```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

# static batch estimation of a shape defined as a quadratic
# function z = f(p, q) + v, and parametrized using a vector x


def shape_basis(p, q):

    f = np.transpose(np.concatenate(
        (np.array([p**2, q**2, p * q, p, q]), np.ones((1, np.size(p))))))

    return f


def shape(p, q, x):
    z = shape_basis(p, q) @ x
    return z


# workspace is the square [s, s]x[-s, s]
s = 10

# true shape parameter (i.e. a symmetric cup)
x_true = np.array([1.2, 1.3, 1, 1, 1, 1])
P = np.diag([16, 16, 16, 16, 16, 16])      # initial prior

# plot true
test = shape(np.array([-s]), np.array([s]), x_true)
s1 = np.arange(-s, s, 0.1)
s2 = np.arange(-s, s, 0.1)
S1, S2 = np.meshgrid(s1, s2)
gt = np.zeros((S1.shape[0], S1.shape[1]))
for i in range(S1.shape[0]):
    for j in range(S1.shape[1]):
        gt[i, j] = shape(np.array([S1[i, j]]), np.array([S2[i, j]]), x_true)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surface = ax.plot_surface(S1, S2, gt, cmap=cm.jet, linewidth=0,
                          antialiased=False, alpha=0.3)

# measurement standard dev
std = 20

# number of measurements
k = 2
x = x_true
for i in range(int(8/k)):
    # generate random measurements
    p = 4 * s * (np.random.rand(k) - 0.5)
    q = 4 * s * (np.random.rand(k) - 0.5)
    z = shape(p, q, x_true) + np.random.randn(k) * std

    # estimate optimal parameters x
    R = np.diag(np.ones(k) * std**2)
    H = shape_basis(p, q)
    P = np.linalg.inv(np.linalg.inv(P) + np.transpose(H) @ np.linalg.inv(R)
                      @ H)
    K = P @ np.transpose(H) @ np.linalg.inv(R)
    x = x + K @ (z - H @ x)

# plot estimated
ge = np.zeros((S1.shape[0], S1.shape[1]))
for i in range(S1.shape[0]):
    for j in range(S1.shape[1]):
        ge[i, j] = shape(np.array([S1[i, j]]), np.array([S2[i, j]]), x)
surface = ax.plot_surface(S1, S2, ge, cmap=cm.jet, linewidth=0,
                          antialiased=False, alpha=0.8)
ax.set_title("k = 2 with 4 iterations")
```

```
# ax.set_title("Batch Estimation")


# print(xs.shape)
plt.show()
```