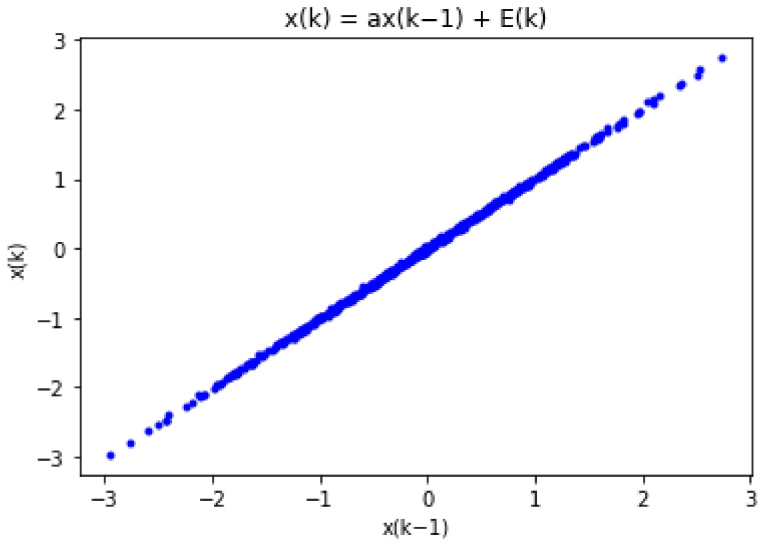
In [3]:

```python
# the coding part for the report in Assignment #1

import numpy as np
from random import gauss
from random import seed
from pandas import Series
import pandas as pd
from pandas.plotting import autocorrelation_plot
import matplotlib.pyplot as plt

# Step 1: Generating the data

# seed random number generator
seed(1)

# equation(1)
# x(k) = ax(k−1) + ε(k)
a = 0.99

number = list(range(1,503))        # k
number_2 = list(range(0,502))      # k-1

# create white noise series  ε(k)
noise = [gauss(0.0, 0.02) for i in range(502)]   # gauss(mean, variance)
noise = Series(noise)

# ------------------------- input values x(k−1) -----------------------------
 #
# create random input samples x(k)

inputs = [gauss(0.0, 0.995) for i in range(502)]   # gauss(mean, variance)
inputs = list(Series(inputs))                      # input values x(k−1)

# --------------------- estimated output values x(k) ------------------------
 #
# By using the quation(1), we can come out with numbers of x(k) shown below:

outputs = list((inputs-noise)/a)                   # estimated output values x
(k)


# ==============================================================================
#
# --the organized form of k, k-1, input values, and estimated output values --
#
# ==============================================================================
#

data = {'k-1': number_2,                           # k-1
    'input values x(k-1)': inputs,                 # input values
    'k': number,                                   # k
    'estimated output values x(k)': outputs} # estimated output values
df = pd.DataFrame(data)


# plot_x(k) = ax(k−1) + ε(k)
```

```
plt.plot(inputs, outputs, 'b.')                  # plot x(k)
plt.title('x(k) = ax(k-1) + E(k)')
plt.xlabel('x(k-1)')
plt.ylabel('x(k)')
plt.show()
df                                               # display the form for all values
```



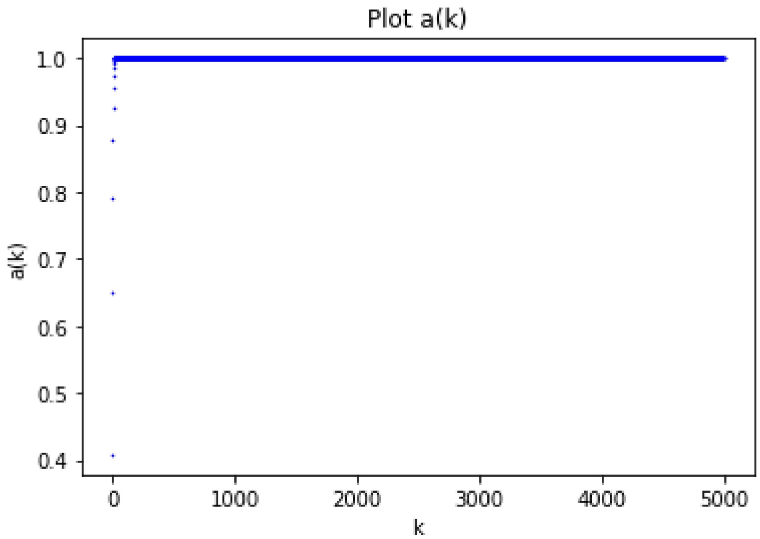
Out[3]:

	k-1	input values $x(k$-1)	k	estimated output values $x(k)$
0	0	-0.040257	1	-0.066687
1	1	-0.007259	2	-0.036614
2	2	-1.016622	3	-1.028231
3	3	0.385505	4	0.404845
4	4	-0.339148	5	-0.320510
...	...	...	...	...
497	497	1.538949	498	1.536485
498	498	-0.446621	499	-0.435470
499	499	0.277009	500	0.287490
500	500	0.430049	501	0.453369
501	501	1.201860	502	1.214361

