Функция prob реализует расчет вероятностей в зависимости от поступающих в нее параметров.

In [1]:

```
import matplotlib.pyplot as plt

rho = 2 / 5
beta = 600 / 19

def prob(n, m, time):  #функция, реализующая вычисление вероятности, исходя из поступающих в нее параметров

    global prob_n      #вероятность отказа в системе без очереди
    global prob_nm     #вероятность отказа в системе с очередью
    global prob_m      #вероятность существования очереди
    global med_m       #матожидание очереди
    global med_n       #матожидание числа операторов

    list1 = [1 for i in range(n+1)]  #список вероятностей для системы без очереди
    sum1 = 1

    for k in range(n):
        list1[k+1] = rho * list1[k] / (k+1)
        sum1 += list1[k+1]  #вероятность оказаться в положении S_0 в степени -1
        prob_n = list1[n] / sum1

    if (time == 0 and m != 0): #если в системе есть очередь, но нет времени ожидания
        list2 = [(rho / n) ** (j+1) for j in range(m)]
        sum2 = 0
        med = 0
        med_n = 0

        for k in range(m):
            sum2 += list2[k]
            med += list2[k] * list1[n] * k

        c = (sum1 + list1[n] * sum2) #вероятность оказаться в положении S_0 в степени -1
        prob_nm = list1[n] * list2[m-1] / c
        prob_m = list1[n] * (1 - (rho / n) ** m) / (1 - rho / n) / c
        med_m = list1[n] * list2[0] * (1 - ((rho / n) ** m) * (1 + m * (1 - rho / n)))  / ((1 - rho / n) ** 2) / c
        med_n += (1 - prob_n) * rho + med / c

    if (time != 0 and m != 0): #если в системе есть очередь и время ожидания заявки
        list2 = [1 for j in range(m)]
        list2[0] = sum2 = rho / (n + beta)
        med = 0
        med_n = 0

        for l in range(m-1):
            list2[l+1] = rho * list2[l] / (n + (l+2) * beta)
            sum2 += list2[l+1]
            med += list2[l+1] * list1[n] * (l+1)

        c = (sum1 + list1[n] * sum2) #вероятность оказаться в положении S_0 в степени -1
        prob_nm = list1[n] * list2[m-1] / c
        prob_m = list1[n] * (1 + sum2)
        med_m = med / c
        med_n += (1 - prob_n) * rho + med / c
```

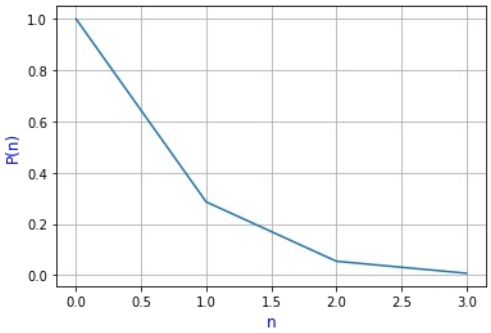
Задание 1.

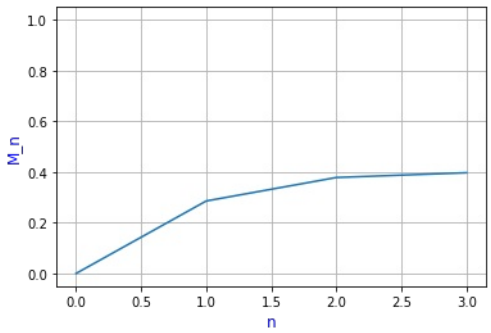
In [2]:

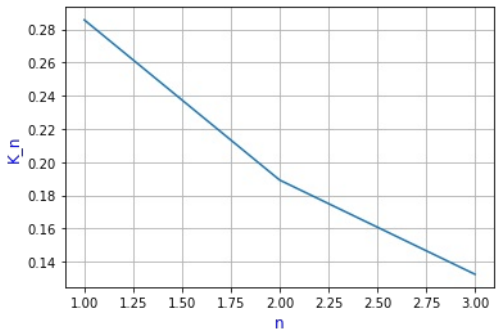
```
n = 0
P_n = []
prob_n = 1

while(prob_n > 0.01):
    prob(n, 0, 0)
    n += 1
    P_n.append(prob_n)
```

