In [2]:
```
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
import pylab
%matplotlib inline
```

Вывод формулы $E(N_t|N_s)$

Воспользуемся линейностью условного матожидания:

$$E(N_t|N_s) = E([N_t - N_s]|N_s) + E(N_s|N_s) = E([N_t - N_s]|N_s) + N_s$$

Известно что
$(N_t - N_s) \sim Pois(\lambda(t-s))$
$(N_t - N_s)$ независимо с N_s

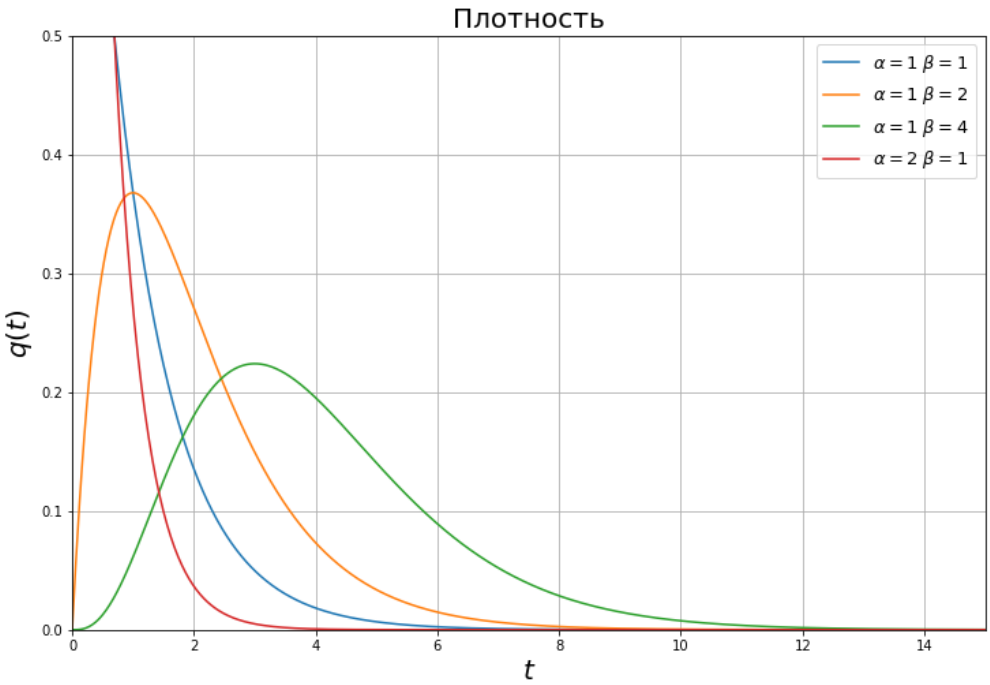
Значит, $E(N_t|N_s) = E(N_t - N_s) + N_s = \lambda(t-s) + N_s$

In [3]:
```
# данные из файла
real_lambda_ = 1/95
t_0 = 500
t = 100000

data = np.loadtxt('6.csv.xls', delimiter=',', skiprows=3)
```

Распределение, сопряженное к экспоненциальному - $\Gamma(\alpha, \beta)$. Тогда байесовская оценка параметра λ равна $\frac{n+\alpha}{\beta+\sum_{i=1}^{n}\xi_i}$,
где ξ_i - время между i-ым и (i + 1)-ым выходами из строя сервера.

Подберём параметры сопряженного распределения, для этого рассмотрим график плотности гамма распределения:

In [4]:
```
alpha_ = np.array([1, 1, 1, 2])
beta_ = np.array([1, 2, 4, 1])

x = np.linspace(0, 20, 1000)
plt.figure(figsize=(12, 8))
for i in range(4):
    plt.plot(x, sps.gamma.pdf(x, a=beta_[i], scale=1/alpha_
[i]), linewidth = 1.5,
             label = r'$\alpha = $' + str(alpha_[i]) + r' $
\beta = $' + str(beta_[i]))
plt.xlim(0, 15)
plt.ylim(0, 0.5)
plt.xlabel('$t$', fontsize=20)
plt.ylabel('$q(t)$', fontsize=20)
plt.title('Плотность', fontsize=20)
plt.legend(fontsize=13)
plt.grid()
plt.show()
```



Поскольку в начальный момент времени мы ничего не знаем о параметре λ, то нам не подходят параметры Гамма распределения, при которых образуется явный "горб" над каким-то конкретным числом. Нам следует этого избегать.

Значит, нужно брать параметр $\alpha = 1$, а за параметр β можно взять 2, при котором график плотности выглядит не очень крутым.

