загрузки операторов}} = \frac{M_{\text{загрузки операторов}}}{n}$$

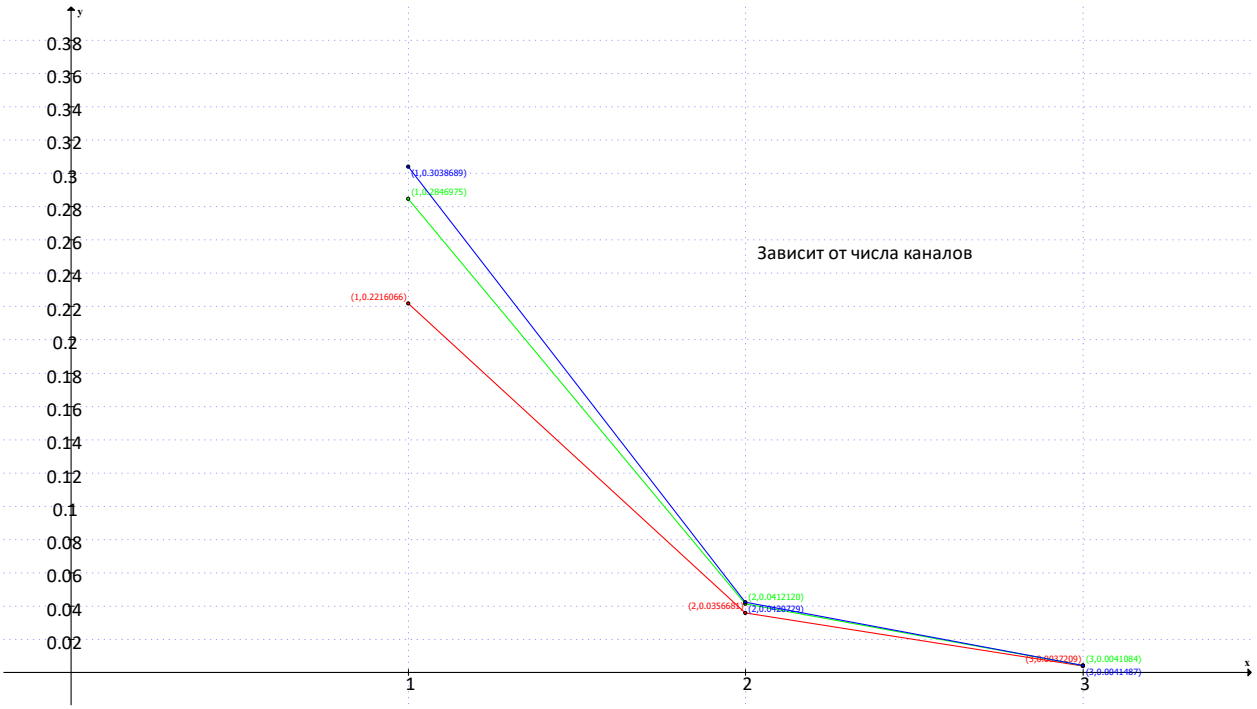




- Вероятности существования очереди

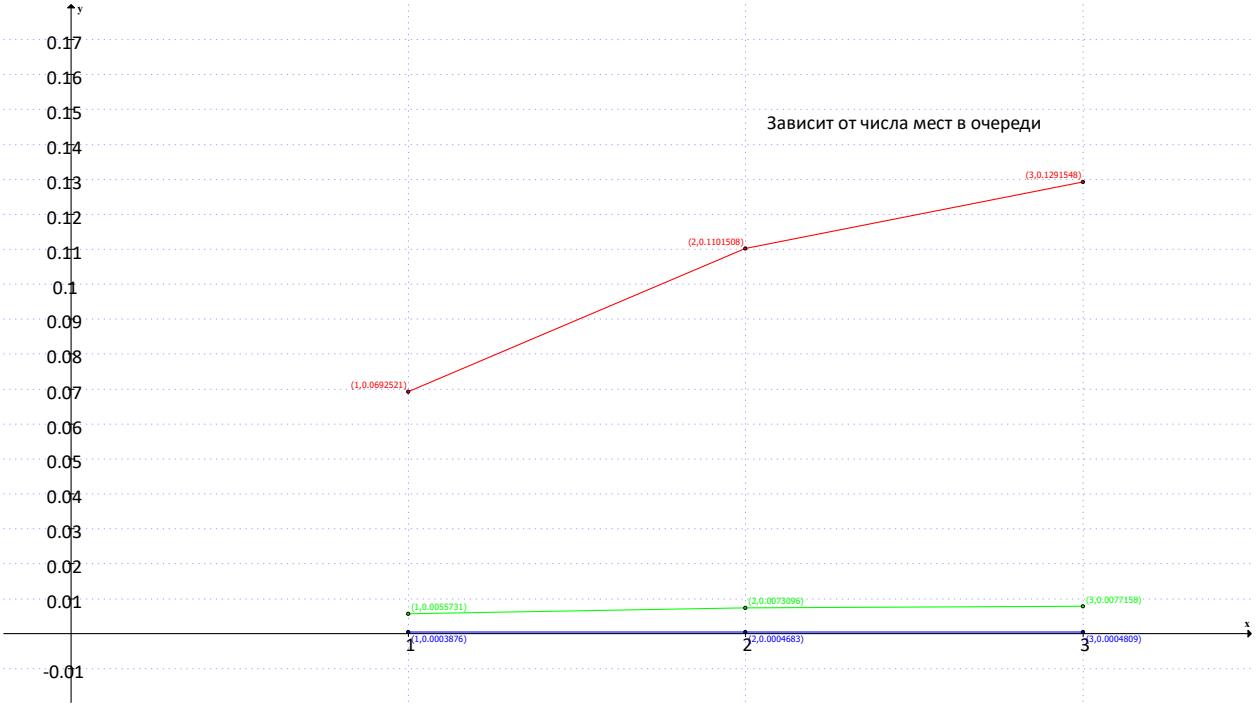
$$P_{\text{существования очереди}} = \frac{\rho^n}{n!} * \frac{1-\left(\frac{\rho}{n}\right)^m}{1-\frac{\rho}{n}} * P_0$$