In [3]:

```
x = [i for i in range(n)]

fig = plt.figure()
plt.plot(x, P_n, 1)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('P(n)', fontsize=12, color='blue')
plt.grid(True)

M = [rho * (1 - P_n[i]) for i in range(n)]

fig = plt.figure()
plt.plot(x, M, 1)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('M_n', fontsize=12, color='blue')
plt.grid(True)

K = [M[i+1] / (i+1)  for i in range(n-1)]
x = [i+1 for i in range(n-1)]

fig = plt.figure()
plt.plot(x, K)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('K_n', fontsize=12, color='blue')
plt.grid(True)
```







Задание 2. Зависимость от числа мест в очереди.

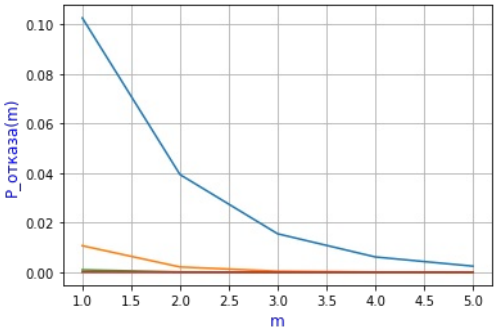
In [4]:

```
m = 5
P_nm = [[] for i in range(n)]
x_k = [[] for i in range(n)]
Med_n = [[] for i in range(n)]

for i in range(n):
    for j in range(m):
        prob(i+1, j+1, 0)
        P_nm[i].append(prob_nm)
        x_k[i].append(j+1)
        Med_n[i].append(med_n)
```

Вероятность отказа от числа мест в очереди.

In [5]:

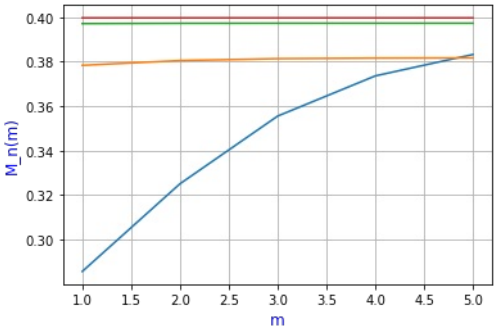
```
fig = plt.figure()
for i in range(n):
    plt.plot(x_k[i], P_nm[i])
    plt.xlabel('m', fontsize=12, color='blue')
    plt.ylabel('P_отказа(m)', fontsize=12, color='blue')
    plt.grid(True)
```



Матожидание числа занятых операторов от мест в очереди.

In [6]:

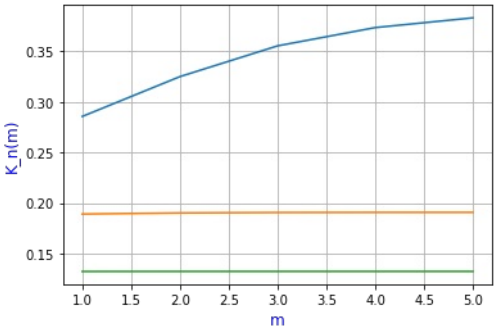
```
fig = plt.figure()
for i in range(n):
    plt.plot(x_k[i], Med_n[i])
    plt.xlabel('m', fontsize=12, color='blue')
    plt.ylabel('M_n(m)', fontsize=12, color='blue')
    plt.grid(True)
```



Коэффициент загрузги операторов от мест в очереди.