In [5]:
```
alpha = 1
beta = 2

# функция вычисляет действительное количество вышедших из с
троя серверов
# к моменту времени t, по данным из файла
def count_real_num(t):
    return np.sum(data <= t)

reality = []      # реальное число вышедших из строя сервер
ов
prediction = []   # предсказанное число вышедших из строя с
ерверов (по известному lambda)
time = np.arange(0, t+t_0, t_0)
for s in time:
    reality.append(count_real_num(s))
    prediction.append(count_real_num(s) + real_lambda_*(t-s
))
```

Предскажем количество серверов, оценив λ:

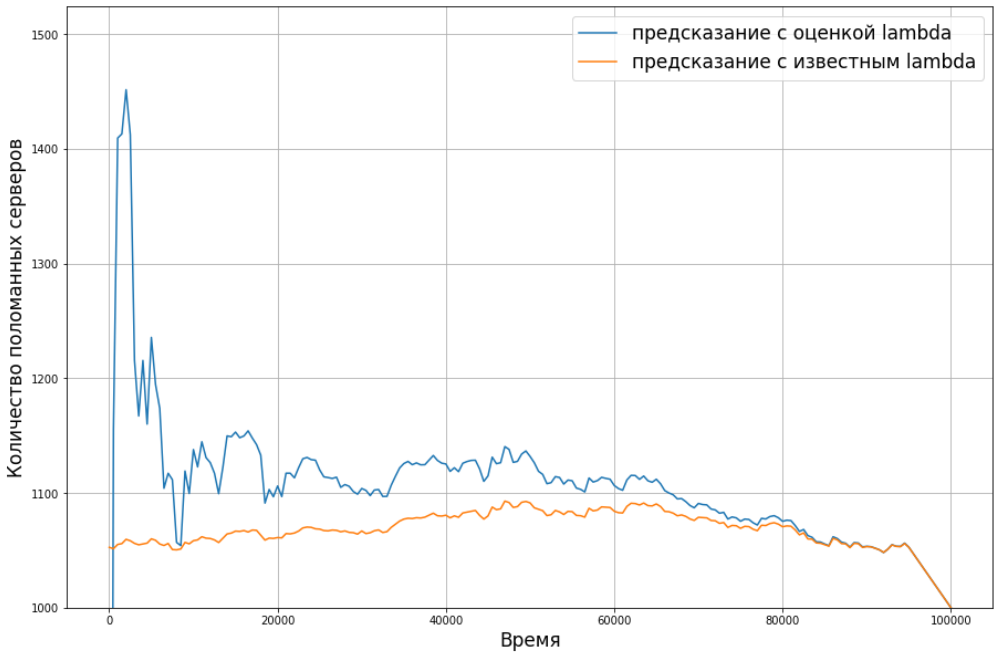
In [6]:
```
xi = np.zeros(data.size)
xi[1:] = data[:data.size - 1]
xi = data - xi

lambda_ = np.zeros(t//t_0 + 1)
for i in range(t//t_0 + 1):
    lambda_[i] = (reality[i] + alpha) / (xi.cumsum()[realit
y[i] - 1] + beta)

result = lambda_ * (t - time) + reality
```

Вывод предсказания того, сколько серверов нужно докупить к моменту времени t:

In [7]:
```
for i in range(time.size):
    print ("Время = %d : количество серверов = %d" % (time[
i], result[i]))
```

