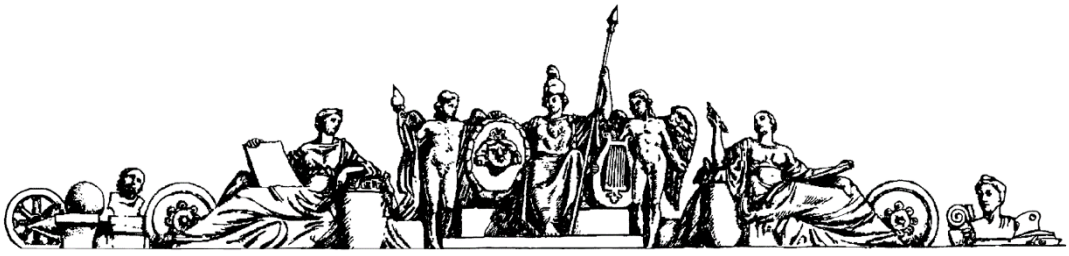


Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)
Факультет «Робототехника и комплексная автоматизация» (РК)
Кафедра «Системы автоматизированного проектирования» (РК6)

Домашнее задание №3 по «Теории вероятности».

Студент: Сергеева Диана

Группа: РК6-36Б

Преподаватель: Берчун Ю.В

Проверил:

Дата:

2020

Задание:
Известно, что среднее время между звонками клиентов составляет T c = R1+G1+B1 (R1=8, G1=7, B1=5), секунд, а среднее время обслуживания T s = R2 секунд (R2=11). 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.
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. Варьируя число место в очереди, построить семейства графиков от числа операторов: вероятности отказа; математического ожидания числа занятых операторов; коэффициента загрузки операторов; вероятности существования очереди; математического ожидания длины очереди; коэффициента занятости мест в очереди.
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.
4. Рассмотреть систему без ограничений на длину очереди, учитывающей фактор ухода клиентов из очереди (среднее приемлемое время ожидания – T w = R3+G3+B3 секунд, где R3=6, G3=9, B3=8). 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.

$T_{\text{заявки}}$ = 20
$T_{\text{обслуживания}}$ = 11
$T_{\text{ожидания}}$ = 23
Число каналов = n
Число мест в очереди = m
Частота обслуживания заявок: $\mu = \frac{1}{T_{\text{обслуживания}}}$ заявки в секунду
Частота появления новой заявки: $\lambda = \frac{1}{T_{\text{заявки}}}$ заявки в секунду
Интенсивность нагрузки системы: $\rho = \frac{\lambda}{\mu} = \frac{T_{\text{обслуживания}}}{T_{\text{заявки}}}$

1.

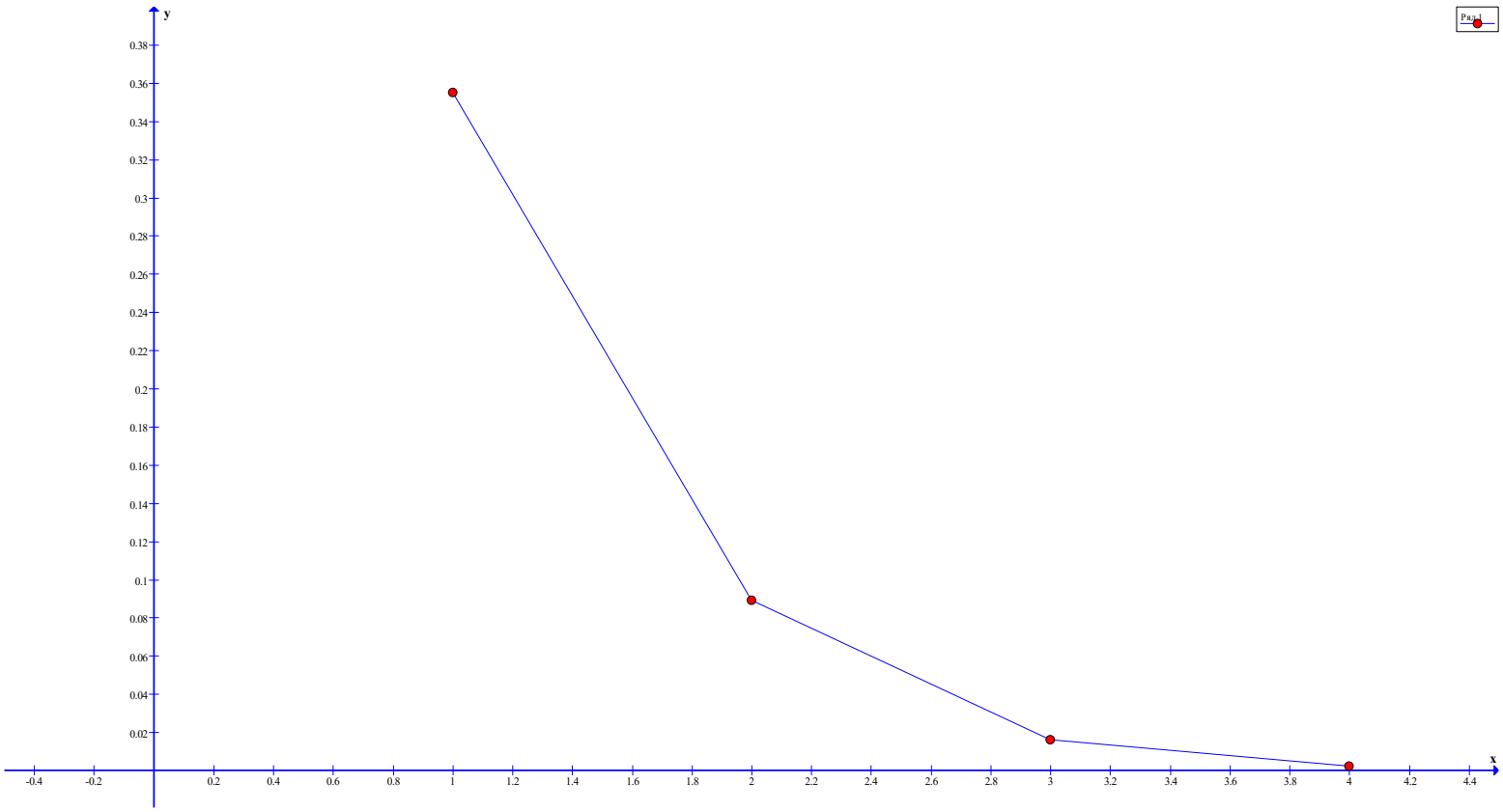
$$P_0 = \frac{1}{\sum_{i=0}^{n} \frac{\rho^i}{i!}}$$

- Вероятность отказа:

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

```
1: 0.35484
2: 0.088905
3: 0.016038
4: 0.0022004
```

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int factorial(int n)
{
    int res = 1;
    for (int i = 1; i <= n; ++i)
    {
        res *= i;
    }
    return res;
}


int main()
{
    int tService = 11, tApplication = 20;
    int operators = 1;
    double rejectionP, P0;
    double r = (double)tService / (double)tApplication;

    bool flag = true;
    do
    {
        P0 = 0;
        for (int i = 0; i <= operators; i++)
        {
            P0 += pow(r, i) / factorial(i);
        }
        P0 = 1 / P0;

        rejectionP = P0 * pow(r, operators) / factorial(operators);

        cout << setprecision(5) << operators << ": " << rejectionP << endl;

        if (rejectionP <= 0.01)
            flag = false;
        operators++;
    }
    while (flag);

    return 0;
}
```

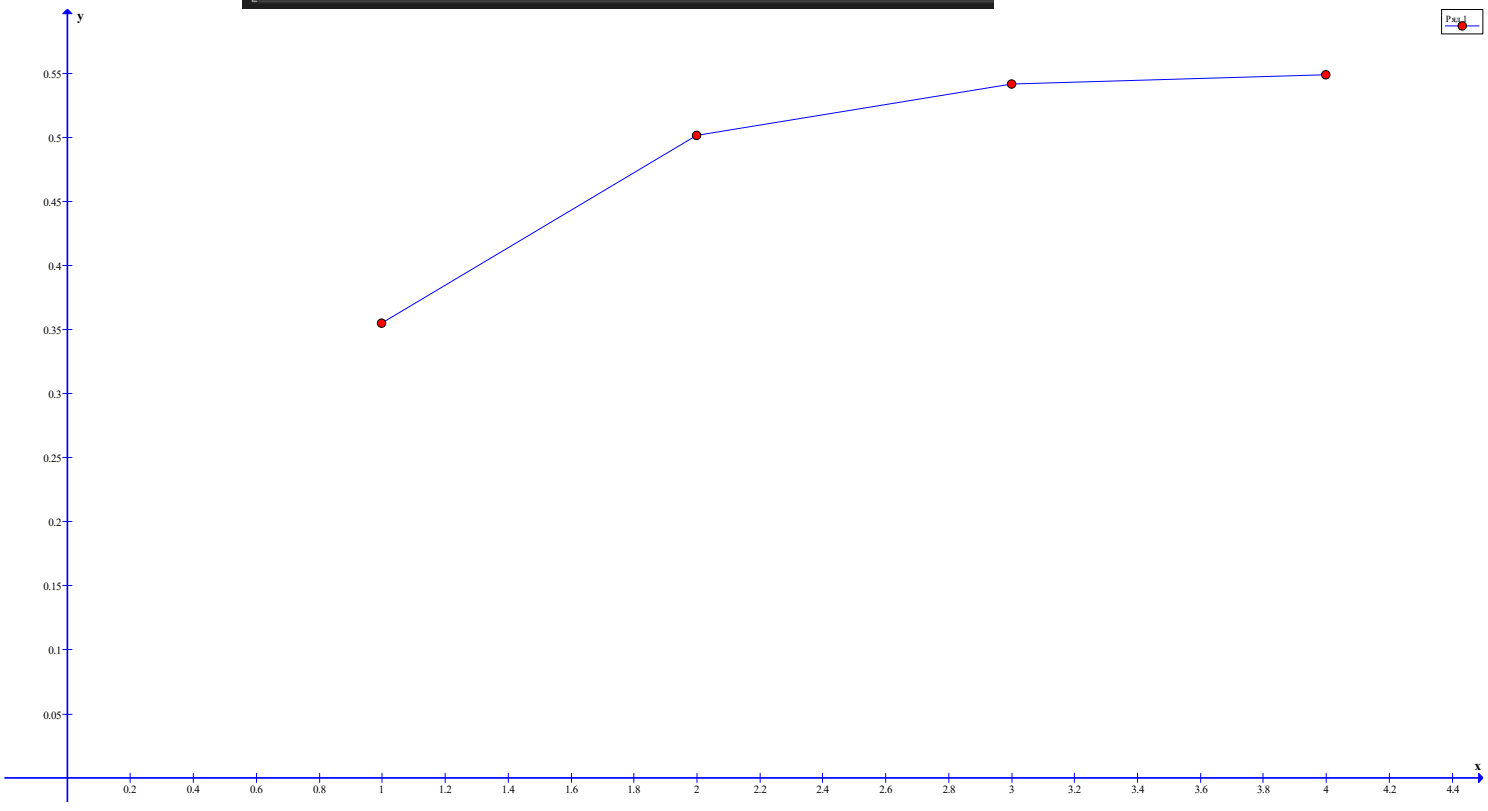


- Математическое ожидание числа занятых операторов:

$$M = \sum_{i=0}^{n} (i * P_i), \text{где } P_i = \frac{\rho^i}{i!} * P_0$$