In [7]:

```
K_n = [[Med_n[i][j] / (i + 1) for j in range(m)] for i in range(n)]

fig = plt.figure()
for i in range(n-1):
    plt.plot(x_k[i], K_n[i])
    plt.xlabel('m', fontsize=12, color='blue')
    plt.ylabel('K_n(m)', fontsize=12, color='blue')
    plt.grid(True)
```



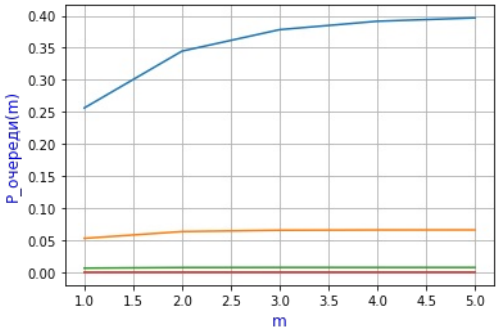
In [8]:

```
P_m = [[] for i in range(n)]
x_k = [[] for i in range(n)]
Med_m = [[] for i in range(n)]

for i in range(n):
    for j in range(m):
        prob(i+1, j+1, 0)
        P_m[i].append(prob_m)
        x_k[i].append(j+1)
        Med_m[i].append(med_m)
```

Вероятность образования очереди от мест в очереди.

In [9]:

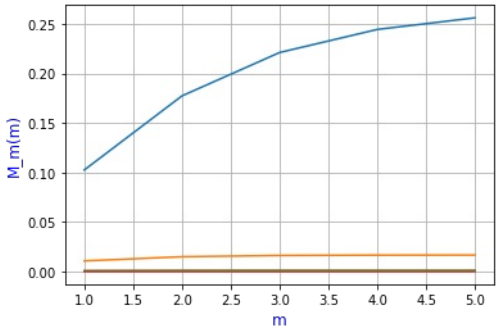
```
fig = plt.figure()
for i in range(n):
    plt.plot(x_k[i], P_m[i])
    plt.xlabel('m', fontsize=12, color='blue')
    plt.ylabel('P_очереди(m)', fontsize=12, color='blue')
    plt.grid(True)
```



Математическое ожидание длины очереди от мест в очереди.

In [10]:

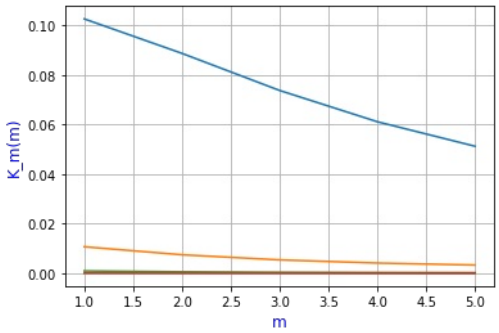
```
fig = plt.figure()
for i in range(n):
    plt.plot(x_k[i], Med_m[i])
    plt.xlabel('m', fontsize=12, color='blue')
    plt.ylabel('M_m(m)', fontsize=12, color='blue')
    plt.grid(True)
```



Коэффициент занятости мест в очереди от длины очереди.

In [11]:

```
x = [x_k[i][-1] for i in range(n)]
K_m = [[Med_m[i][j] / (j + 1) for j in range(x[i])] for i in range(n)]

fig = plt.figure()
for i in range(n):
    plt.plot(x_k[i], K_m[i])
    plt.xlabel('m', fontsize=12, color='blue')
    plt.ylabel('K_m(m)', fontsize=12, color='blue')
    plt.grid(True)
```



Задание 2. Зависимость от числа операторов.

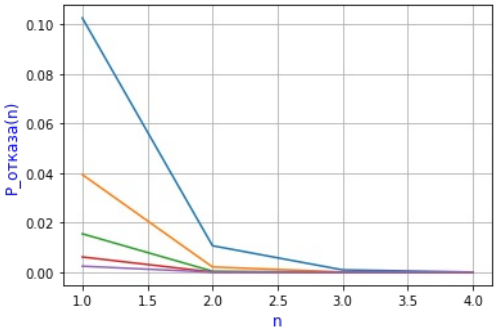
In [12]:

```
P_nm = [[] for i in range(m)]
x_k = [[] for i in range(m)]
Med_n = [[] for i in range(m)]

for i in range(m):
    for j in range(n):
        prob(j+1, i+1, 0)
        P_nm[i].append(prob_nm)
        x_k[i].append(j+1)
        Med_n[i].append(med_n)
```

Вероятность отказа от числа операторов.

