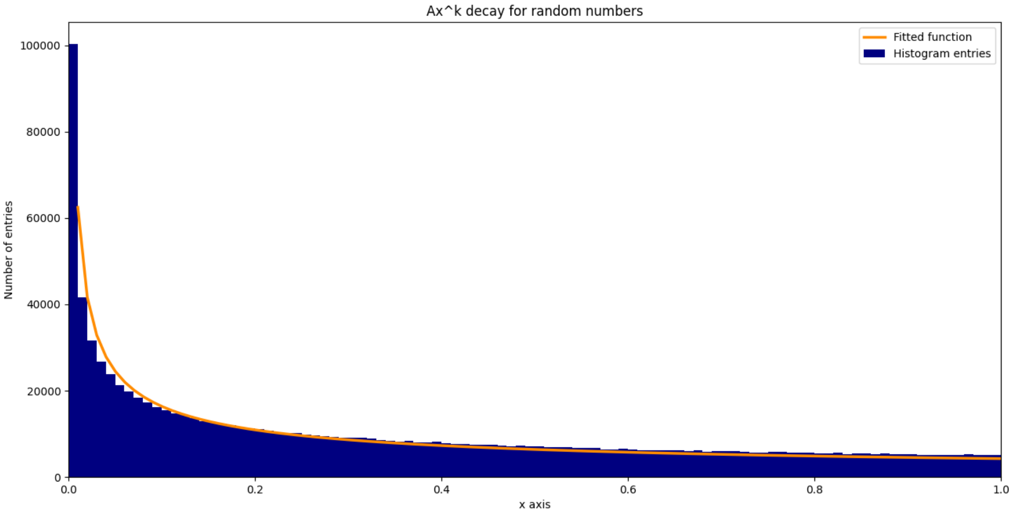
Random number problem

```
# 1.) Necessary imports.
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


# 2.) Define fit function.
def fit_function(x, A, k):
    return (A * x ** k)

# 3.) Generate exponential and gaussian data and histograms.
N = 1000000
data = np.random.rand(N,1)**2
bins = np.linspace(0, 1, 101)
data_entries, bins = np.histogram(data, bins=bins)

# 4.) Add histograms of exponential and gaussian data.
binscenters = np.array([0.5 * (bins[i] + bins[i+1]) for i in range(len(bins)-1)])

# 5.) Fit the function to the histogram data.
popt, pcov = curve_fit(fit_function, xdata=binscenters, ydata=data_entries)
print(popt)

# 6.)
# Generate enough x values to make the curves look smooth.
xspace = np.linspace(0, 1, 100)

# Plot the histogram and the fitted function.
plt.bar(binscenters, data_entries, width=bins[1] - bins[0], color='navy', label=r'Histogram entries')
plt.plot(xspace, fit_function(xspace, *popt), color='darkorange', linewidth=2.5, label=r'Fitted function')

# Make the plot nicer.
plt.xlim(0,1)
plt.xlabel(r'x axis')
plt.ylabel(r'Number of entries')
plt.title(r'Ax^k decay for random numbers')
plt.legend(loc='best')
plt.show()
plt.clf()
```

Results

Fitted parameter

For the function, $y = Ax^k$ fitted values are A = 4303.39744 and k = -0.5804 for bin size of 100. The fitting parameter k is not changing much with increasing value of bin. So, I treat it a good fit.


Ax^k decay for random numbers
Fitted function
Histogram entries
Number of entries
0
20000
40000
60000
80000
100000
x axis
0.0
0.2
0.4
0.6
0.8
1.0