```
1: 0.35484
2: 0.5011
3: 0.54118
4: 0.54879
```

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int factorial(int n)
{
    int res = 1;
    for (int i = 1; i <= n; ++i)
    {
        res *= i;
    }
    return res;
}


int main()
{
    int tService = 11, tApplication = 20;
    int operators = 1;
    double mathExpect, P0, rejectionP;
    double r = (double)tService / (double)tApplication;

    bool flag = true;
    do
    {
        P0 = 0;
        mathExpect = 0;
        for (int i = 0; i <= operators; i++)
        {
            P0 += pow(r, i) / factorial(i);
        }
        P0 = 1 / P0;

        rejectionP = P0 * pow(r, operators) / factorial(operators);

        for (int i = 0; i <= operators; i++)
        {
            mathExpect += i * P0 * pow(r, i) / factorial(i);
        }

        cout << setprecision(5) << operators << ": " << mathExpect << endl;

        if (rejectionP <= 0.01)
            flag = false;
        operators++;
    } while (flag);

    return 0;
}
```

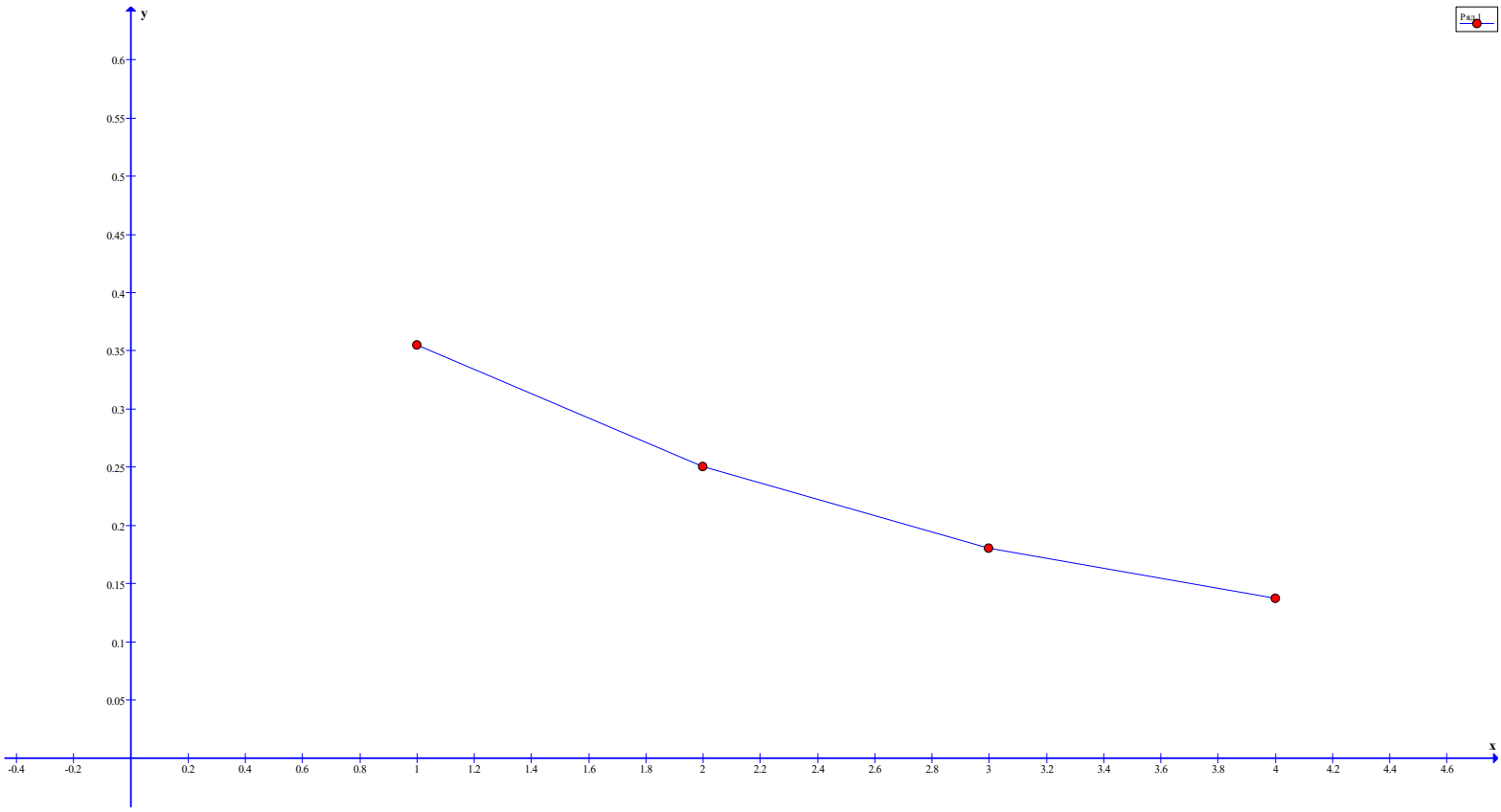


- Коэффициент загрузки операторов:

$$K_{\text{загрузки}} = \frac{M}{n}$$

```
1: 0.35484
2: 0.25055
3: 0.18039
4: 0.1372
```

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int factorial(int n)
{
    int res = 1;
    for (int i = 1; i <= n; ++i)
    {
        res *= i;
    }
    return res;
}


int main()
{
    int tService = 11, tApplication = 20;
    int operators = 1;
    double mathExpect, P0, rejectionP, opLoadFactor;
    double r = (double)tService / (double)tApplication;

    bool flag = true;
    do
    {
        P0 = 0;
        mathExpect = 0;
        for (int i = 0; i <= operators; i++)
        {
            P0 += pow(r, i) / factorial(i);
        }
        P0 = 1 / P0;

        rejectionP = P0 * pow(r, operators) / factorial(operators);

        for (int i = 0; i <= operators; i++)
        {
            mathExpect += i * P0 * pow(r, i) / factorial(i);
        }

        opLoadFactor = mathExpect / (double)operators;

        cout << setprecision(5) << operators << ": " << opLoadFactor << endl;

        if (rejectionP <= 0.01)
            flag = false;
        operators++;
    } while (flag);

    return 0;
}
```



2.

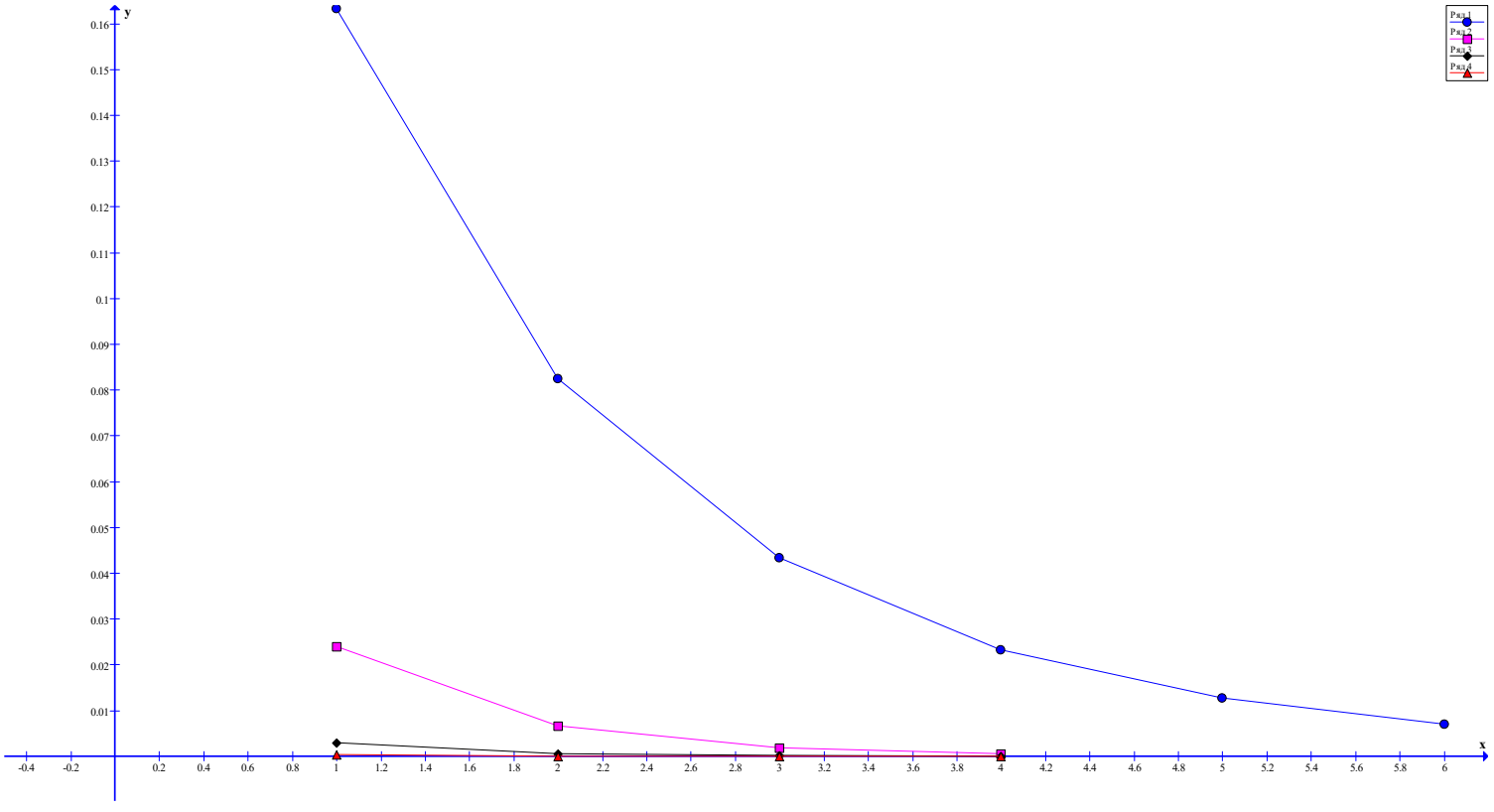
$$P_0 = \frac{1}{\sum_{i=0}^{n} \frac{\rho^i}{i!} + \frac{\rho^{n+1}}{n! * (n-\rho)} * (1 - \left(\frac{\rho}{n}\right)^m)}$$

- Вероятность отказа:

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

```
1-1: 0.16329
1-2: 0.08241
1-3: 0.04336
1-4: 0.023293
1-5: 0.012649
1-6: 0.0069088