In [13]:

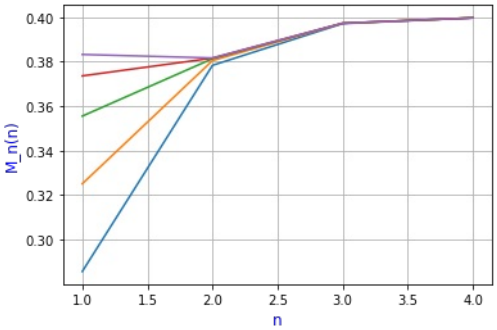
```
fig = plt.figure()
for i in range(m):
    plt.plot(x_k[i], P_nm[i])
    plt.xlabel('n', fontsize=12, color='blue')
    plt.ylabel('P_отказа(n)', fontsize=12, color='blue')
    plt.grid(True)
```



Математическое ожидания числа занятых операторов от числа занятых операторов.

In [14]:

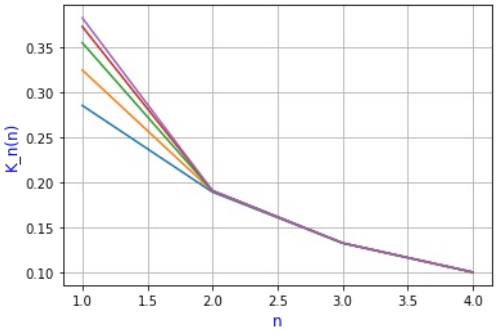
```
fig = plt.figure()
for i in range(m):
    plt.plot(x_k[i], Med_n[i])
    plt.xlabel('n', fontsize=12, color='blue')
    plt.ylabel('M_n(n)', fontsize=12, color='blue')
    plt.grid(True)
```



Коэффициент загрузки от числа операторов.

In [15]:

```
x = [x_k[i][-1] for i in range(m)]
K_n = [[Med_n[i][j] / (j + 1) for j in range(x[i])] for i in range(m)]

fig = plt.figure()
for i in range(m):
    plt.plot(x_k[i], K_n[i])
    plt.xlabel('n', fontsize=12, color='blue')
    plt.ylabel('K_n(n)', fontsize=12, color='blue')
    plt.grid(True)
```



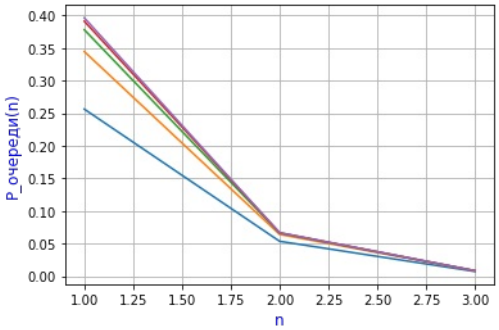
In [16]:

```
n = 3
m = 5
P_m = [[] for i in range(m)]
x_k = [[] for i in range(m)]
Med_m = [[] for i in range(m)]

for i in range(m):
    for j in range(n):
        prob_m_ = prob_m
        prob(j+1, i+1, 0)
        P_m[i].append(prob_m)
        x_k[i].append(j+1)
        Med_m[i].append(med_m)
```

Вероятность образования очереди от числа операторов.

In [17]:

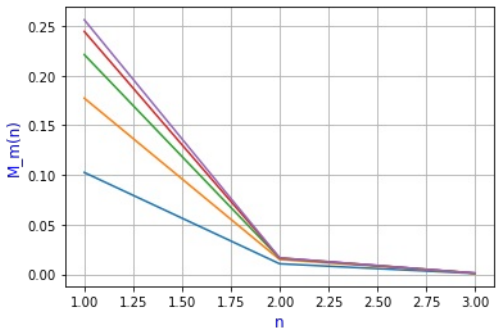
```
fig = plt.figure()
for i in range(m):
    plt.plot(x_k[i], P_m[i])
    plt.xlabel('n', fontsize=12, color='blue')
    plt.ylabel('P_очереди(n)', fontsize=12, color='blue')
    plt.grid(True)
```



Матожидание длины очереди от числа операторов.