502 rows × 4 columns

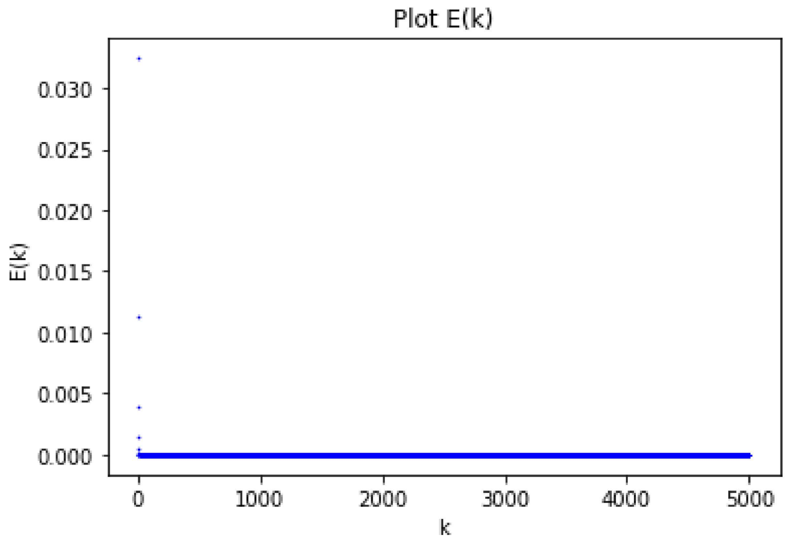
In [7]:

```python
# ============================================================================
#
# ---------------------------- LMS Algorithm, a(k) -------------------------
#
# ============================================================================
#

# Step 2: estimate the parameter a in equation (1) using the LMS algorithm
# Step 3: Record the final estimated a value.
#         And Plot a(n) and also the instantaneous cost

a = 0.99      # the value from Step 1
aa = 0        # start from a(0) = 0
iter = 5000
lr = 0.001    # learning rate
k = 500


# -------------------------- input values (1~500) --------------------------
#
inputv = np.array(df.loc[1:500, 'input values x(k-1)'])
# print(inputv)

# -------------------- desire output values (2~501) -------------------------
#
d_outputv = np.array(df.loc[1:500, 'input values x(k-1)'])
# print(d_outputv)

# ------------------- estimated output values (2~501) -----------------------
#
outputv = np.array(df.loc[2:501, 'estimated output values x(k)'])
# print(outputv)


# ---------------------------------------------------------------------------
#


# plot - k vs. a(k)
for _ in range(iter):
    for i in range(k):
        error = d_outputv[i] - inputv[i]*aa  # error function
        aa = aa + lr*error*inputv[i]         # find the last estimated paramet
er a
#     print(aa)                              # print to see the pattern of
#                                              running estimated values
    plt.plot(_, aa, 'xb-', markersize=1)     # plot a(k)
plt.title('Plot a(k)')
plt.xlabel('k')
plt.ylabel('a(k)')
plt.show()


# True value of a vs. estimate of a
print('True value of a: ', a)
print('Estimate of a: ', aa)
```




```
True value of a:  0.99
Estimate of a:  0.9999999999999937
```

```
In [9]: # ============================================================================
        #
        # --------------------- LMS Algorithm, cost function E(k)---------------------
        #
        # ============================================================================
        #

        a = 0.99     # the value from Step 1
        aa = 0       # start from a(0) = 0
        iter = 5000
        lr = 0.001   # learning rate
        k = 500


        # -------------------------- input values (1~500) ----------------------------
        #
        inputv = np.array(df.loc[1:500, 'input values x(k-1)'])
        # print(inputv)

        # -------------------- desire output values (2~501) --------------------------
        #
        d_outputv = np.array(df.loc[1:500, 'input values x(k-1)'])
        # print(d_outputv)

        # ------------------- estimated output values (2~501) ------------------------
        #
        outputv = np.array(df.loc[2:501, 'estimated output values x(k)'])
        # print(outputv)


        # ----------------------------------------------------------------------------
        #



        # plot - k vs. E(k) -- cost function
        for _ in range(iter):
            for i in range(k):
                error = d_outputv[i] - inputv[i]*aa  # error function
                aa = aa + lr*error*inputv[i]
                cost = 0.5*error*error               # find the last estimated value o
        f E(k)
        #     print(cost)                            # print to see the pattern of
        #                                               running estimated values
            plt.plot(_, cost, 'xb-', markersize=1)   # E(k)
        plt.title('Plot E(k)')
        plt.xlabel('k')
        plt.ylabel('E(k)')
        plt.show()

        # The last estimated value of E(k) before the errors are no more
        print('Estimate of E(k): ', cost)     # the value can be considered zero
```




```
Estimate of E(k):  3.6993262371790026e-30
```