2-1: 0.023865
2-2: 0.0065202
2-3: 0.0017898
2-4: 0.00049197

3-1: 0.0029317
3-2: 0.00053718
3-3: 9.8474e-05
3-4: 1.8053e-05

4-1: 0.00030246
4-2: 4.1586e-05
4-3: 5.7181e-06
4-4: 7.8623e-07
```

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int factorial(int n)
{
    int res = 1;
    for (int i = 1; i <= n; ++i)
    {
        res *= i;
    }
    return res;
}


int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, rejectionP;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        bool flag = true;
        int m = 1;
        do
        {
            P0 = 0;
            rejectionP = 0;

            for (int i = 0; i <= n; i++)
            {
                P0 += pow(r, i) / factorial(i);
            }
            P0 += pow(r, (double)n + 1) * (1 - pow((r / n), m)) / (factorial(n) * (n - r));
            P0 = 1 / P0;

            rejectionP = P0 * pow(r, (double)m + (double)n) / (pow(n, m) * factorial(n));
            if ((rejectionP <= 0.01) && (m >= 4))
            {
                flag = false;
            }

            cout << setprecision(5) << n << "-" << m << ": " << rejectionP << endl;

            m++;

        } while (flag);
        cout << endl;
    }

    return 0;
}
```

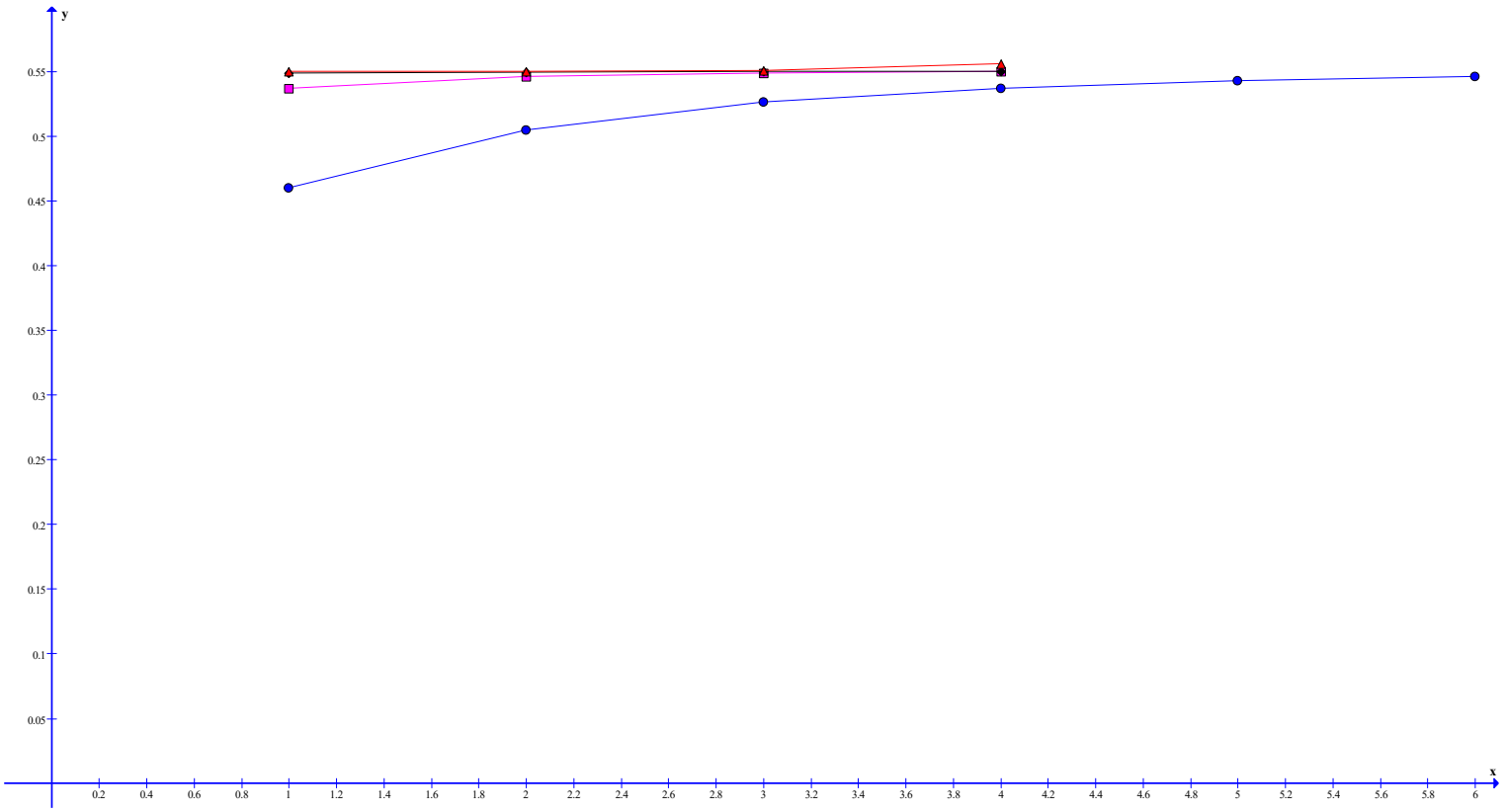


- Математическое ожидание числа занятых операторов

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

```
1-1: 0.46019
1-2: 0.50467
1-3: 0.52615
1-4: 0.53719
1-5: 0.54304
1-6: 0.5462

2-1: 0.53687
2-2: 0.54641
2-3: 0.54902
2-4: 0.54973

3-1: 0.54839
3-2: 0.5497
3-3: 0.54995
3-4: 0.54999

4-1: 0.54983
4-2: 0.54998
4-3: 0.55
4-4: 0.55
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, rejectionP, mathExpectOp;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        bool flag = true;
        int m = 1;
        do
        {
            P0 = 0;
            rejectionP = 0;
            mathExpectOp = 0;

            for (int i = 0; i <= n; i++)
            {
                P0 += pow(r, i) / factorial(i);
            }
            P0 += pow(r, (double)n + 1) * (1 - pow((r / n), m)) / (factorial(n) * (n - r));
            P0 = 1 / P0;

            rejectionP = P0 * pow(r, (double)m + (double)n) / (pow(n, m) * factorial(n));
            if ((rejectionP <= 0.01) && (m >= 4))
            {
                flag = false;
            }

            for (int i = 0; i <= n; i++)
            {
                mathExpectOp += i * P0 * pow(r, i) / factorial(i);
            }
            for (int i = 1; i <= m; i++)
            {
                mathExpectOp += n * P0 * pow(r, n + i) / (pow(n, i) * factorial(n));
            }

            cout << setprecision(5) << n << "-" << m << ": " << mathExpectOp << endl;

            m++;

        } while (flag);
        cout << endl;
    }

    return 0;
}
```

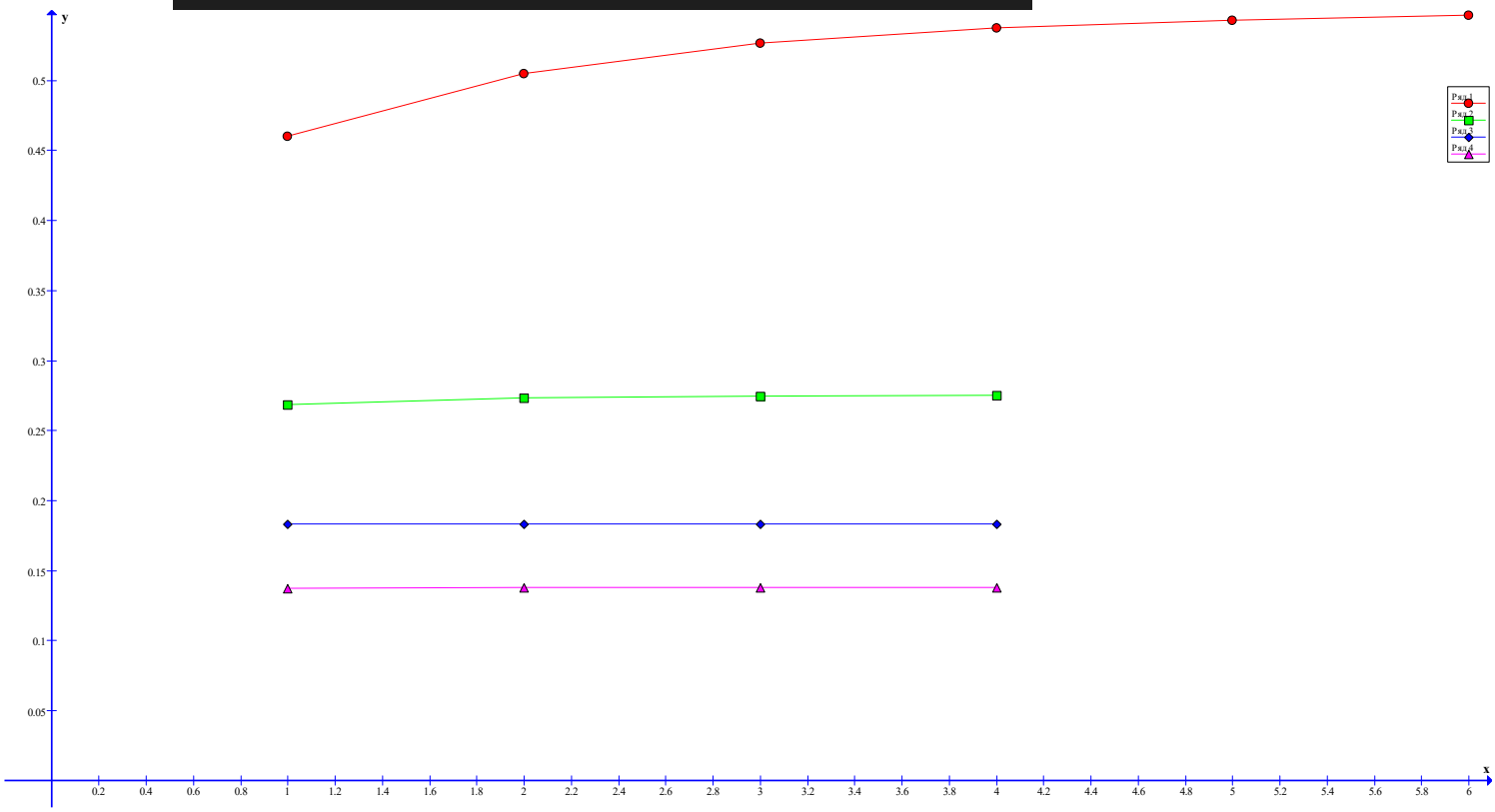


- Коэффициент загрузки операторов:

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

```
1-1: 0.46019
1-2: 0.50467
1-3: 0.52615
1-4: 0.53719
1-5: 0.54304
1-6: 0.5462

2-1: 0.26844
2-2: 0.27321
2-3: 0.27451
2-4: 0.27486

3-1: 0.1828
3-2: 0.18323
3-3: 0.18332
3-4: 0.18333

4-1: 0.13746
4-2: 0.13749
4-3: 0.1375
4-4: 0.1375
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, rejectionP, mathExpectOp, opLoadFactor;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        bool flag = true;
        int m = 1;
        do
        {
            P0 = 0;
            rejectionP = 0;
            mathExpectOp = 0;
            opLoadFactor = 0;

            for (int i = 0; i <= n; i++)
            {
                P0 += pow(r, i) / factorial(i);
            }
            P0 += pow(r, (double)n + 1) * (1 - pow((r / n), m)) / (factorial(n) * (n - r));
            P0 = 1 / P0;

            rejectionP = P0 * pow(r, (double)m + (double)n) / (pow(n, m) * factorial(n));
            if ((rejectionP <= 0.01) && (m >= 4))
            {
                flag = false;
            }

            for (int i = 0; i <= n; i++)
            {
                mathExpectOp += i * P0 * pow(r, i) / factorial(i);
            }
            for (int i = 1; i <= m; i++)
            {
                mathExpectOp += n * P0 * pow(r, n + i) / (pow(n, i) * factorial(n));
            }

            opLoadFactor = mathExpectOp / n;

            cout << setprecision(5) << n << "-" << m << ": " << opLoadFactor << endl;

            m++;

        } while (flag);
        cout << endl;
    }

    return 0;
}
```

