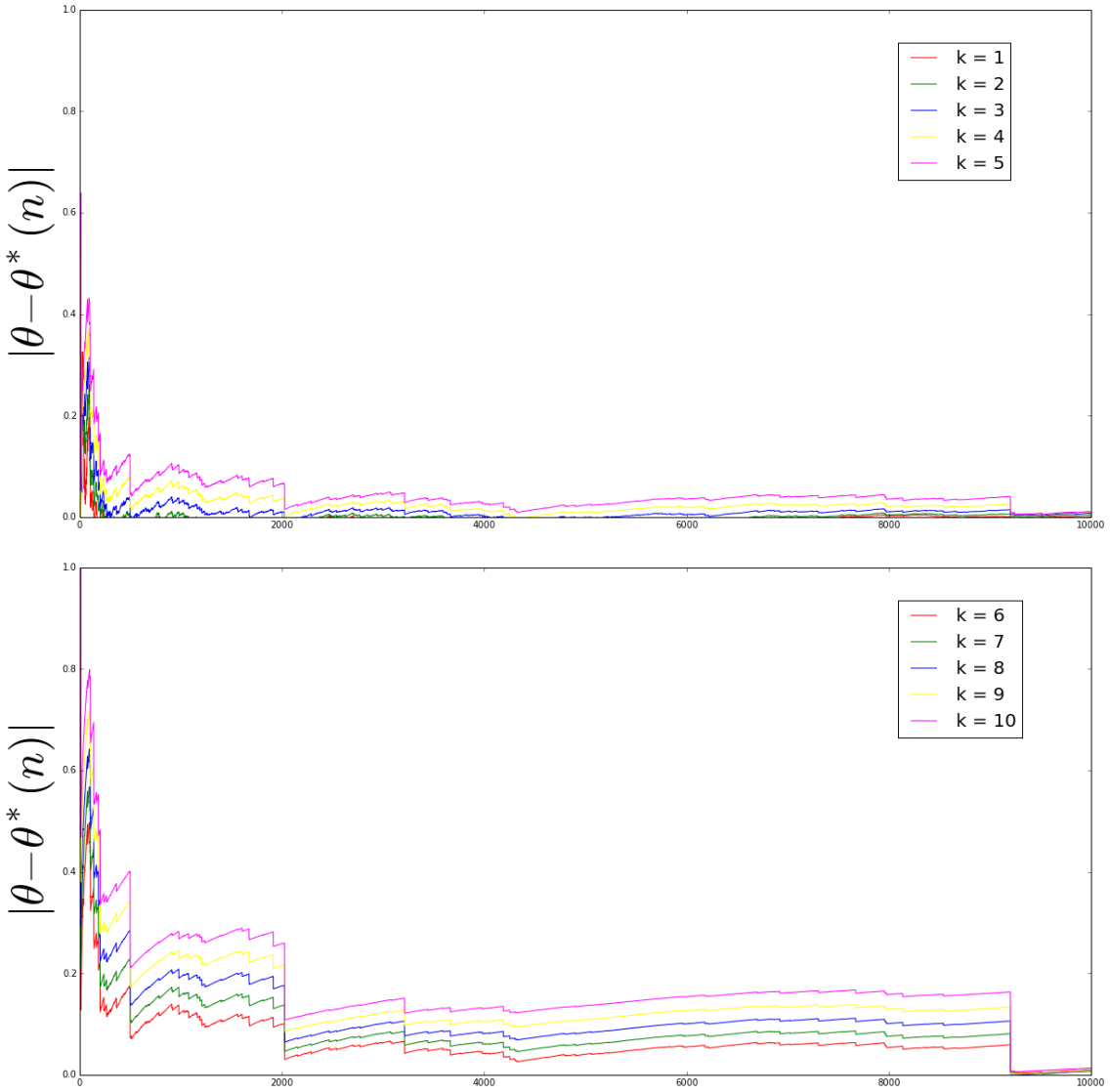
In [ ]:

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

sample_size = 10000
parameter = 1
magic_number = 11

#генерируем выборку из экспоненциального распределения с параметром 1
buf_sample = sps.expon.rvs(size = sample_size, loc = 0, scale = parameter)

#считаем страшную оценку, точнее сразу же разность между оценкой и истинным значен
ием
fact = 1
buf_sum = 0

#сразу же строим графики, построим на одном графике не более 5-ти кривых
color = ['red', 'green', 'blue', 'yellow', 'magenta']
check = 1
counter = 5
plt.figure(figsize=(20,10))

for k in range(1, magic_number, 1):

    result = []
    fact *= k
    buf_sum = 0

    for i in range(sample_size):
        buf_sum += buf_sample[i] ** ( k )
        result.append( ( fact / buf_sum * (i + 1) )**( 1 / ( k ) ) - parameter )
    plt.plot(np.arange(1, sample_size + 1), result, color = color[(k - 1) % 5], la
bel =('k = ' + str( k ) ) )

    plt.legend(loc='center left', bbox_to_anchor=(0.8, 0.8), fontsize = 20)

    if ( k == counter ):

        #чтобы нарисовать не более 5 графиков
        plt.ylabel('$|\\theta - \\theta^*(n)|$', fontsize = 50)
        plt.ylim(0, parameter)
        plt.show()

        plt.figure(figsize=(20,10))
        counter += 5
```

Мы видим,что при меньших к оценки ведут себя лучше при малых n

In [33]:



<matplotlib.figure.Figure at 0x7f0f04186860>