In [18]:

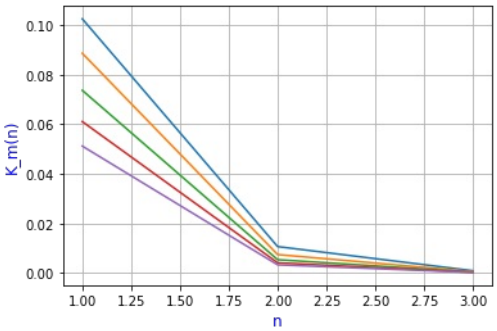
```
fig = plt.figure()
for i in range(m):

    plt.plot(x_k[i], Med_m[i])
    plt.xlabel('n', fontsize=12, color='blue')
    plt.ylabel('M_m(n)', fontsize=12, color='blue')
    plt.grid(True)
```



Коэффициент загруженности очереди от числа операторов.

In [19]:

```
K_m = [[Med_m[i][j] / (i + 1) for j in range(n)] for i in range(m)]

fig = plt.figure()
for i in range(m):
    plt.plot(x_k[i], K_m[i])
    plt.xlabel('n', fontsize=12, color='blue')
    plt.ylabel('K_m(n)', fontsize=12, color='blue')
    plt.grid(True)
```



Задание 3.

Математическое ожидание числа операторов.

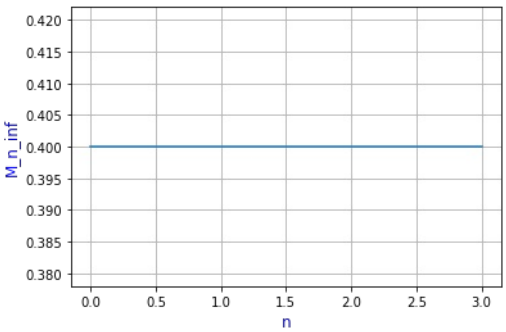
In [20]:

```
n = 1
M_inf = [rho]
prob_n = 1
P_inf = []

while(prob_n > 0.01):
    prob(n, m, 0)
    n += 1
    M_inf.append(rho)
    P_inf.append(prob_n)
```

Матожидание числа операторов.

In [21]:

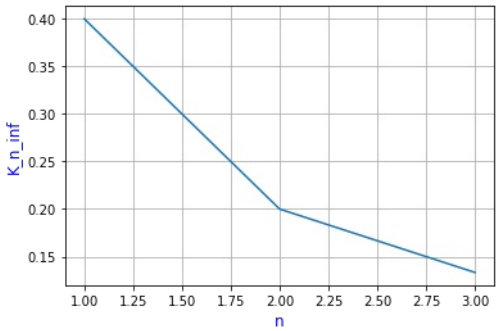
```
x = [i for i in range(n)]

fig = plt.figure()
plt.plot(x, M_inf)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('M_n_inf', fontsize=12, color='blue')
plt.grid(True)
```



Коэффициент загрузки операторов.

In [22]:

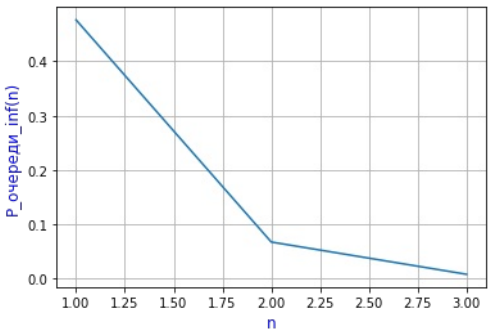
```
K_inf = [rho / (i + 1) for i in range(n-1)]
x = [i+1 for i in range(n-1)]

fig = plt.figure()
plt.plot(x, K_inf)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('K_n_inf', fontsize=12, color='blue')
plt.grid(True)
```



Вероятность существования очереди.