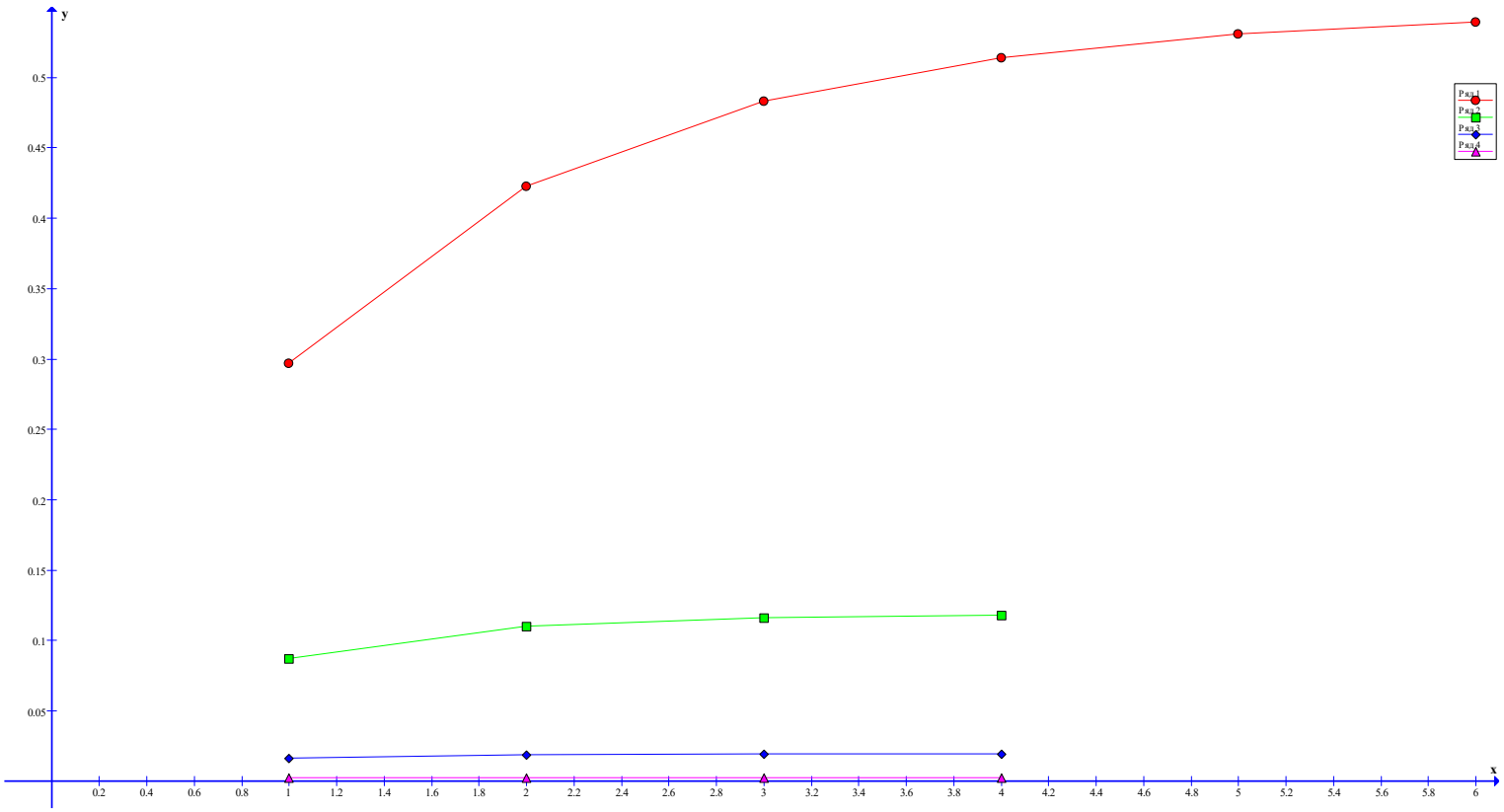


- Вероятность существования очереди

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

```
1-1: 0.2969
1-2: 0.42226
1-3: 0.48279
1-4: 0.5139
1-5: 0.53039
1-6: 0.53929

2-1: 0.086783
2-2: 0.10993
2-3: 0.11624
2-4: 0.11797

3-1: 0.015991
3-2: 0.018912
3-3: 0.019448
3-4: 0.019546

4-1: 0.0021997
4-2: 0.002502
4-3: 0.0025436
4-4: 0.0025493
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, rejectionP, Pq;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        bool flag = true;
        int m = 1;
        do
        {
            P0 = 0;
            rejectionP = 0;
            Pq = 0;

            for (int i = 0; i <= n; i++)
            {
                P0 += pow(r, i) / factorial(i);
            }
            P0 += pow(r, (double)n + 1) * (1 - pow((r / n), m)) / (factorial(n) * (n - r));
            P0 = 1 / P0;

            rejectionP = P0 * pow(r, (double)m + (double)n) / (pow(n, m) * factorial(n));
            if ((rejectionP <= 0.01) && (m >= 4))
            {
                flag = false;
            }

            Pq = P0 * pow(r, n) * (1 - pow((r / n), m)) / (factorial(n) * (1 - (r / n)));


            cout << setprecision(5) << n << "-" << m << ": " << Pq << endl;

            m++;

        } while (flag);
        cout << endl;
    }

    return 0;
}
```

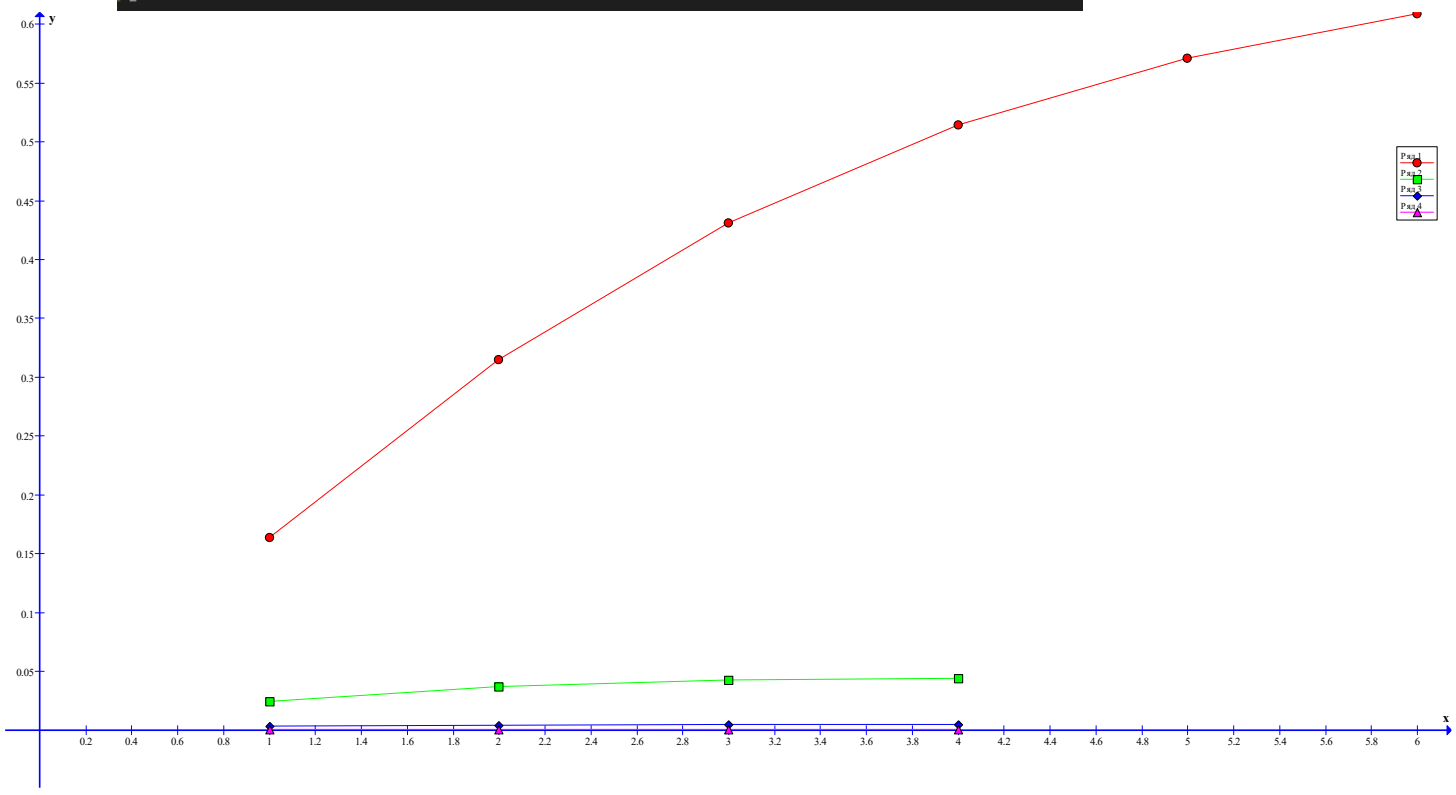


- Математическое ожидание длины очереди:

$$\text{M}_{\text{загрузки очереди}} = \sum_{i=1}^{m} (i * P_{n+i})$$

```
1-1: 0.16329
1-2: 0.31466
1-3: 0.43109
1-4: 0.51422
1-5: 0.57096
1-6: 0.60847

2-1: 0.023865
2-2: 0.03675
2-3: 0.042054
2-4: 0.044001

3-1: 0.0029317
3-2: 0.0040044
3-3: 0.0042995
3-4: 0.0043716

4-1: 0.00030246
4-2: 0.00038562
4-3: 0.00040277
4-4: 0.00040591
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, rejectionP, MathExpectingQu;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        bool flag = true;
        int m = 1;
        do
        {
            P0 = 0;
            rejectionP = 0;
            MathExpectingQu = 0;

            for (int i = 0; i <= n; i++)
            {
                P0 += pow(r, i) / factorial(i);
            }
            P0 += pow(r, (double)n + 1) * (1 - pow((r / n), m)) / (factorial(n) * (n - r));
            P0 = 1 / P0;

            rejectionP = P0 * pow(r, (double)m + (double)n) / (pow(n, m) * factorial(n));
            if ((rejectionP <= 0.01) && (m >= 4))
            {
                flag = false;
            }


            for (int i = 1; i <= m; i++)
            {
                MathExpectingQu += i * P0 * pow(r, n + i) / (pow(n, i) * factorial(n));
            }


            cout << setprecision(5) << n << "-" << m << ": " << MathExpectingQu << endl;

            m++;

        } while (flag);
        cout << endl;
    }

    return 0;
}
```