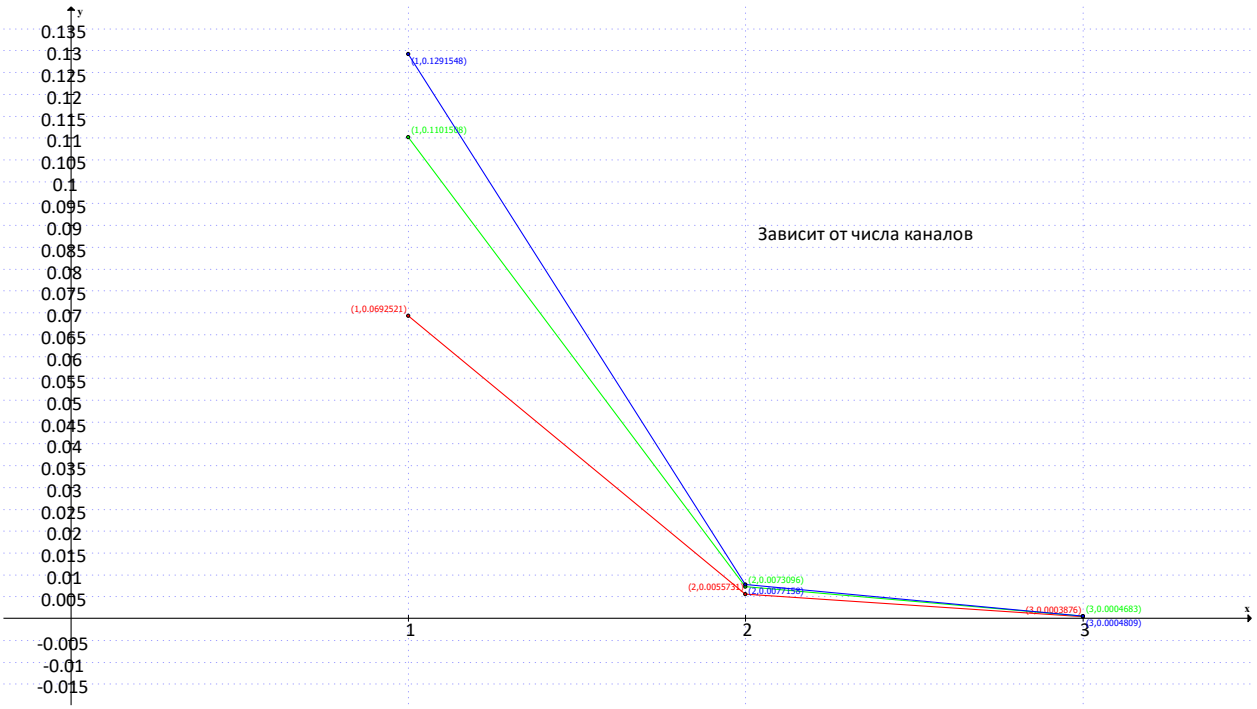




- Математическое ожидание длины очереди

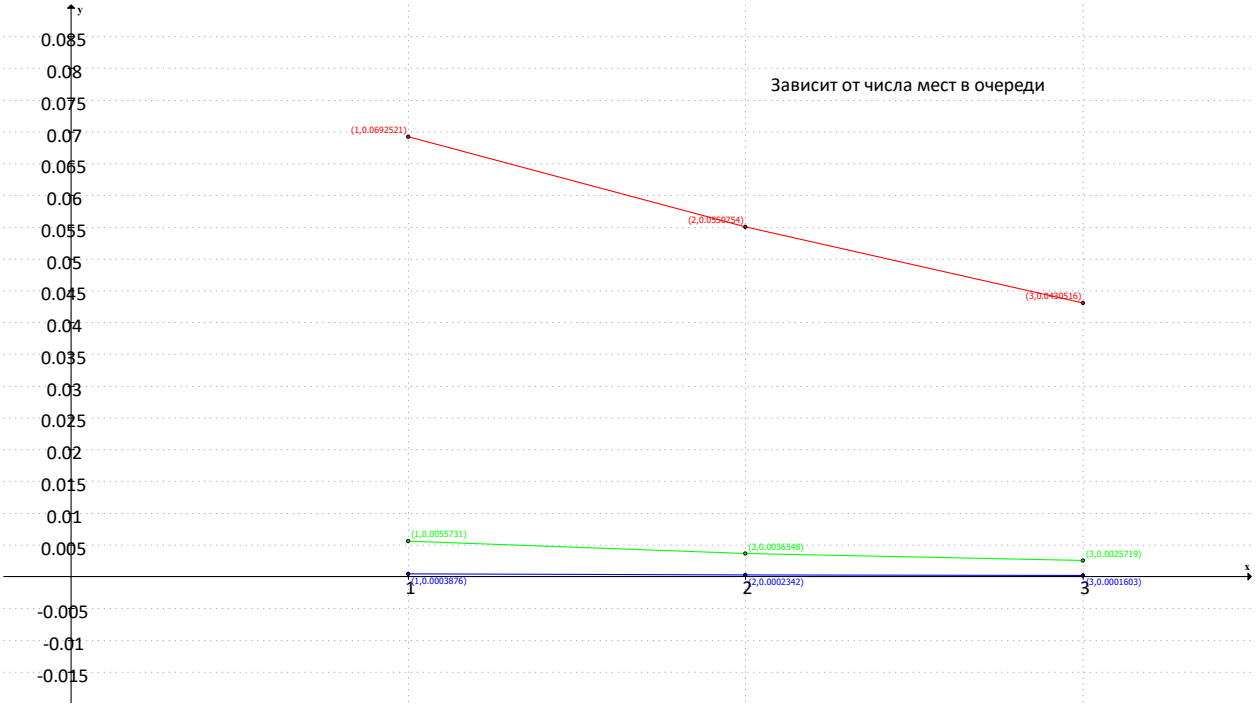
$$M_{\text{Загрузки очереди}} = \sum_{i=1}^{m} i * P_{n+i}$$

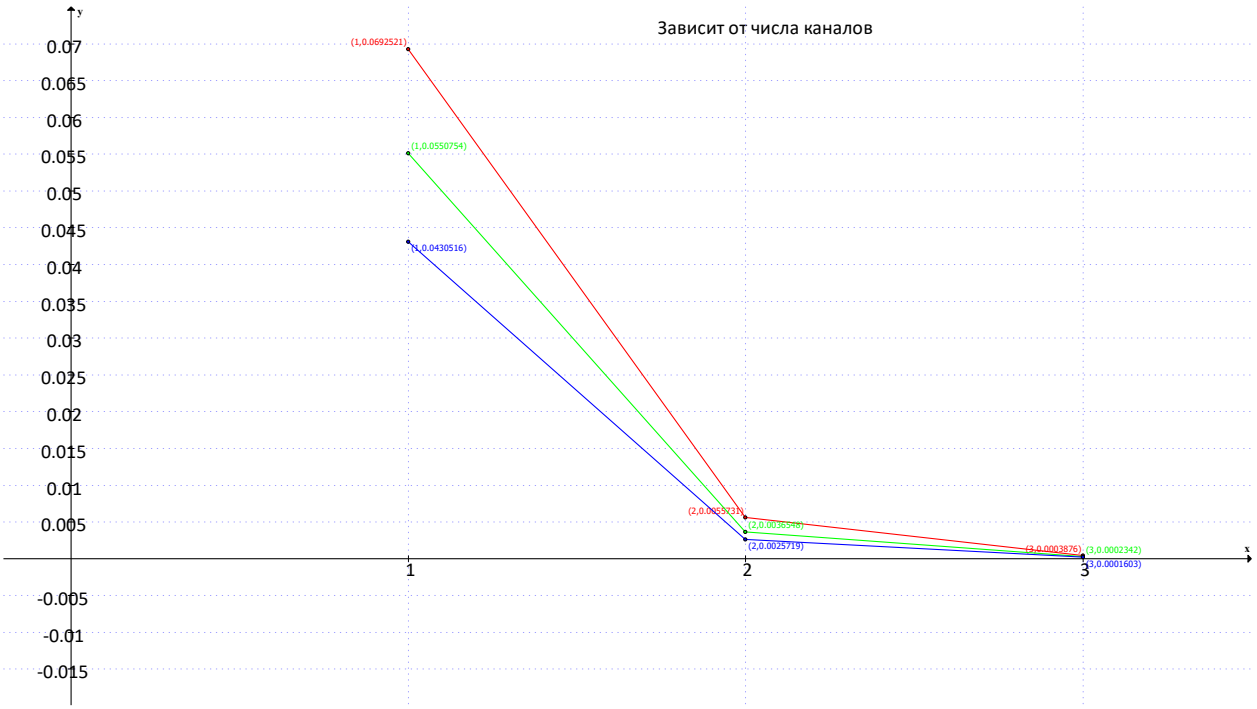




- Коэффициента занятости мест в очереди.

$$K_{\text{занятости очереди}} = \frac{M_{\text{Загрузки очереди}}}{m}$$





Табличные данные:

Операторов	Мест в очереди	Вероятность отказа	Математическое ожидание числа занятых операторов	Коэффициент загрузки операторов	Вероятность существования очереди	Математическое ожидание длины очереди	Коэффициент занятости мест в очереди
1	1	0.0692521	0.2908587	0.2908587	0.2216066	0.0692521	0.0692521
1	2	0.0211829	0.3058804	0.3058804	0.2846975	0.1101508	0.0550754
1	3	0.0065761	0.3104450	0.3104450	0.3038689	0.1291548	0.0430516
2	1	0.0055731	0.3107584	0.1553792	0.0356681	0.0055731	0.0055731
2	2	0.0008702	0.3122784	0.1561392	0.0412120	0.0073096	0.0036548
2	3	0.0001359	0.3124811	0.1562406	0.0420729	0.0077158	0.0025719
3	1	0.0003876	0.3123789	0.1041263	0.0037209	0.0003876	0.0003876
3	2	0.0000404	0.3124903	0.1041634	0.0041084	0.0004683	0.0002342
3	3	0.0000042	0.3124999	0.1041666	0.0041487	0.0004809	0.0001603

3. **Рассмотреть систему без ограничений на длину очереди. Построить графики от числа операторов (вплоть до числа каналов, соответствующего 1% отказов в системе без очереди): математического ожидания числа занятых операторов; коэффициента загрузки операторов; вероятности существования очереди; математического ожидания длины очереди.**

Пусть:

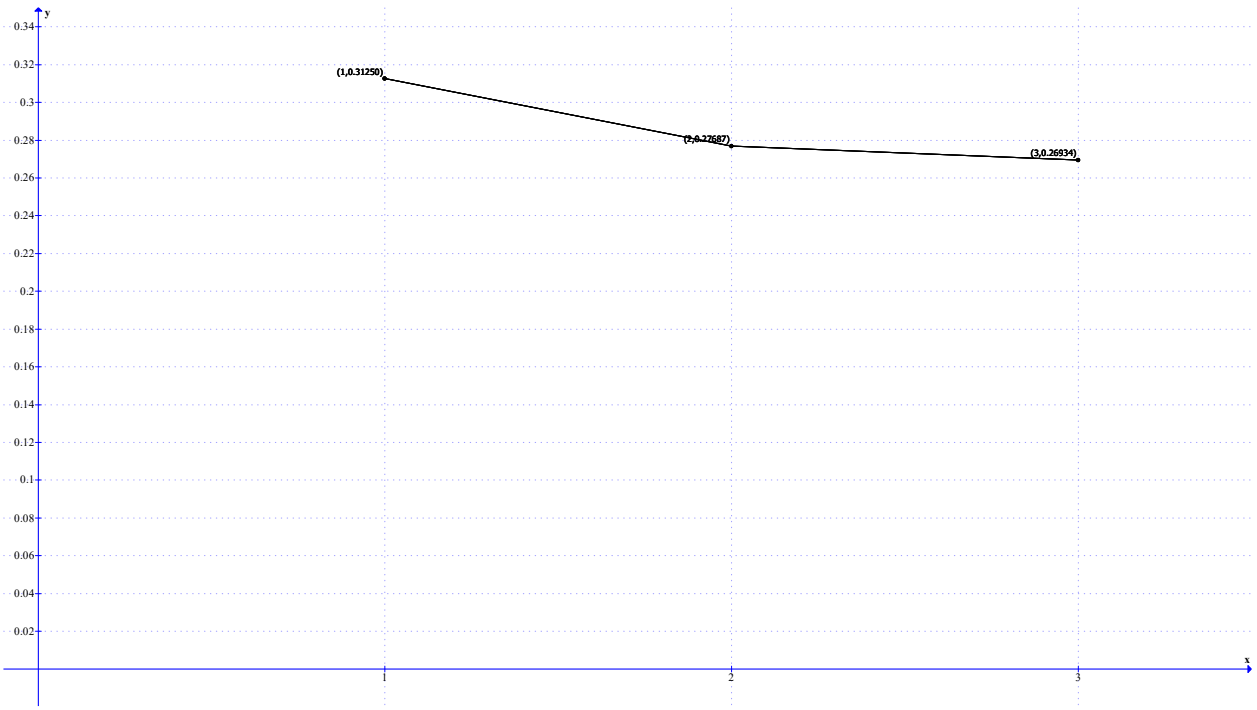
- число каналов – n.
- число мест в очереди – m.
- Частоты обслуживания заявок: $\mu = \frac{1}{T_{\text{обслуживания}}} \frac{\text{заявки}}{\text{секунду}}$
- Частота появления новой заявки: $\lambda = \frac{1}{T_{\text{заявки}}} \frac{\text{заявок}}{\text{секунду}}$
- Интенсивность нагрузки системы: $\rho = \frac{\lambda}{\mu} = \frac{T_{\text{обслуживания}}}{T_{\text{заявки}}}$

$$P_0 = \left(1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \cdots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{n!\,(n-\rho)}\right)^{-1}$$

- Математическое ожидание числа занятых операторов;

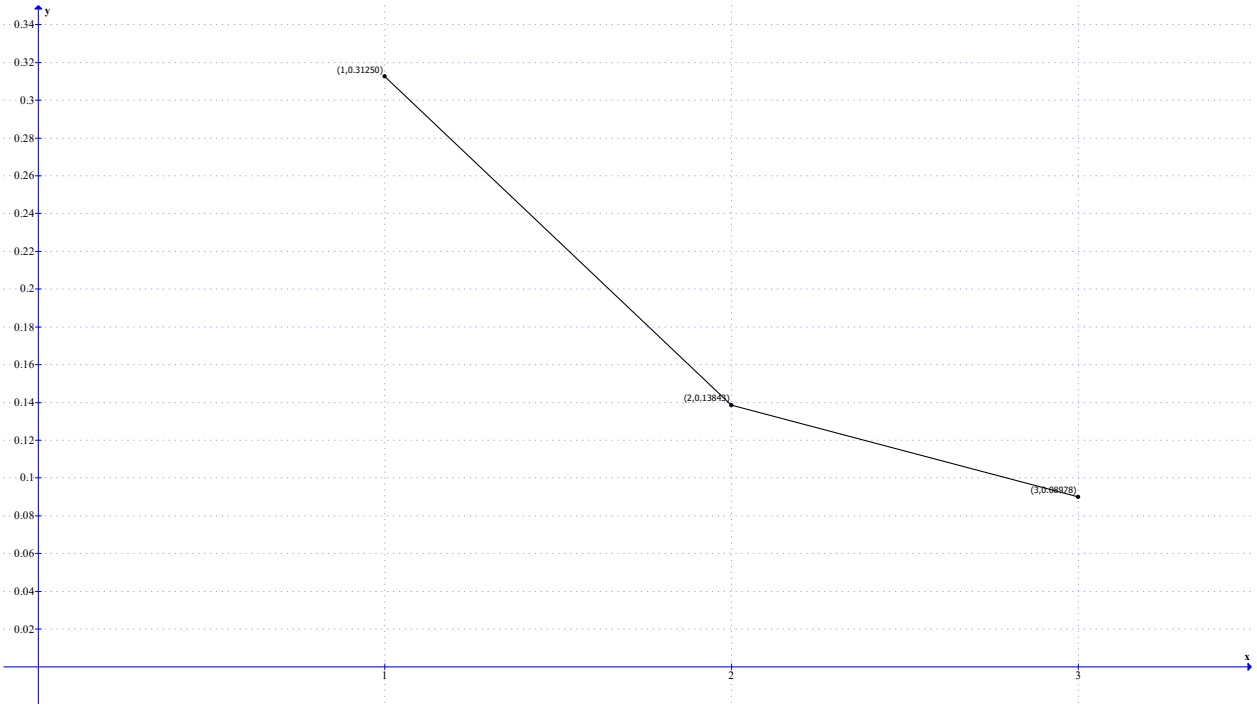
$$M_{\text{загрузки операторов}} = \sum_{i=0,j=0}^{n,\infty} i \cdot P_{i+j} = \sum_{i=0}^{n} iP_i + n\sum_{j=1}^{\infty} P_{n+j}$$

$$= P_0 \sum_{i=0}^{n} \frac{i\rho^i}{i!} + nP_n \sum_{j=1}^{\infty} \left(\frac{\rho}{n}\right)^j$$