In [23]:

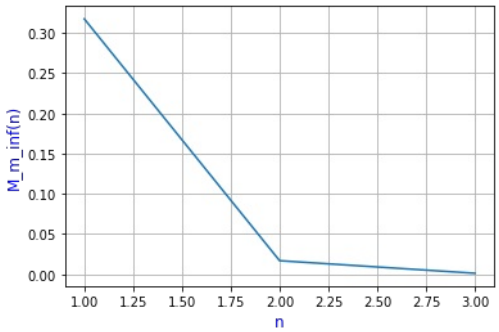
```
P_inf_m = [P_inf[i] * (i+1) / ((i+1) - rho) for i in range(n-1)]

fig = plt.figure()
plt.plot(x, P_inf_m)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('P_очереди_inf(n)', fontsize=12, color='blue')
plt.grid(True)
```



Матожидание длины очереди.

In [24]:

```
M_inf_m = [P_inf[i] * (i+1) * rho / ((i+1) - rho)**2 for i in range(n-1)]

fig = plt.figure()
plt.plot(x, M_inf_m)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('M_m_inf(n)', fontsize=12, color='blue')
plt.grid(True)
```



Задание 4.

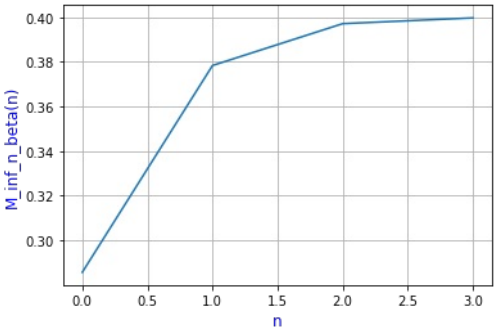
In [25]:

```
M_inf = []
P_inf = []
Med_m_inf = []
for i in range(n):
    prob(i+1, 100, beta)
    M_inf.append(med_n)
    Med_m_inf.append(med_m)
    P_inf.append(prob_m)
```

Матожидание числа операторов.

In [26]:

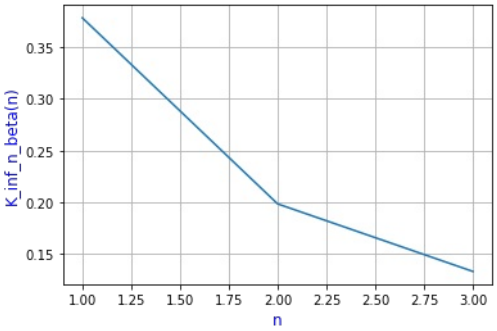
```
x = [i for i in range(n)]

fig = plt.figure()
plt.plot(x, M_inf)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('M_inf_n_beta(n)', fontsize=12, color='blue')
plt.grid(True)
```



Коэффициент загрузки операторов.

In [27]:

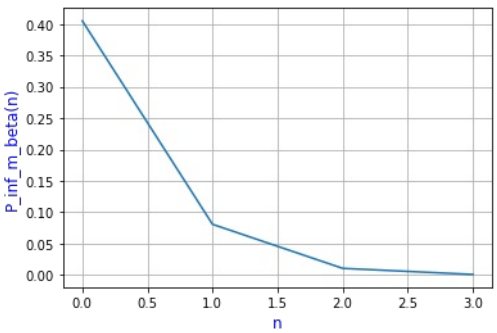
```
K_inf = [M_inf[i+1] / (i + 1) for i in range(n-1)]
x = [i+1 for i in range(n-1)]

fig = plt.figure()
plt.plot(x, K_inf)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('K_inf_n_beta(n)', fontsize=12, color='blue')
plt.grid(True)
```



Вероятность существования очереди.

In [28]:

```
x = [i for i in range(n)]

fig = plt.figure()
plt.plot(x, P_inf)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('P_inf_m_beta(n)', fontsize=12, color='blue')
plt.grid(True)
```



Матожидание очереди.

In [29]:

```
fig = plt.figure()
plt.plot(x, Med_m_inf)
plt.xlabel('n', fontsize=12, color='blue')
plt.ylabel('M_m_inf_beta(n)', fontsize=12, color='blue')
plt.grid(True)
```