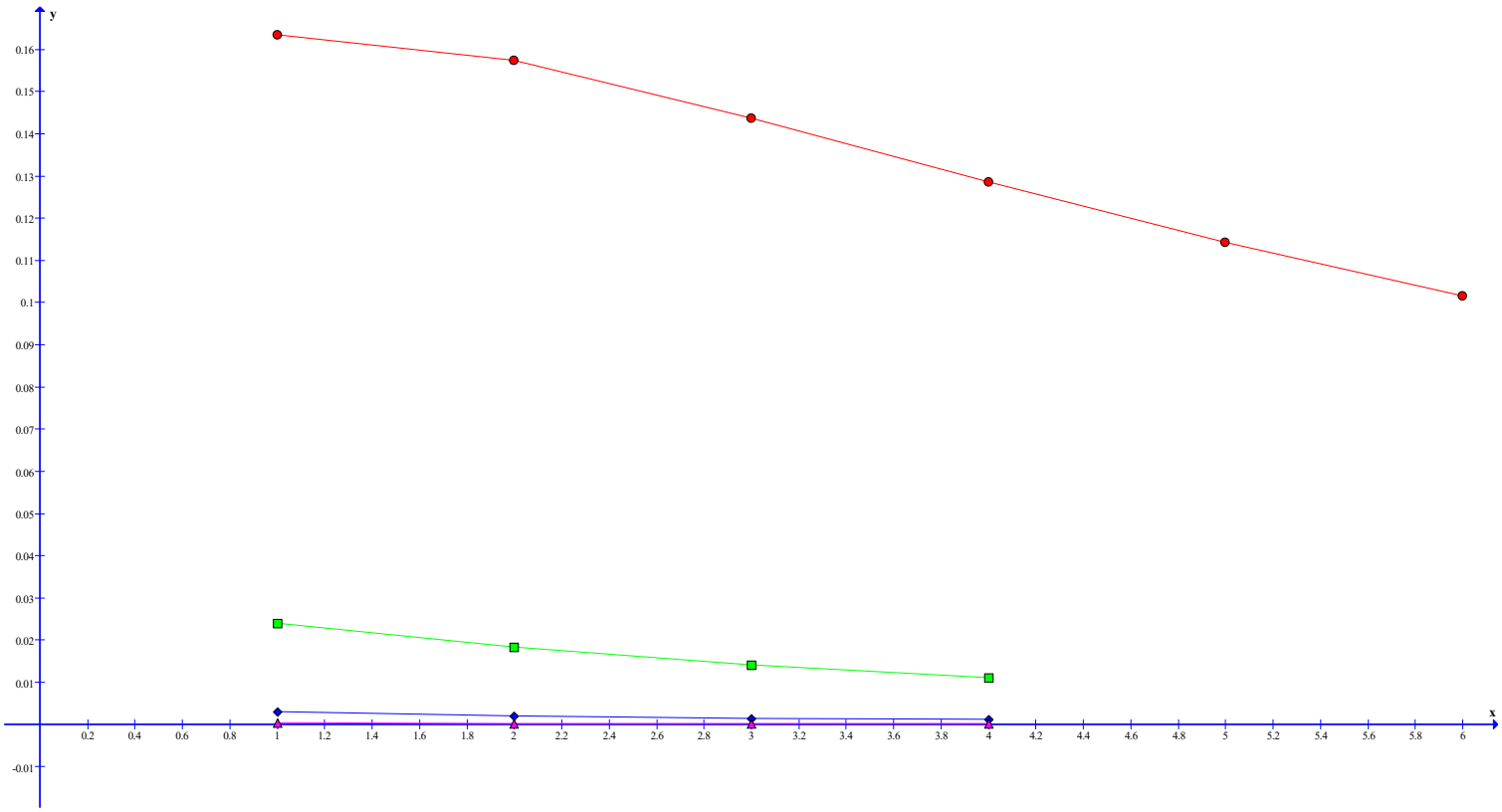


- Коэффициент занятости мест в очереди:

$$K_{\text{занятости очереди}} = \frac{\text{M}_{\text{загрузки очереди}}}{m}$$

```
1-1: 0.16329
1-2: 0.15733
1-3: 0.1437
1-4: 0.12856
1-5: 0.11419
1-6: 0.10141

2-1: 0.023865
2-2: 0.018375
2-3: 0.014018
2-4: 0.011

3-1: 0.0029317
3-2: 0.0020022
3-3: 0.0014332
3-4: 0.0010929

4-1: 0.00030246
4-2: 0.00019281
4-3: 0.00013426
4-4: 0.00010148
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, rejectionP, MathExpectingQu, mLoadFactor;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        bool flag = true;
        int m = 1;
        do
        {
            P0 = 0;
            rejectionP = 0;
            MathExpectingQu = 0;
            mLoadFactor = 0;

            for (int i = 0; i <= n; i++)
            {
                P0 += pow(r, i) / factorial(i);
            }
            P0 += pow(r, (double)n + 1) * (1 - pow((r / n), m)) / (factorial(n) * (n - r));
            P0 = 1 / P0;

            rejectionP = P0 * pow(r, (double)m + (double)n) / (pow(n, m) * factorial(n));
            if ((rejectionP <= 0.01) && (m >= 4))
            {
                flag = false;
            }


            for (int i = 1; i <= m; i++)
            {
                MathExpectingQu += i * P0 * pow(r, n + i) / (pow(n, i) * factorial(n));
            }

            mLoadFactor = MathExpectingQu / m;


            cout << setprecision(5) << n << "-" << m << ": " << mLoadFactor << endl;

            m++;

        } while (flag);
        cout << endl;
    }

    return 0;
}
```



3.

$$P_0 = \frac{1}{1 + \frac{\rho^1}{1!} + \frac{\rho^2}{2!} + \frac{\rho^3}{3!} + \cdots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{n!\,(n-\rho)}}$$

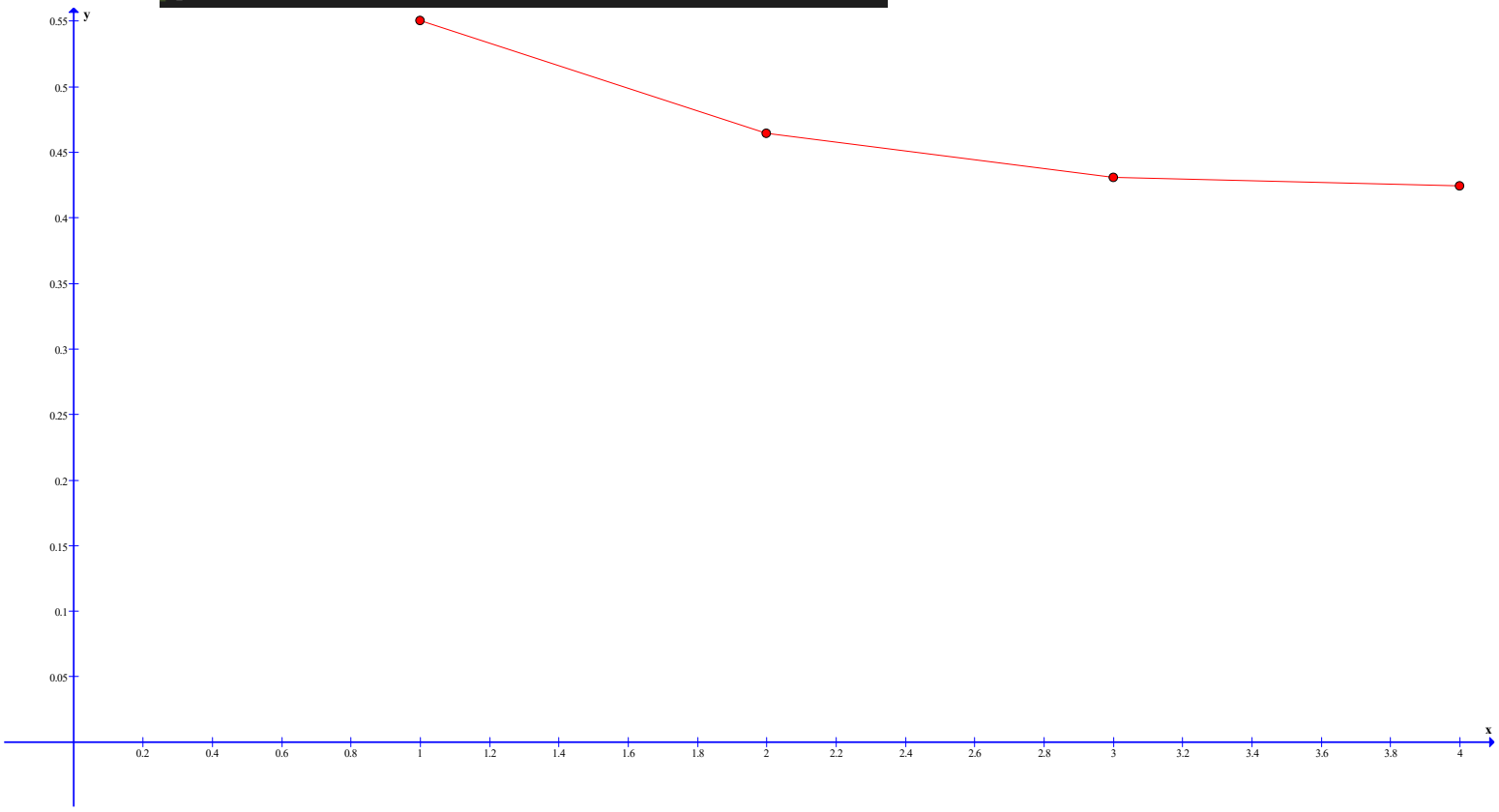
- Математическое ожидание числа занятых операторов:

$$\mathrm{M}_{\text{загрузки операторов}} = \sum_{i=0,j=0}^{n,\infty} (i * P_{i+j}) = \sum_{i=0}^{n} (i * P_i) + n * \sum_{j=1}^{\infty} P_{n+j}$$

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

```
1: 0.55
2: 0.463995
3: 0.430874
4: 0.424151
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, Pn, MathExpectOP;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        MathExpectOP = 0;

        for (int i = 0; i <= n; i++)
        {
            P0 += pow(r, i) / factorial(i);
        }
        P0 += pow(r, (double)(n + 1)) / (factorial(n) * (n - r));
        P0 = 1 / P0;

        Pn = P0;
        for (int i = 1; i <= n; ++i) {
            Pn *= r;
            Pn /= i;
        }

        for (int i = 1; i <= n; ++i) {
            MathExpectOP += P0 * pow(r, i) / factorial(i);
        }
        MathExpectOP += n * Pn * (r / n) / (1 - (r / n));

        cout << n << ": " << MathExpectOP << endl;
    }
}
```

