Задача 7.4

In [1]:

```
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:

```
file_obj = open('6.csv')
print file_obj.readline().strip()
print file_obj.readline().strip()
print file_obj.readline().strip()
```

```
lambda = 95
t_0 = 500
t = 100000
```

In [3]:

```
t_0 = 500
t = 100000
```

In [4]:

```
data = np.array([float(line.strip()) for line in file_obj])
```

Найдём $E(N_t|N_s)$:

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

$(N_t - N_s)\perp\!\!\!\perp N_s \Rightarrow E(N_t - N_s|N_s) = E(N_t - N_s)$

$N_t - N_s \sim Pois(\lambda \cdot (t - s)) \Rightarrow E(N_t - N_s) = \lambda \cdot (t - s)$

N_s - F_{N_s}-измеримая случайная величина, значит $E(N_s|N_s) = N_s$. Тогда

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

Сопряженное распределение к экспоненциальному - $\Gamma(\alpha, \beta)$. Тогда байесовская оценка параметра λ - $\frac{n+\alpha}{\beta+\sum_{i=1}^{n} \xi_i}$, где ξ_i - время между i-ым и (i + 1)-ым выходами из строя сервера, $\xi_1, \ldots \xi_n \sim Exp(\lambda)$.

In [5]:

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

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

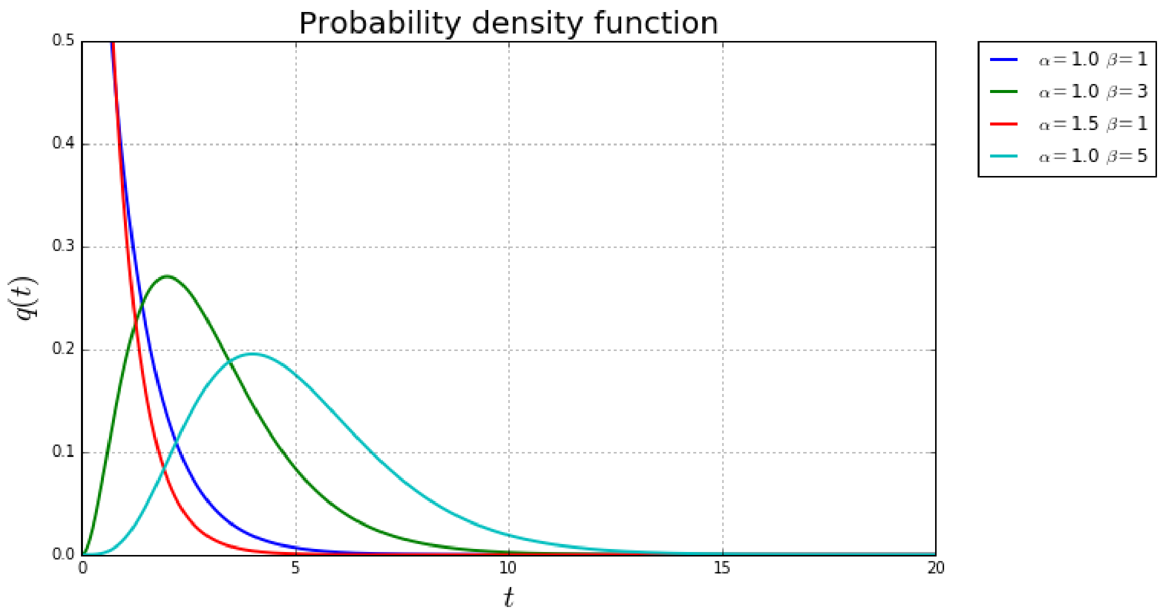
In [21]:

```
x = np.linspace(0, 20, 1000)

_alpha = np.array([1, 1, 1.5, 1])
_beta = np.array([1, 3, 1, 5])

plt.figure(figsize=(10, 6))

for i in range(4):
    plt.plot(x, sps.gamma.pdf(x, a=_beta[i], scale=1/_alpha[i]), linewidth = 2, label
             r' $\beta = $' + str(_beta[i]))
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlim(0, 20)
plt.ylim(0, 0.5)
plt.xlabel('$t$', fontsize = 20)
plt.ylabel('$q(t)$', fontsize = 20)
plt.title('Probability density function', fontsize = 20)
plt.grid()
plt.show()
```



Из графиков плотности гамма распределения видно, что при $\beta > 1$ максимальная плотность при $t > 0$, значит, так как нет никаких данных о λ, нужно взять параметр априорного распределения $\beta = 1$. От параметра α зависит то, как график прижимается к вертикальной оси. Считаем, что скорее всего значение λ - в районе 0. Тогда положим $\alpha = 1$:

In [22]:

```
alpha = 1
beta = 1
```

In [26]:

```
s = np.arange(0, t + t_0, t_0)
N_s = np.array([(data <= cur_t).sum(0) for cur_t in s])
lambda_ = np.zeros(N_s.size)
xi = xi.cumsum()
for i in range(N_s.size):
    lambda_[i] = (N_s[i] + alpha) / (xi[N_s[i] - 1] + beta)
result = lambda_ * (t - s) + N_s
```

Вывод программы, предсказывающей сколько серверов нужно докупить к моменту времени t:

In [19]:

```
for i in range(s.size):
    print r'time = %d: %d' % (s[i], result[i])
```

```
time = 0: 1
time = 500: 952
time = 1000: 1330
time = 1500: 1331
time = 2000: 1349
time = 2500: 1255
time = 3000: 1173
time = 3500: 1120
time = 4000: 1087
time = 4500: 1119
time = 5000: 1177
time = 5500: 1176
time = 6000: 1112
time = 6500: 1090
time = 7000: 1032
time = 7500: 1026
time = 8000: 1037
time = 8500: 1038
time = 9000: 1104
time = 9500: 1092
time = 10000: 1106
time = 10500: 1114
time = 11000: 1116
time = 11500: 1129
time = 12000: 1110
time = 12500: 1103
time = 13000: 1091
time = 13500: 1116
time = 14000: 1137
time = 14500: 1135
time = 15000: 1148
time = 15500: 1140
time = 16000: 1145
time = 16500: 1136
time = 17000: 1129
time = 17500: 1133
time = 18000: 1091
time = 18500: 1089
time = 19000: 1087
time = 19500: 1093
time = 20000: 1093
time = 20500: 1095
time = 21000: 1109
time = 21500: 1110
time = 22000: 1110
time = 22500: 1119
time = 23000: 1128
time = 23500: 1129
time = 24000: 1126
time = 24500: 1119
time = 25000: 1117
time = 25500: 1110
```

```
time = 26000: 1107
time = 26500: 1108
time = 27000: 1104
time = 27500: 1102
time = 28000: 1104
time = 28500: 1096
time = 29000: 1098
time = 29500: 1091
time = 30000: 1101
time = 30500: 1093
time = 31000: 1095
time = 31500: 1098
time = 32000: 1096
time = 32500: 1092
time = 33000: 1095
time = 33500: 1102
time = 34000: 1113
time = 34500: 1119
time = 35000: 1123
time = 35500: 1122
time = 36000: 1123
time = 36500: 1124
time = 37000: 1121
time = 37500: 1124
time = 38000: 1126
time = 38500: 1131
time = 39000: 1124
time = 39500: 1122
time = 40000: 1121
time = 40500: 1116
time = 41000: 1120
time = 41500: 1116
time = 42000: 1122
time = 42500: 1125
time = 43000: 1126
time = 43500: 1124
time = 44000: 1116
time = 44500: 1108
time = 45000: 1114
time = 45500: 1130
time = 46000: 1124
time = 46500: 1125
time = 47000: 1138
time = 47500: 1130
time = 48000: 1125
time = 48500: 1125
time = 49000: 1133
time = 49500: 1133
time = 50000: 1130
time = 50500: 1120
time = 51000: 1116
time = 51500: 1111
time = 52000: 1105
time = 52500: 1106
time = 53000: 1113
time = 53500: 1111
time = 54000: 1106
```

```
time = 54500: 1109
time = 55000: 1108
time = 55500: 1103
time = 56000: 1101
time = 56500: 1099
time = 57000: 1112
time = 57500: 1108
time = 58000: 1109
time = 58500: 1112
time = 59000: 1112
time = 59500: 1109
time = 60000: 1105
time = 60500: 1103
time = 61000: 1102
time = 61500: 1110
time = 62000: 1114
time = 62500: 1112
time = 63000: 1111
time = 63500: 1113
time = 64000: 1109
time = 64500: 1108
time = 65000: 1110
time = 65500: 1106
time = 66000: 1100
time = 66500: 1099
time = 67000: 1097
time = 67500: 1093
time = 68000: 1094
time = 68500: 1092
time = 69000: 1087
time = 69500: 1086
time = 70000: 1090
time = 70500: 1089
time = 71000: 1088
time = 71500: 1085
time = 72000: 1084
time = 72500: 1081
time = 73000: 1081
time = 73500: 1076
time = 74000: 1077
time = 74500: 1077
time = 75000: 1074
time = 75500: 1077
time = 76000: 1076
time = 76500: 1073
time = 77000: 1071
time = 77500: 1077
time = 78000: 1077
time = 78500: 1079
time = 79000: 1079
time = 79500: 1077
time = 80000: 1075
time = 80500: 1075
time = 81000: 1075
time = 81500: 1070
time = 82000: 1065
time = 82500: 1066
```

```
time = 83000: 1061
time = 83500: 1060
time = 84000: 1057
time = 84500: 1056
time = 85000: 1055
time = 85500: 1053
time = 86000: 1061
time = 86500: 1060
time = 87000: 1056
time = 87500: 1055
time = 88000: 1052
time = 88500: 1056
time = 89000: 1056
time = 89500: 1052
time = 90000: 1053
time = 90500: 1052
time = 91000: 1051
time = 91500: 1050
time = 92000: 1047
time = 92500: 1050
time = 93000: 1054
time = 93500: 1053
time = 94000: 1053
time = 94500: 1056
time = 95000: 1052
time = 95500: 1047
time = 96000: 1042
time = 96500: 1036
time = 97000: 1031
time = 97500: 1026
time = 98000: 1021
time = 98500: 1015
time = 99000: 1010
time = 99500: 1005
time = 100000: 1000
```