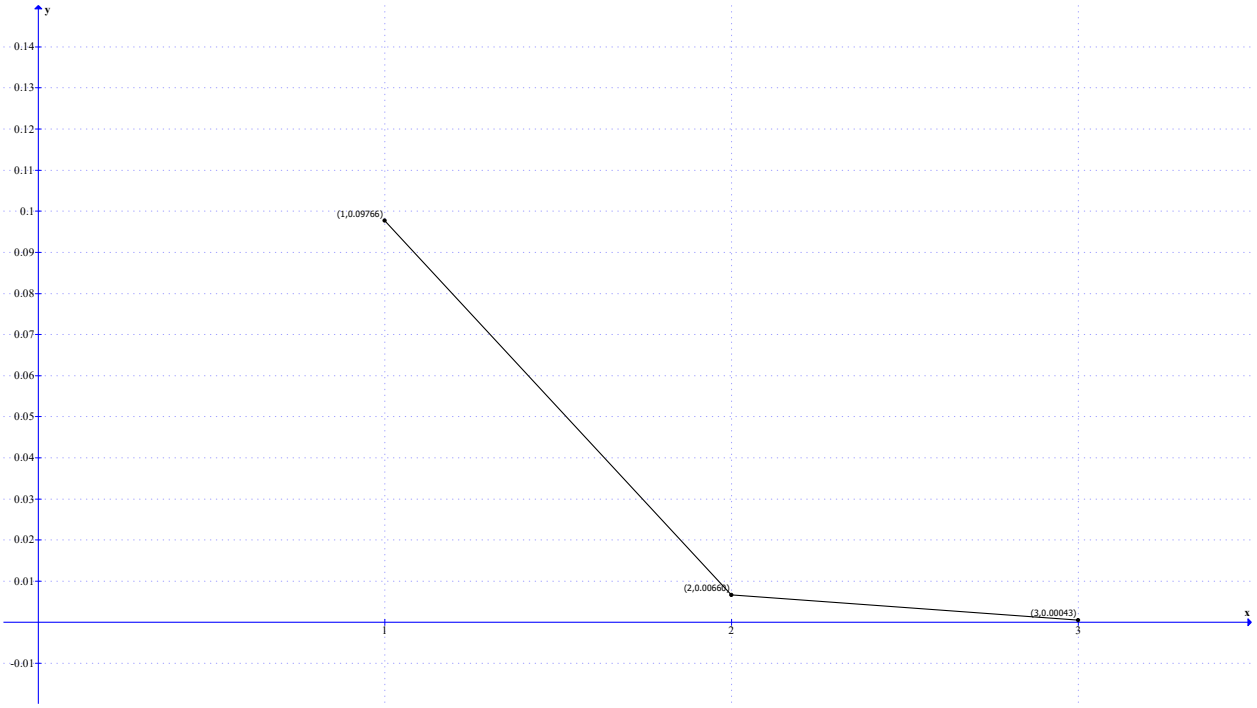
- Коэффициент загрузки операторов;

$$K_{\text{загрузки операторов}} = \frac{M_{\text{загрузки операторов}}}{n}$$



- Вероятность существования очереди;

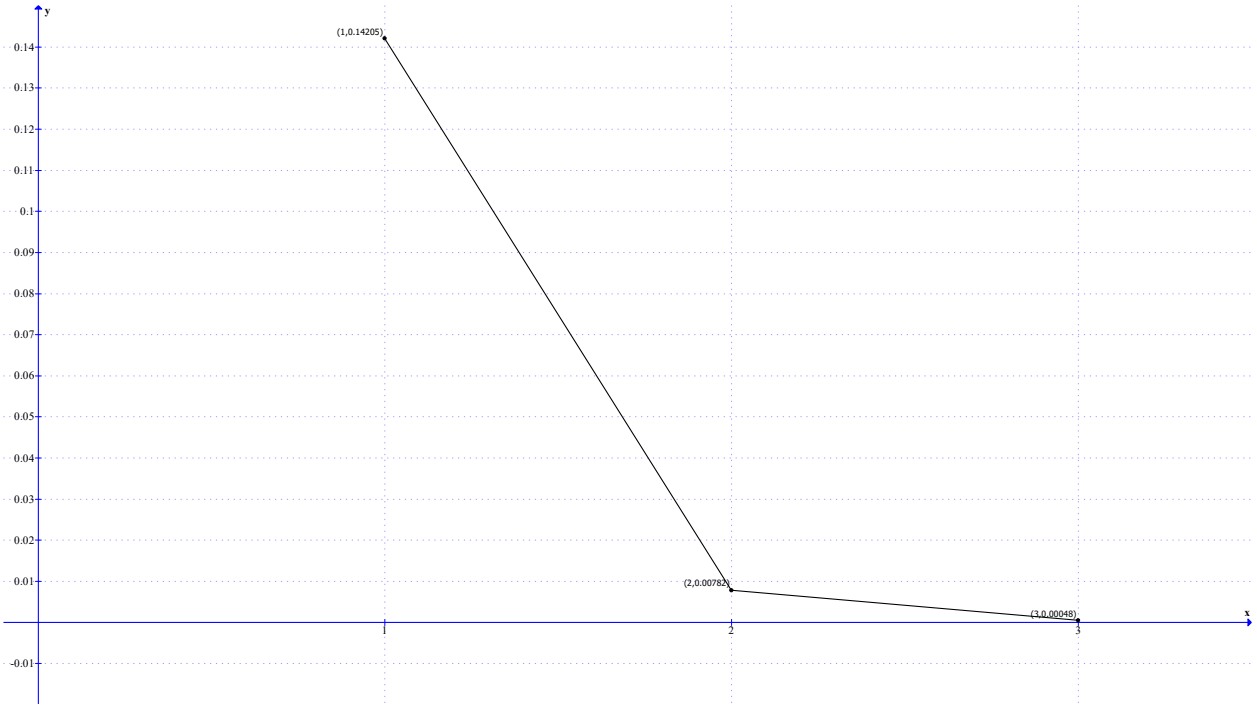
$$P_{\text{существования очереди}} = \sum_{j=1}^{\infty} P_{n+j} = P_n * \sum_{j=1}^{\infty} \left(\frac{\rho}{n}\right)^j$$



- Математической ожидание длины очереди;

$$M_{\text{Загрузки очереди}} = \sum_{j=1}^{\infty} jP_{n+j} = P_n \sum_{j=1}^{\infty} j\left(\frac{\rho}{n}\right)^j =$$

$$= P_n \sum_{j=1}^{\infty} j\left(\frac{\rho}{n}\right)^{j-1} = P_n \frac{d}{d\frac{\rho}{n}} \sum_{j=1}^{\infty} \left(\frac{\rho}{n}\right)^j = P_n a \frac{d\frac{\frac{\rho}{n}}{1-\frac{\rho}{n}}}{d\frac{\rho}{n}} = P_n \frac{\rho}{n} \frac{1}{\left(1-\frac{\rho}{n}\right)^2}$$



Табличные данные:

Каналов	Математическое ожидание числа занятых операторов	Коэффициент загрузки операторов	Вероятность существования очереди	Математической ожидание длины очереди
1	0.31250	0.31250	0.09766	0.14205
2	0.27687	0.13843	0.00660	0.00782
3	0.26934	0.08978	0.00043	0.00048

4. **Рассмотреть систему без ограничений на длину очереди, учитывающей фактор ухода клиентов из очереди (среднее приемлемое время ожидания – $\mathbf{T_w} = \boldsymbol{T}_{\text{ожидания}} = \mathbf{R3} + \mathbf{G3} + \mathbf{B3} = \mathbf{7}$ секунд). Построить графики от числа операторов (вплоть до числа каналов, соответствующего 1% отказов в системе без очереди): математического ожидания числа занятых операторов; коэффициента загрузки операторов; вероятности существования очереди; математического ожидания длины очереди.**

Пусть:

- число каналов – n.
- число мест в очереди – m.
- Частоты обслуживания заявок: $\mu = \frac{1}{T_{\text{обслуживания}}} \frac{\text{заявки}}{\text{секунду}}$
- Частота появления новой заявки: $\lambda = \frac{1}{T_{\text{заявки}}} \frac{\text{заявок}}{\text{секунду}}$
- Интенсивность нагрузки системы: $\rho = \frac{\lambda}{\mu} = \frac{T_{\text{обслуживания}}}{T_{\text{заявки}}}$

$$\nu = \frac{1}{T_w} = \frac{1}{T_{\text{ожидания}}}$$

$$P_{i+k} = \frac{\rho^i}{i!} \prod_{j=1}^{k} \frac{\frac{1}{T_{\text{заявки}}}}{\frac{n}{T_{\text{обслуживания}}} + \frac{k}{T_{\text{ожидания}}}} P_0$$

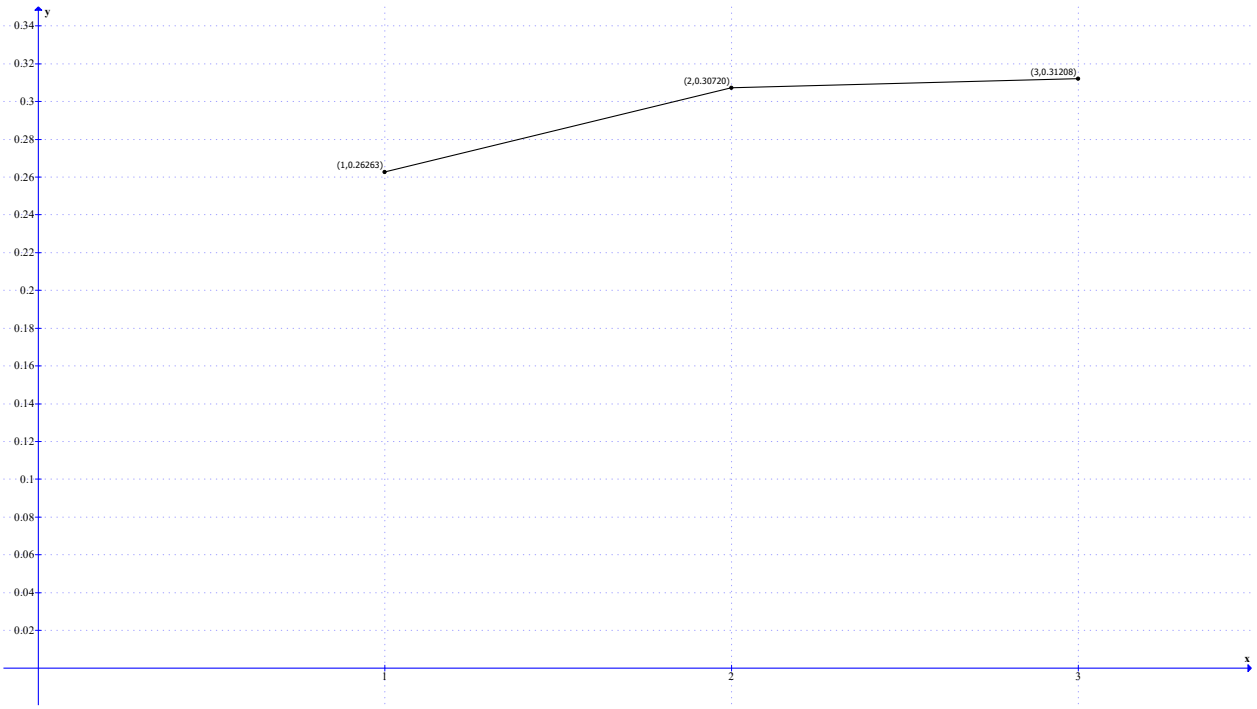
Тогда:

$$P_0 = \frac{1}{\sum_{i=0}^{n} \frac{\rho^i}{i!} + \frac{\rho^n}{n!} \sum_{j=1}^{\infty} \prod_{k=1}^{j} \frac{\frac{1}{T_{\text{заявки}}}}{\frac{n}{T_{\text{обслуживания}}} + \frac{k}{T_{\text{ожидания}}}}}$$

- Математическое ожидания числа занятых операторов;

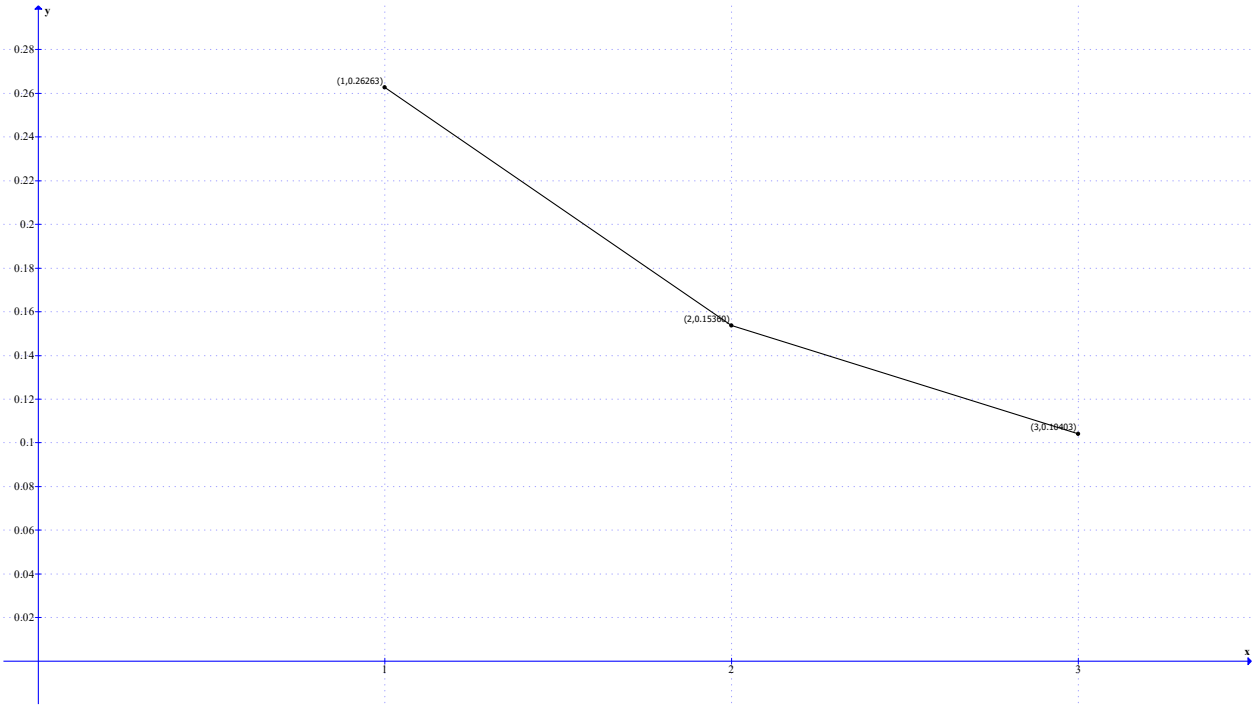
$$M = \sum_{i=0,j=0}^{n,\infty} i * P_{i+j} = \sum_{i=0}^{n} i * P_i + n * \sum_{j=1}^{\infty} P_{n+j}$$

$$= P_0 \sum_{i=0}^{n} \frac{i\rho^i}{i!} + nP_n \sum_{j=1}^{\infty} \prod_{k=1}^{j} \frac{\frac{1}{T_{\text{заявки}}}}{\frac{n}{T_{\text{обслуживания}}} + \frac{k}{T_{\text{ожидания}}}}$$