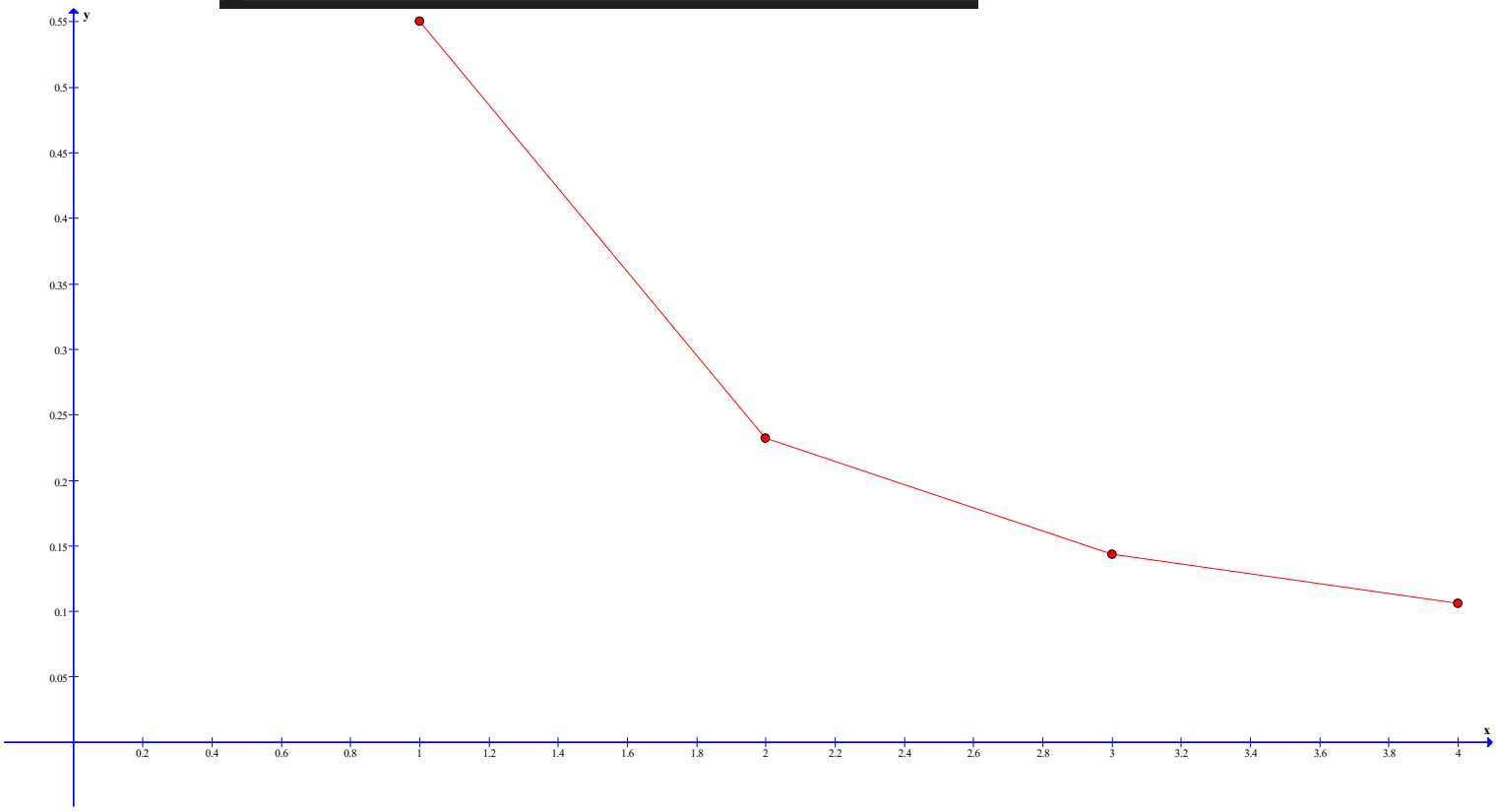


- Коэффициент загрузки операторов:

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

```
1: 0.55
2: 0.231998
3: 0.143625
4: 0.106038
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, Pn, MathExpectOP, loadFactorOP;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        MathExpectOP = 0;
        loadFactorOP = 0;

        for (int i = 0; i <= n; i++)
        {
            P0 += pow(r, i) / factorial(i);
        }
        P0 += pow(r, (double)(n + 1)) / (factorial(n) * (n - r));
        P0 = 1 / P0;

        Pn = P0;
        for (int i = 1; i <= n; ++i) {
            Pn *= r;
            Pn /= i;
        }

        for (int i = 1; i <= n; ++i) {
            MathExpectOP += P0 * pow(r, i) / factorial(i);
        }
        MathExpectOP += n * Pn * (r / n) / (1 - (r / n));

        loadFactorOP = MathExpectOP / n;

        cout << n << ": " << loadFactorOP << endl;
    }
}
```

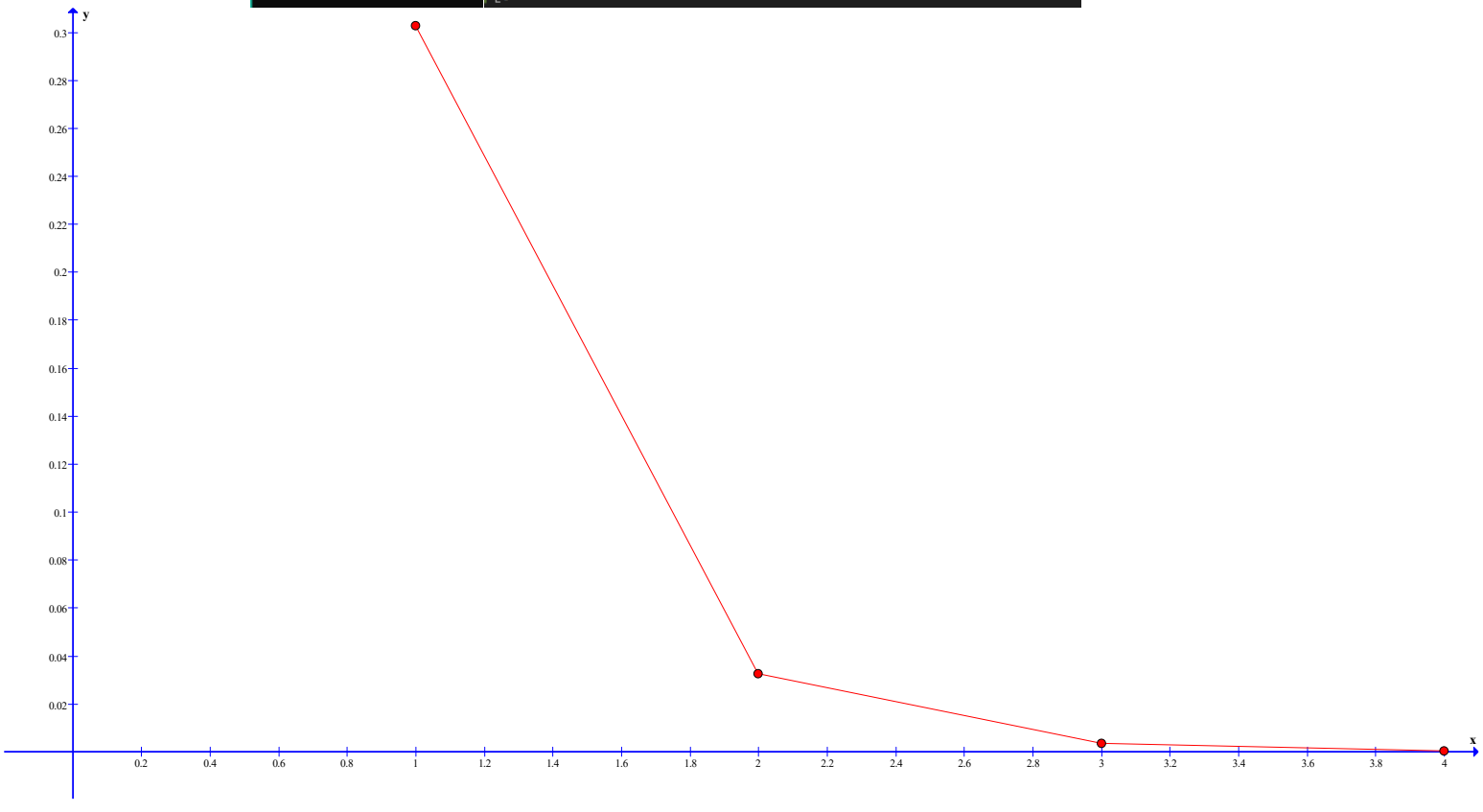


- Вероятность существования очереди:

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

```cpp
#include <iostream>
#include <iomanip>
using namespace std;


int factorial(int n)
{
    int res = 1;
    for (int i = 1; i <= n; ++i)
    {
        res *= i;
    }
    return res;
}


int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, Pn, MathExpectOP, Pq;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        Pq = 0;

        for (int i = 0; i <= n; i++)
        {
            P0 += pow(r, i) / factorial(i);
        }
        P0 += pow(r, (double)(n + 1)) / (factorial(n) * (n - r));
        P0 = 1 / P0;

        Pn = P0;
        for (int i = 1; i <= n; ++i) {
            Pn *= r;
            Pn /= i;
        }

        Pq = Pn * (r / n) / (1 - r / n);

        cout << n << ": " << Pq << endl;
    }
}
```

```
1: 0.3025
2: 0.0326225
3: 0.00358743
4: 0.000350659
```