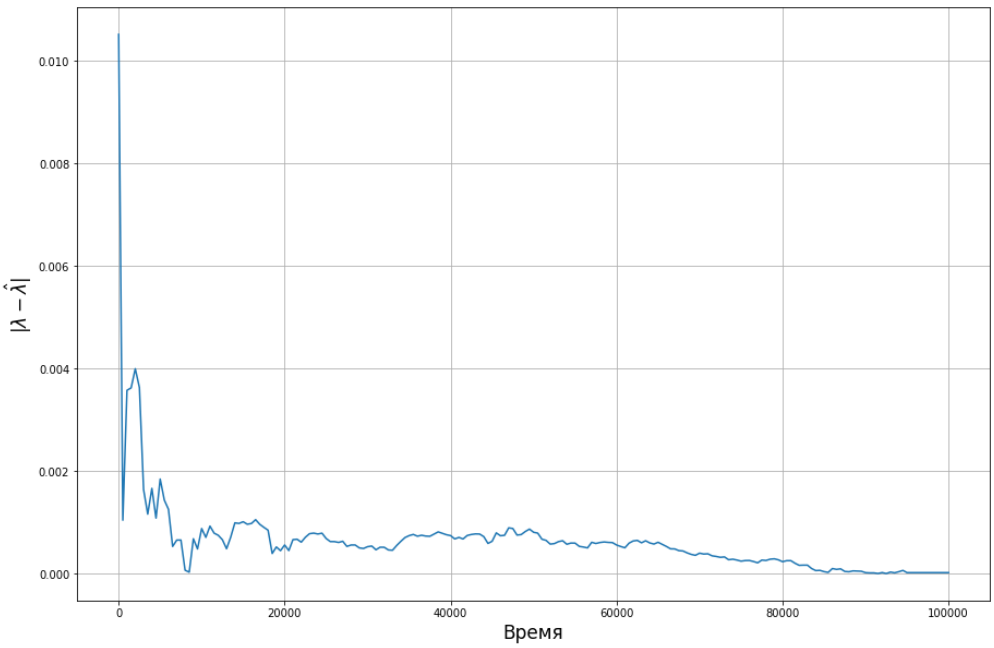
```
Время = 0 : количество серверов = 1
Время = 500 : количество серверов = 1155
Время = 1000 : количество серверов = 1409
Время = 1500 : количество серверов = 1412
Время = 2000 : количество серверов = 1451
Время = 2500 : количество серверов = 1412
Время = 3000 : количество серверов = 1215
Время = 3500 : количество серверов = 1167
Время = 4000 : количество серверов = 1215
Время = 4500 : количество серверов = 1159
Время = 5000 : количество серверов = 1235
Время = 5500 : количество серверов = 1194
Время = 6000 : количество серверов = 1173
Время = 6500 : количество серверов = 1104
Время = 7000 : количество серверов = 1117
Время = 7500 : количество серверов = 1111
Время = 8000 : количество серверов = 1056
Время = 8500 : количество серверов = 1054
Время = 9000 : количество серверов = 1119
Время = 9500 : количество серверов = 1099
Время = 10000 : количество серверов = 1137
Время = 10500 : количество серверов = 1122
Время = 11000 : количество серверов = 1144
Время = 11500 : количество серверов = 1130
Время = 12000 : количество серверов = 1126
Время = 12500 : количество серверов = 1117
Время = 13000 : количество серверов = 1099
Время = 13500 : количество серверов = 1122
Время = 14000 : количество серверов = 1149
Время = 14500 : количество серверов = 1148
Время = 15000 : количество серверов = 1153
Время = 15500 : количество серверов = 1148
Время = 16000 : количество серверов = 1149
Время = 16500 : количество серверов = 1154
Время = 17000 : количество серверов = 1147
Время = 17500 : количество серверов = 1142
Время = 18000 : количество серверов = 1132
Время = 18500 : количество серверов = 1091
Время = 19000 : количество серверов = 1103
Время = 19500 : количество серверов = 1096
Время = 20000 : количество серверов = 1106
Время = 20500 : количество серверов = 1096
Время = 21000 : количество серверов = 1117
Время = 21500 : количество серверов = 1117
Время = 22000 : количество серверов = 1113
Время = 22500 : количество серверов = 1122
Время = 23000 : количество серверов = 1129
Время = 23500 : количество серверов = 1130
Время = 24000 : количество серверов = 1129
Время = 24500 : количество серверов = 1128
Время = 25000 : количество серверов = 1120
Время = 25500 : количество серверов = 1114
Время = 26000 : количество серверов = 1113
Время = 26500 : количество серверов = 1112
Время = 27000 : количество серверов = 1113
Время = 27500 : количество серверов = 1104
```

In [8]:
```
plt.figure(figsize=(15,10))
plt.plot(time, result, label='предсказание с оценкой lambda
')
plt.plot(time, prediction, label='предсказание с известным
lambda')
plt.ylabel("Количество поломанных серверов", fontsize=17)
plt.xlabel("Время", fontsize=17)
plt.ylim((1000))
plt.legend(fontsize=17)
plt.grid()
plt.show()
```



Построим график зависимости модуля разности значения параметра λ и его оценки от времени.

In [9]:
```
plt.figure(figsize=(15,10))
plt.plot(time, abs(lambda_ - real_lambda_))
plt.ylabel(r'$| \lambda - \hat{\lambda}|$', fontsize = 17)
plt.xlabel("Время", fontsize=17)
plt.grid()
plt.show()
```



Вывод:

Сначала оценка плохо приближает параметр λ, но с увеличением количества считанных данных ситуация улучшается и оценка сходится.