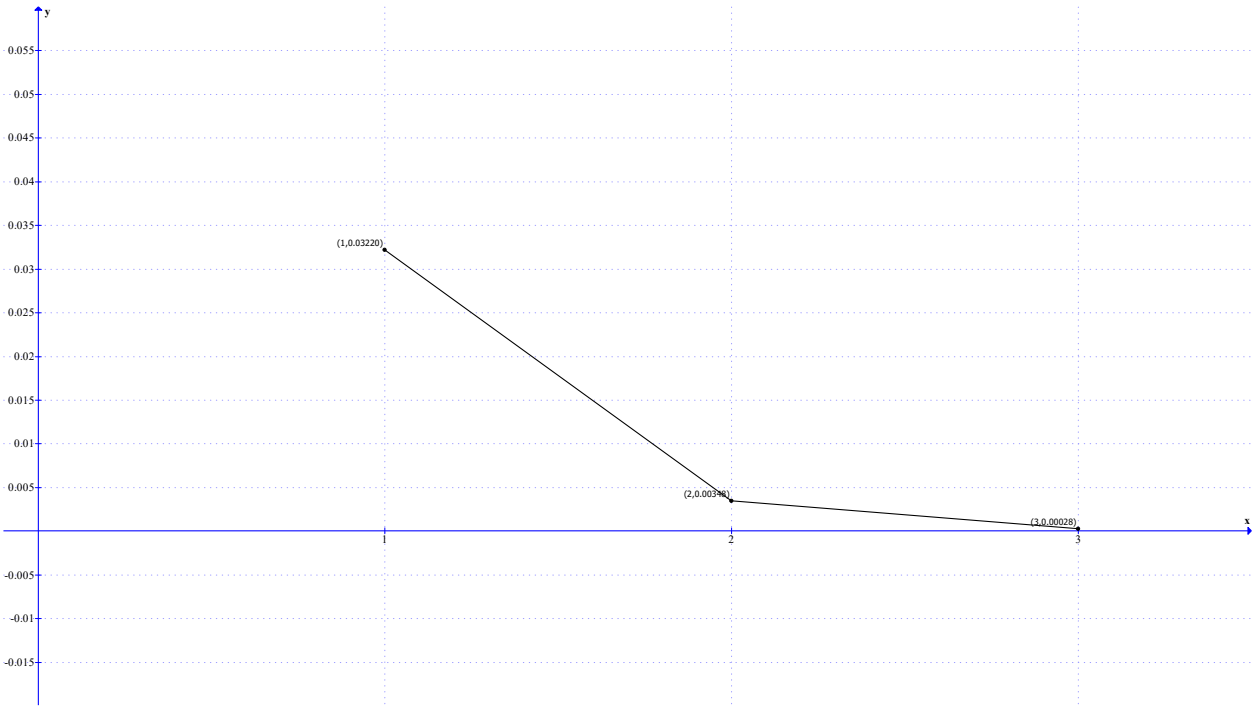
- Коэффициент загрузки операторов;

$$K_{\text{загрузки операторов}} = \frac{M}{n}$$



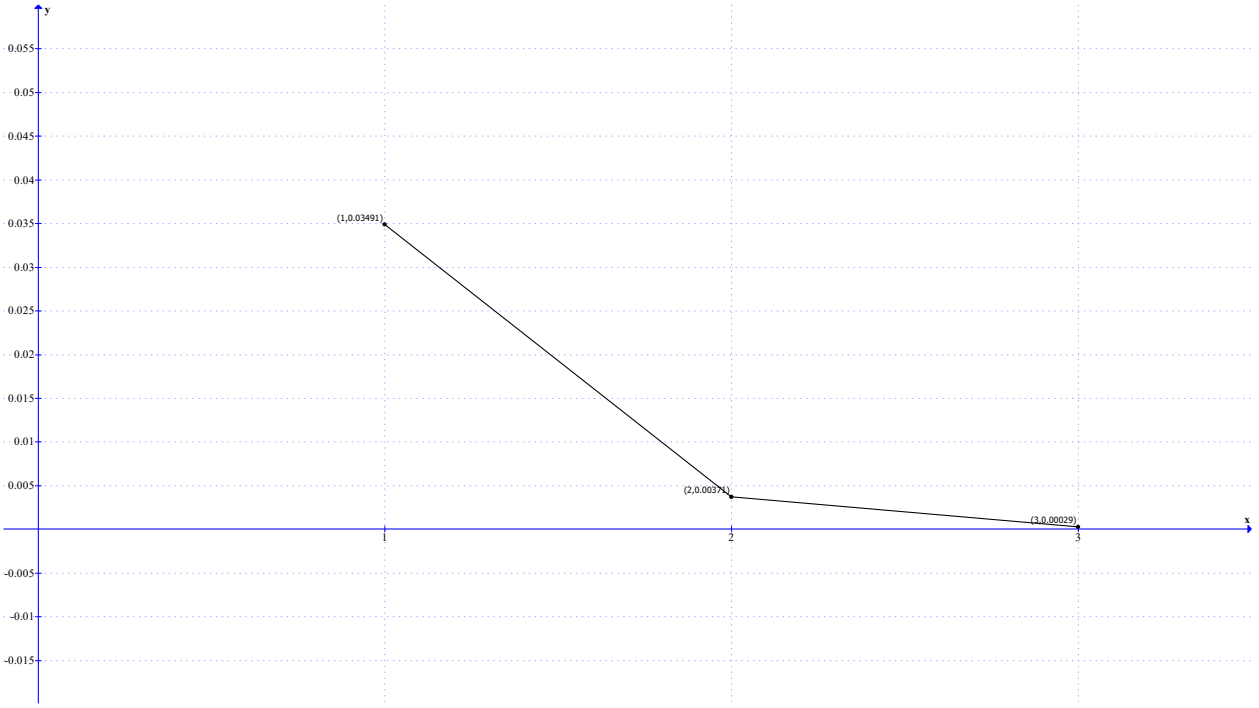
- Вероятность существования очереди;

$$P_{\text{существования очереди}} = \sum_{j=1}^{\infty} P_{n+j} = P_n \sum_{j=1}^{\infty} \prod_{k=1}^{j} \frac{\frac{1}{T_{\text{заявки}}}}{\frac{n}{T_{\text{обслуживания}}} + \frac{k}{T_{\text{ожидания}}}}$$



- Математическое ожидание длины очереди

$$M_{\text{Загрузки очереди}} = \sum_{j=1}^{\infty} jP_{n+j} = P_n \sum_{j=1}^{\infty} j \prod_{k=1}^{j} \frac{\frac{1}{T_{\text{заявки}}}}{\frac{n}{T_{\text{обслуживания}}} + \frac{k}{T_{\text{ожидания}}}}$$