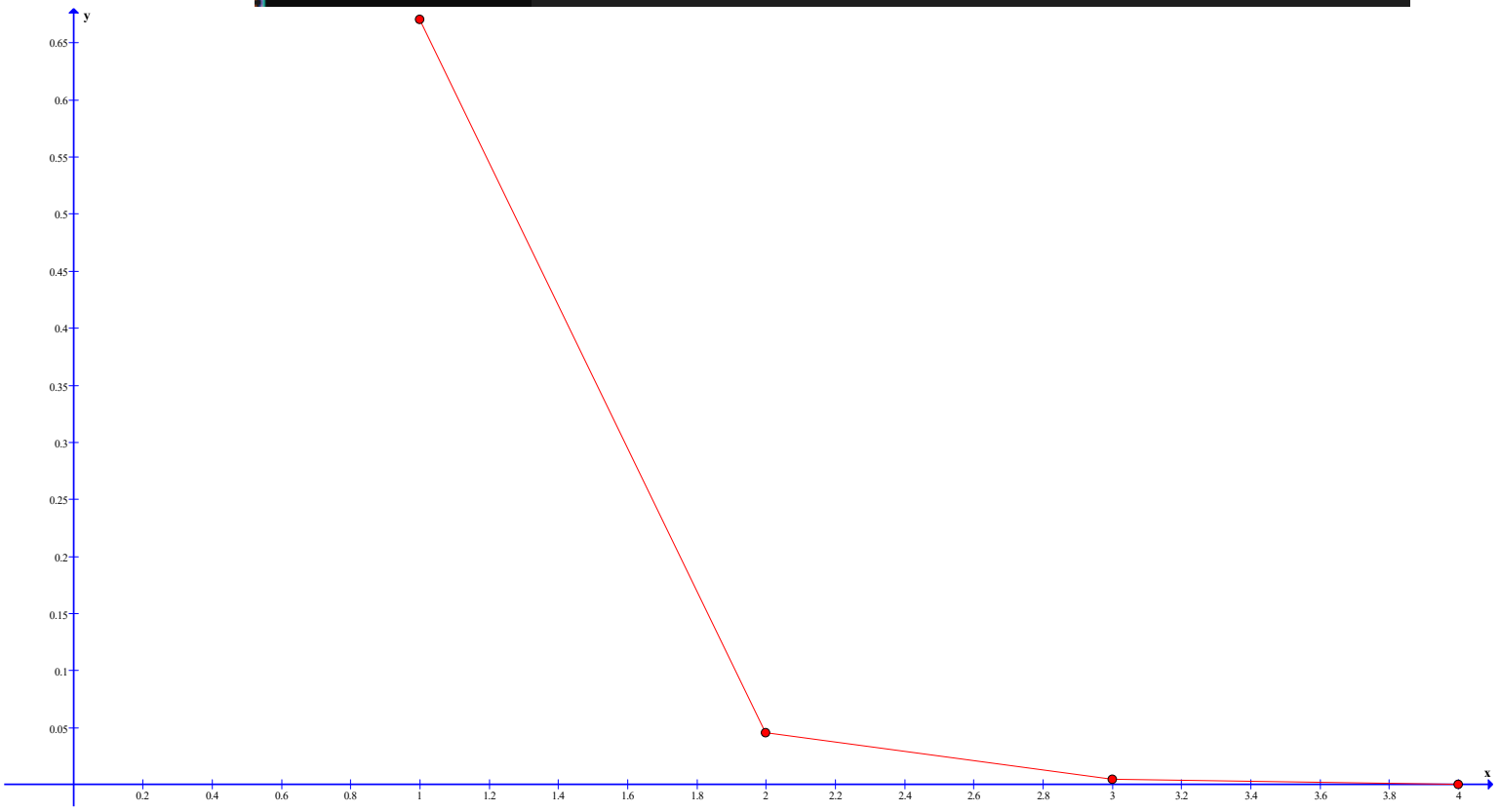


- Математическое ожидание длины очереди:

$$M_{\text{длины очереди}} = \sum_{i=1}^{\infty}(i * P_{n+i}) = P_n * \sum_{i=1}^{\infty}\left(i * \left(\frac{\rho}{n}\right)^i\right) = P_n * \frac{\rho}{n} * \frac{1}{\left(1-\frac{\rho}{n}\right)^2}$$

```
1: 0.672222
2: 0.0449966
3: 0.00439277
4: 0.000406561
```

```cpp
int main()
{
    int tService = 11, tApplication = 20;
    int operators = 4;
    double P0, Pn, MathExpectQ;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        MathExpectQ = 0;

        for (int i = 0; i <= n; i++)
        {
            P0 += pow(r, i) / factorial(i);
        }
        P0 += pow(r, (double)(n + 1)) / (factorial(n) * (n - r));
        P0 = 1 / P0;

        Pn = P0;
        for (int i = 1; i <= n; ++i) {
            Pn *= r;
            Pn /= i;
        }

        MathExpectQ = Pn * (r / n) * ((double)1 / pow(((double)1 - (r / n)), 2));

        cout << n << ": " << MathExpectQ << endl;
    }
}
```



4.

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

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

- Математическое ожидание числа занятых операторов:

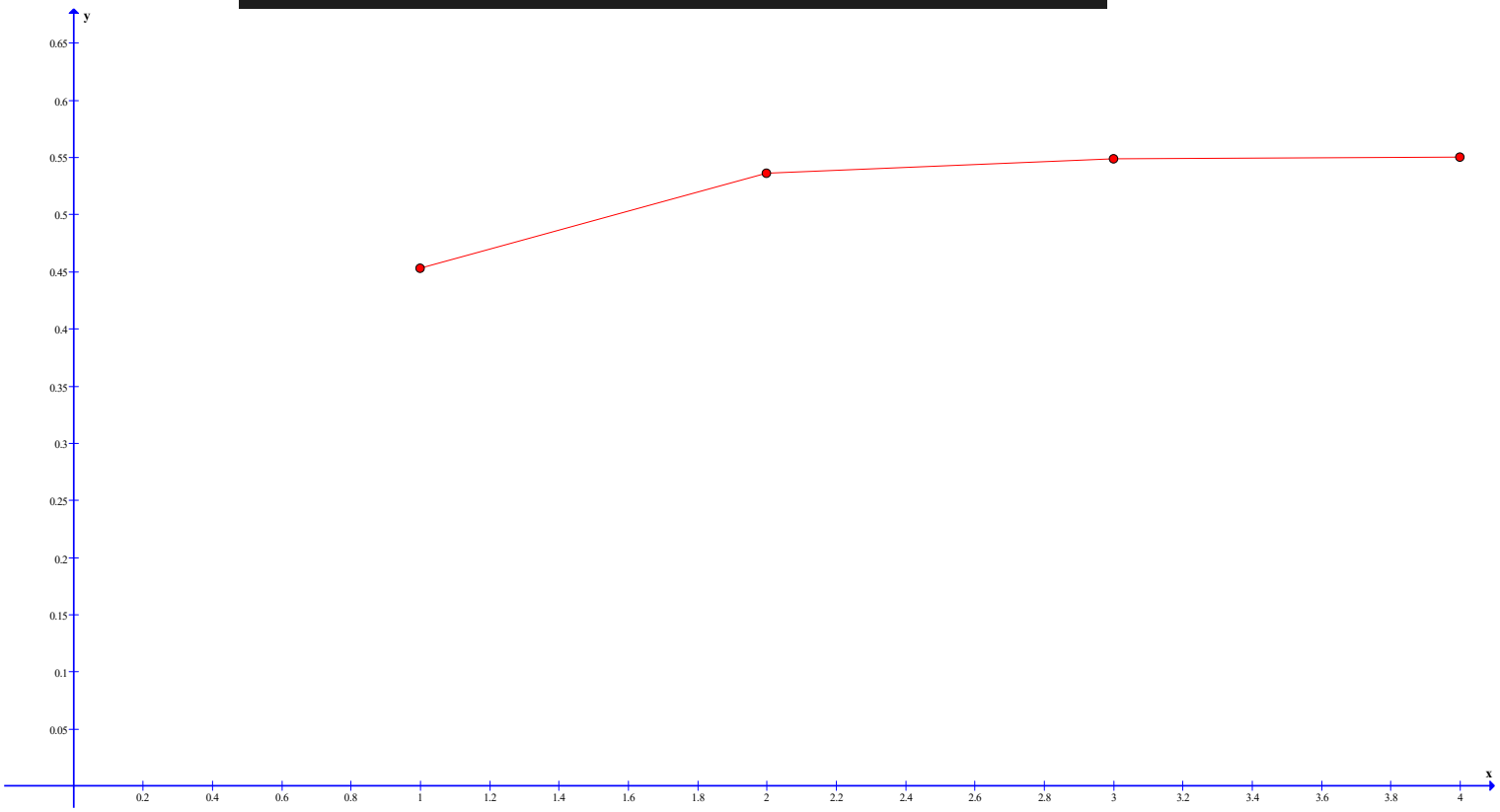
$$\text{M}_{\text{занятых операторов}} = \sum_{i=0,j=0}^{n,\infty} (i * P_i) = \sum_{i=0}^{n} (i * P_{i+j}) + n * \sum_{j=1}^{\infty} P_{n+j}$$

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

```
1: 0.452974
2: 0.536351
3: 0.548381
4: 0.549837
```

```cpp
#include <iostream>
#include <iomanip>
using namespace std;


int factorial(int n)
{
    int res = 1;
    for (int i = 1; i <= n; ++i)
    {
        res *= i;
    }
    return res;
}


double SumOfMult(double tService, double tApplication, double tWaiting, double n, int i)
{
    double result = 0;
    for (int j = 1; j <= i; j++)
    {
        double temp = 1;
        for (int k = 1; k <= j; k++)
        {
            temp *= (1 / tApplication) / (n / tService + k / tWaiting);
        }
        result += temp;
    }
    return  result;

}


double calcMathExpectingOP(double tService, double tApplication, double tWaiting, double n, int j, double Pn, double P0)
{
    double result = 0;
    for (int i = 0; i <= n; i++)
    {
        result += i * powl(tService / tApplication, i) / factorial(i);
    }
    result *= P0;
    result += n * Pn * SumOfMult(tService, tApplication, tWaiting, n, j);
    return result;
}


double calcP0(double tService, double tApplication, double tWaiting, double n, int j)
{
    double result = 0;
    for (int i = 0; i <= n; i++)
    {
        result += pow(tService / tApplication, i) / factorial(i);
    }
    result += powl(tService / tApplication, n) * SumOfMult(tService, tApplication, tWaiting, n, j) / factorial(n);
    return 1 / result;
}
```

```cpp
int main()
{
    int tService = 11, tApplication = 20, tWaiting = 23;
    int operators = 4;
    double P0, temp, Pn, MathExpectOP;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        MathExpectOP = 0;

        temp = calcP0(tService, tApplication, tWaiting, n, 1);
        P0 = calcP0(tService, tApplication, tWaiting, n, 2);
        int k = 2;
        while (abs(temp - P0) > 0e-7)
        {
            k++;
            temp = P0;
            P0 = calcP0(tService, tApplication, tWaiting, n, k);
        }

        Pn = P0 * pow(r, n) / factorial(n);

        temp = calcMathExpectingOP(tService, tApplication, tWaiting, n, 1, Pn, P0);
        MathExpectOP = calcMathExpectingOP(tService, tApplication, tWaiting, n, 2, Pn, P0);
        k = 2;
        while (abs(temp - MathExpectOP) > 0e-7)
        {
            k++;
            temp = MathExpectOP;
            MathExpectOP = calcMathExpectingOP(tService, tApplication, tWaiting, n, k, Pn, P0);
        }

        cout << n << ": " << MathExpectOP << endl;
    }

    return 0;
}
```