Каналов	Математическое ожидание числа занятых операторов	Коэффициент загрузки операторов	Вероятность существования очереди	Математической ожидание длины очереди
1	0.26263	0.26263	0.03220	0.03491
2	0.30720	0.15360	0.00348	0.00371
3	0.31208	0.10403	0.00028	0.00029

Текст программы

```
#include <iostream>

#include <math.h>

#include <iomanip>

#define EPS 0e-7

using namespace std;


long double fact(int);


long double SumOfMult(long double maintenanceTime, long double appear-
anceTime, long double waitingTime, long double channels, int i)

{

	long double result = 0;

	for (int j = 1; j <= i; j++)

	{

		long double temp = 1;

		for (int k = 1; k <= j; k++)

		{

			temp *= (1 / appearanceTime) / (channels / maintenanceTime +
k / waitingTime);

		}

		result += temp;

	}

	return  result;


}

long double SumOfMultComlex(long double maintenanceTime, long double ap-
pearanceTime, long double waitingTime, long double channels, int i)

{

	long double result = 0;
```

```
	for (int j = 1; j <= i; j++)
	{
		long double temp = 1;
		for (int k = 1; k <= j; k++)
		{
			temp *= (1 / appearanceTime) / (channels / maintenanceTime + k / waitingTime);
		}
		result += j*temp;
	}
	return  result;

}
long double Punkt4P_0(long double maintenanceTime, long double appearanceTime, long double waitingTime, long double channels, int j)
{
	long double result = 0;
	for (int i = 0; i <= channels; i++)
	{
		result += powl(maintenanceTime / appearanceTime, i) / fact(i);
	}
	result += powl(maintenanceTime / appearanceTime, channels) * SumOfMult(maintenanceTime, appearanceTime, waitingTime, channels, j) / fact(channels);
	return 1 / result;

}
long double Punkt4MathExpectingChannels(long double maintenanceTime, long double appearanceTime, long double waitingTime, long double channels, int j, long double P_n, long double P_0)
```

```
{
	long double result = 0;
	for (int i = 0; i <= channels; i++)
	{
		result += i * powl(maintenanceTime / appearanceTime, i) / fact(i);
	}
	result *= P_0;
	result += channels * P_n * SumOfMult(maintenanceTime, appearanceTime,
waitingTime, channels, j);
	return result;

}
long double Punkt4P_queue(long double maintenanceTime, long double appear-
anceTime, long double waitingTime, long double channels, int j, long double P_n,
long double P_0)
{
	return P_n*SumOfMult(maintenanceTime, appearanceTime, waitingTime,
channels, j);

}
long double Punkt4MathExpectingQueues(long double maintenanceTime, long
double appearanceTime, long double waitingTime, long double channels, int j,
long double P_n, long double P_0)
{
	return P_n * SumOfMultComlex(maintenanceTime, appearanceTime, wait-
ingTime, channels, j);

}
long double fact(int n)
{
	long double r = 1;
```

```
	for (int i = 2; i <= n; ++i)
		r *= i;
	return r;
}
void Punkt1(long double maintenanceTime, long double appearanceTime)
{
	int channels = 1; // Кол-во каналов
	long double P_denial; // Вероятность отказов
	long double MathExpecting; // Математическое ожидание числа занятых операторов
	long double load_Factor; // Коэффициент загрузки операторов
	long double P_0;
	long double ro = maintenanceTime / appearanceTime; // Коэффициент загрузки СМО
	cout << "Канал\t\t";
	cout << "Отказ\t\t";
	cout << "Матож\t\t";
	cout << "Коэф\n";
	bool flag = true;
	do
	{

#pragma region P_0
		P_0 = 0;
		MathExpecting = 0;
		for (int i = 0; i <= channels; i++)
		{
			P_0 += powl(ro, i) / fact(i);
		}
```

```
            P_0 = 1 / P_0;
#pragma endregion

#pragma region Вероятность отказа
            P_denial = P_0 * powl(ro, (long double)channels) / fact(channels);
#pragma endregion

#pragma region Мат ожидание
            for (int i = 0; i <= channels; i++)
            {
                MathExpecting += i * P_0 * powl(ro, i) / fact(i);
            }
#pragma endregion

#pragma region Коэф. загрузки операторов
            load_Factor = (1 - P_denial) * ro / channels;
#pragma endregion

#pragma region Вывод
            cout << setprecision(5) << channels << "\t\t";
            cout << setprecision(5) << P_denial << "\t\t";
            cout << setprecision(5) << MathExpecting << "\t\t";
            cout << setprecision(5) << load_Factor << "\n";
#pragma endregion
            if (P_denial <= 0.01)
                flag = false;
            channels++;
        } while (flag);
}
```

```
void Punkt2(long double maintenanceTime, long double appearanceTime, long double maxChannels)
{
	long double P_0 = 0.0;
	long double P_denial;
	long double MathExpectingChannels; // Мат ожидание Каналов
	long double MathExpectingQueues; // Мат ожидание очереди
	long double P_queue; // Вероятность образования очереди
	long double loadFactorChannels; // Коэф. загрузки операторов
	long double loadFactorQueues; // Коэф. загрузки очереди

	long double ro = maintenanceTime / appearanceTime; // Коэффициент загрузки СМО
#pragma region Вывод для таблички
	cout << "Операт\t";
	cout << "Очередь\t";
	cout << "Отказ\t";
	cout << "МатОжОп\t";
	cout << "КоэфОп\t";
	cout << "ВерОч\t";
	cout << "МатОжОч\t";
	cout << "КоэфОч\n";
#pragma endregion
	for (int channels = 1; channels <= maxChannels; channels++)
	{

		int queue = 1;
		bool flag = true;
		do
```

```
                {
#pragma region Инициализация
                        P_0 = 0;
                        P_denial = 0;
                        MathExpectingChannels = 0;
                        MathExpectingQueues = 0;
                        P_queue = 0;
                        loadFactorChannels = 0;
                        loadFactorQueues = 0;
#pragma endregion

#pragma region P_0
                        for (int i = 0; i <= channels; i++)
                        {
                                P_0 += powl(ro, i) / fact(i);
                        }
                        P_0 += powl(ro, (long double)channels + 1) * (1 - powl(ro,
queue) / channels) / (fact(channels) * (channels - ro));
                        P_0 = 1 / P_0;
#pragma endregion

#pragma region Вероятность отказов
                        P_denial = P_0 * powl(ro, (long double)queue + (long dou-
ble)channels) / (powl(channels, queue) * fact(channels));
                        if (P_denial <= 0.01 && queue >= 3)
                                flag = false;
#pragma endregion

#pragma region Математическое ожидание числа занятых операторов
```

```
                    for (int i = 0; i <= channels; i++)
                    {
                        MathExpectingChannels += i * P_0 * powl(ro, i) / fact(i);
                    }
                    for (int i = 1; i <= queue; i++)
                    {
                        MathExpectingChannels += channels * P_0 * pow(ro,
channels + i) / (pow(channels, i) * fact(channels));
                    }
#pragma endregion

#pragma region Коэффициент загрузки операторов
                    loadFactorChannels = MathExpectingChannels / channels;
#pragma endregion

#pragma region Вероятность существования очереди
                    P_queue = P_0 * (powl(ro, channels) * (1 - powl(ro / channels,
queue))) / (fact(channels) * (1 - ro / channels));
#pragma endregion

#pragma region Математическое ожидания длины очереди
                    for (int i = 1; i <= queue; i++)
                    {
                        MathExpectingQueues += i * P_0 * pow(ro, channels + i)
/ (pow(channels, i) * fact(channels));
                    }
#pragma endregion

#pragma region Коэффициент занятости мест в очереди
                    loadFactorQueues = MathExpectingQueues / queue;
```

```
#pragma endregion


#pragma region Вывод
                    cout << setprecision(7) << channels << "\t";
                    cout << setprecision(7) << queue << "\t";
                    cout << setprecision(7) << P_denial << "\t";
                    cout << setprecision(7) << MathExpectingChannels << "\t";
                    cout << setprecision(7) << loadFactorChannels << "\t";
                    cout << setprecision(7) << P_queue << "\t";
                    cout << setprecision(7) << MathExpectingQueues << "\t";
                    cout << setprecision(7) << loadFactorQueues << "\n";
#pragma endregion


                    queue++;
                } while (flag);
        }
}
void Punkt3(long double maintenanceTime, long double appearanceTime, long
double maxChannels)
{
        long double P_0 = 0.0;
        long double P_n = 0.0;
        long double P_queue = 0.0;
        long double MathExpectingChannels; // Мат ожидание Каналов
        long double MathExpectingQueues; // Мат ожидание очереди
        long double loadFactorChannels; // Коэф. загрузки операторов
        long double temp;
        long double ro = maintenanceTime / appearanceTime; // Коэффициент за-
грузки СМО
```

```
        cout << "Опер\t";
        cout << "МатОжОп\t";
        cout << "КоэфОп\t";
        cout << "ВерОч\t";
        cout << "МатОжОч\n";
        for (int channels = 1; channels <= maxChannels; channels++)
        {
            P_0 = 0;
            MathExpectingChannels = 0;
            MathExpectingQueues = 0;
            P_queue = 0;
            loadFactorChannels = 0;
            temp = 0;
#pragma region P_0
            for (int i = 0; i <= channels; i++)
            {
                P_0 = P_0 + powl(ro, i) / fact(i);
            }
            P_0 = P_0 + powl(ro, (long double)channels + 1) / (fact(channels) *
(channels - ro));
            P_0 = 1 / P_0;
#pragma endregion

#pragma region P_n
            P_n = P_0;
            for (int i = 1; i <= channels; ++i) {
                P_n *= ro;
                P_n /= i;
            }
```

```
#pragma endregion

#pragma region Математическое ожидание каналов
		for (int i = 1; i <= channels; ++i) {
			MathExpectingChannels += P_0 * powl(ro, i) / fact(i);
		}
		MathExpectingChannels += channels * P_n * (ro / channels) / (1 - ro / channels);
#pragma endregion

#pragma region Коэффициент загрузки операторов
		// Коэф загрузки операторов
		loadFactorChannels = MathExpectingChannels / channels;
#pragma endregion

#pragma region Вероятность очереди
		P_queue = P_n * (ro / channels) / (1 - ro / channels);
#pragma endregion

#pragma region Мат ожидание очереди
		MathExpectingQueues = P_n * (ro / channels) * (1.0L / pow(1.0L - (ro / channels), 2));
#pragma endregion

#pragma region Вывод
		cout << setprecision(5) << channels << "\t";
		cout << setprecision(5) << MathExpectingChannels << "\t";
		cout << setprecision(5) << loadFactorChannels << "\t";
		cout << setprecision(5) << P_queue << "\t";
```

```
            cout << setprecision(5) << MathExpectingQueues << "\t";


            cout << "\n";
#pragma endregion


        }
}
void Punkt4(long double maintenanceTime, long double appearanceTime, long double waitingTime, long double maxChannels)
{
        long double P_0 = 0.0;
        long double P_n = 0.0;
        long double P_queue = 0.0;
        long double MathExpectingChannels; // Мат ожидание Каналов
        long double MathExpectingQueues; // Мат ожидание очереди
        long double loadFactorChannels; // Коэф. загрузки операторов
        long double temp;
        long double ro = maintenanceTime / appearanceTime; // Коэффициент загрузки СМО
        cout << "Опер\t";
        cout << "МатОжОп\t";
        cout << "КоэфОп\t";
        cout << "ВерОч\t";
        cout << "МатОжОч\n";
        for (int channels = 1; channels <= maxChannels; channels++)
        {
#pragma region Инициализация
            P_0 = 0;
            MathExpectingChannels = 0;
```

```
            MathExpectingQueues = 0;
            P_queue = 0;
            loadFactorChannels = 0;
            temp = 0;
#pragma endregion

#pragma region P_0
            temp = Punkt4P_0(maintenanceTime, appearanceTime, waitingTime, channels, 1);
            P_0 = Punkt4P_0(maintenanceTime, appearanceTime, waitingTime, channels, 2);
            int z = 2;
            while (abs(temp - P_0) > EPS)
            {
                z++;
                temp = P_0;
                P_0 = Punkt4P_0(maintenanceTime, appearanceTime, waiting-Time, channels, z);
            }
#pragma endregion

#pragma region P_n
            P_n = P_0 * powl(ro, channels) / fact(channels);
#pragma endregion

#pragma region Математическое ожидание каналов
            temp = Punkt4MathExpectingChannels(maintenanceTime, appear-anceTime, waitingTime, channels, 1, P_n, P_0);
            MathExpectingChannels = Punkt4MathExpectingChannels(mainte-nanceTime, appearanceTime, waitingTime, channels, 2, P_n, P_0);
```

```
            z = 2;
            while (abs(temp - MathExpectingChannels) > EPS)
            {
                z++;
                temp = MathExpectingChannels;
                MathExpectingChannels = Punkt4MathExpectingChannels(maintenanceTime, appearanceTime, waitingTime, channels, z, P_n, P_0);
            }
#pragma endregion

#pragma region Коэффициент загрузки операторов
            loadFactorChannels = MathExpectingChannels / channels;
#pragma endregion

#pragma region Вероятность очереди
            temp = Punkt4P_queue(maintenanceTime, appearanceTime, waitingTime, channels, 1, P_n, P_0);
            P_queue = Punkt4P_queue(maintenanceTime, appearanceTime, waitingTime, channels, 2, P_n, P_0);
            z = 2;
            while (abs(temp - P_queue) > EPS)
            {
                z++;
                temp = P_queue;
                P_queue = Punkt4P_queue(maintenanceTime, appearanceTime, waitingTime, channels, z, P_n, P_0);
            }
#pragma endregion

#pragma region Мат ожидание очереди
```

```
		temp = Punkt4MathExpectingQueues(maintenanceTime, appearanceTime, waitingTime, channels, 1, P_n, P_0);
		MathExpectingQueues = Punkt4MathExpectingQueues(maintenanceTime, appearanceTime, waitingTime, channels, 2, P_n, P_0);
		z = 2;
		while (abs(temp - MathExpectingQueues) > EPS)
		{
			z++;
			temp = MathExpectingQueues;
			MathExpectingQueues = Punkt4MathExpectingQueues(maintenanceTime, appearanceTime, waitingTime, channels, z, P_n, P_0);
		}
#pragma endregion

#pragma region Вывод
		cout << setprecision(5) << channels << "\t";
		cout << setprecision(5) << MathExpectingChannels << "\t";
		cout << setprecision(5) << loadFactorChannels << "\t";
		cout << setprecision(5) << P_queue << "\t";
		cout << setprecision(5) << MathExpectingQueues << "\t";

		cout << "\n";
#pragma endregion

	}
}
int main()
{
	cout.setf(ios::fixed);
	setlocale(LC_ALL, "Russian");
```

```
    long double maintenanceTime = 10; // Время обслуживания
    long double appearanceTime = 32; // Время появления заявки
    long double waitingTime = 7; // Время ожидания
    long double max = 3;
    //Punkt1(maintenanceTime, appearanceTime);
    //Punkt2(maintenanceTime, appearanceTime, max);
    //Punkt3(maintenanceTime, appearanceTime, max);
    //Punkt4(maintenanceTime, appearanceTime, waitingTime, max);
    return 0;
}
```