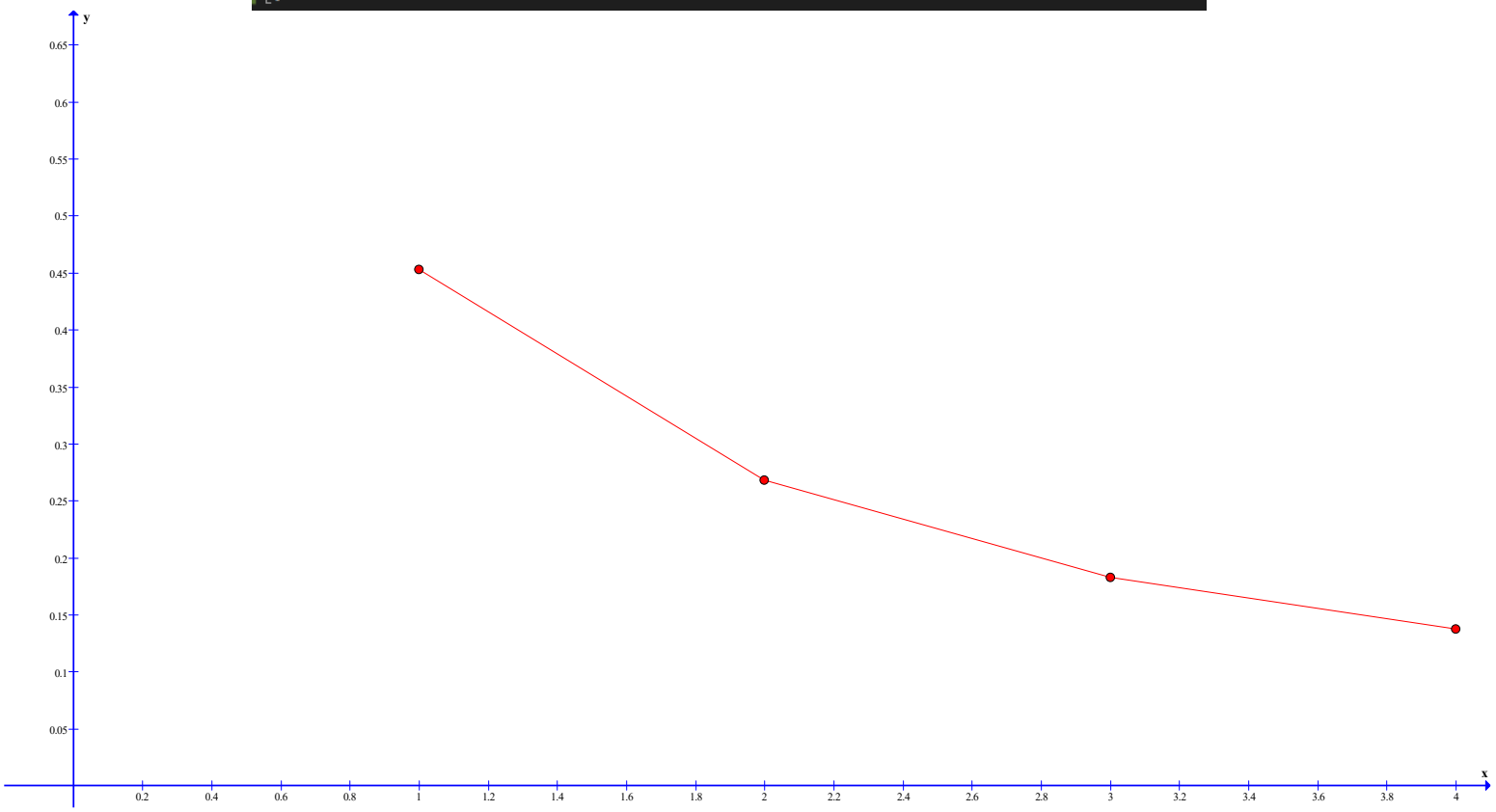


- Коэффициент загрузки операторов:

$$K = \frac{\text{M}_{\text{занятых операторов}}}{n}$$

```
1: 0.452974
2: 0.268176
3: 0.182794
4: 0.137459
```

```cpp
int main()
{
    int tService = 11, tApplication = 20, tWaiting = 23;
    int operators = 4;
    double P0, temp, Pn, MathExpectOP, loadFactorOP;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        MathExpectOP = 0;
        loadFactorOP = 0;

        temp = calcP0(tService, tApplication, tWaiting, n, 1);
        P0 = calcP0(tService, tApplication, tWaiting, n, 2);
        int k = 2;
        while (abs(temp - P0) > 0e-7)
        {
            k++;
            temp = P0;
            P0 = calcP0(tService, tApplication, tWaiting, n, k);
        }

        Pn = P0 * pow(r, n) / factorial(n);

        temp = calcMathExpectingOP(tService, tApplication, tWaiting, n, 1, Pn, P0);
        MathExpectOP = calcMathExpectingOP(tService, tApplication, tWaiting, n, 2, Pn, P0);
        k = 2;
        while (abs(temp - MathExpectOP) > 0e-7)
        {
            k++;
            temp = MathExpectOP;
            MathExpectOP = calcMathExpectingOP(tService, tApplication, tWaiting, n, k, Pn, P0);
        }

        loadFactorOP = MathExpectOP / n;

        cout << n << ": " << loadFactorOP << endl;
    }

    return 0;
}
```

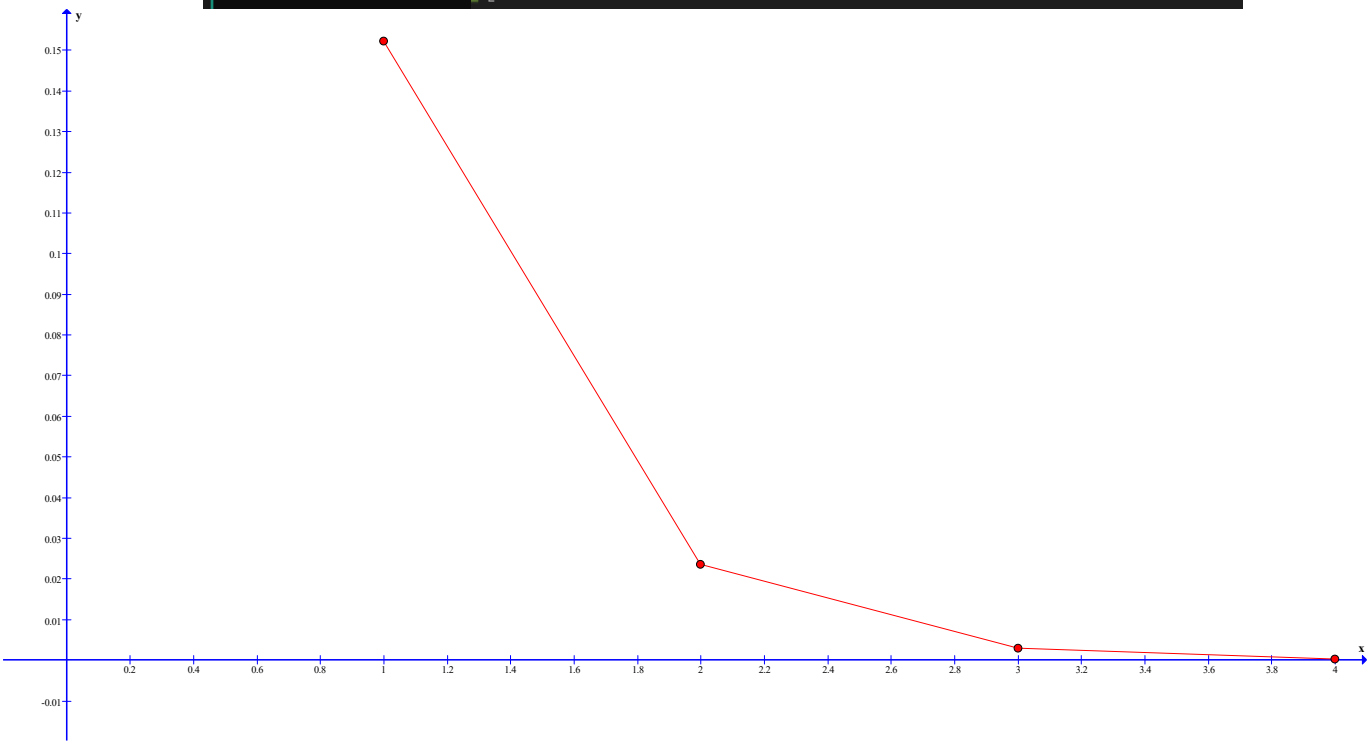


- Вероятность существования очереди:

$$\mathrm{P} = \sum_{i=1}^{\infty} P_{n+i} = P_n * \sum_{i=1}^{\infty} \prod_{k=1}^{i} \frac{\frac{1}{T_{\text{заявки}}}}{\frac{n}{T_{\text{обслуживания}}} + \frac{\mathrm{k}}{T_{\text{ожидания}}}}$$

```cpp
int main()
{
    int tService = 11, tApplication = 20, tWaiting = 23;
    int operators = 4;
    double P0, temp, Pn, Pq;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        Pq = 0;

        temp = calcP0(tService, tApplication, tWaiting, n, 1);
        P0 = calcP0(tService, tApplication, tWaiting, n, 2);
        int k = 2;
        while (abs(temp - P0) > 0e-7)
        {
            k++;
            temp = P0;
            P0 = calcP0(tService, tApplication, tWaiting, n, k);
        }

        Pn = P0 * pow(r, n) / factorial(n);

        temp = Pn * SumOfMult(tService, tApplication, tWaiting, n, 1);
        Pq = Pn * SumOfMult(tService, tApplication, tWaiting, n, 2);
        k = 2;
        while (abs(temp - Pq) > 0e-7)
        {
            k++;
            temp = Pq;
            Pq = Pn * SumOfMult(tService, tApplication, tWaiting, n, k);
        }

        cout << n << ": " << Pq << endl;
    }

    return 0;
}
```

```
1: 0.15211
2: 0.0235169
3: 0.00292908
4: 0.000303476
```



- Математическое ожидание длины очереди:

$$\mathrm{M}_{\text{длины очереди}} = \sum_{i=1}^{\infty}(i * P_{n+i}) = P_n * \sum_{i=1}^{\infty} i \prod_{k=1}^{i} \frac{\frac{1}{T_{\text{заявки}}}}{\frac{n}{T_{\text{обслуживания}}} + \frac{\mathrm{k}}{T_{\text{ожидания}}}}$$

```cpp
double SumOfMultCommon(double tService, double tApplication, double tWaiting, double n, int i)
{
    double result = 0;
    for (int j = 1; j <= i; j++)
    {
        long double temp = 1;
        for (int k = 1; k <= j; k++)
        {
            temp *= (1 / tApplication) / (n / tService + k / tWaiting);
        }
        result += j * temp;
    }
    return  result;
}


int main()
{
    int tService = 11, tApplication = 20, tWaiting = 23;
    int operators = 4;
    double P0, temp, Pn, MathExpectQ;
    double r = (double)tService / (double)tApplication;

    for (int n = 1; n <= operators; n++)
    {
        P0 = 0;
        MathExpectQ = 0;

        temp = calcP0(tService, tApplication, tWaiting, n, 1);
        P0 = calcP0(tService, tApplication, tWaiting, n, 2);
        int k = 2;
        while (abs(temp - P0) > 0e-7)
        {
            k++;
            temp = P0;
            P0 = calcP0(tService, tApplication, tWaiting, n, k);
        }

        Pn = P0 * pow(r, n) / factorial(n);

        temp = Pn * SumOfMultCommon(tService, tApplication, tWaiting, n, 1);
        MathExpectQ = Pn * SumOfMultCommon(tService, tApplication, tWaiting, n, 2);
        k = 2;
        while (abs(temp - MathExpectQ) > 0e-7)
        {
            k++;
            temp = MathExpectQ;
            MathExpectQ = Pn * SumOfMultCommon(tService, tApplication, tWaiting, n, k);
        }

        cout << n << ": " << MathExpectQ << endl;
    }

    return 0;
}
```

```
1: 0.202872
2: 0.0285378
3: 0.00338466
4: 0.000340476
```