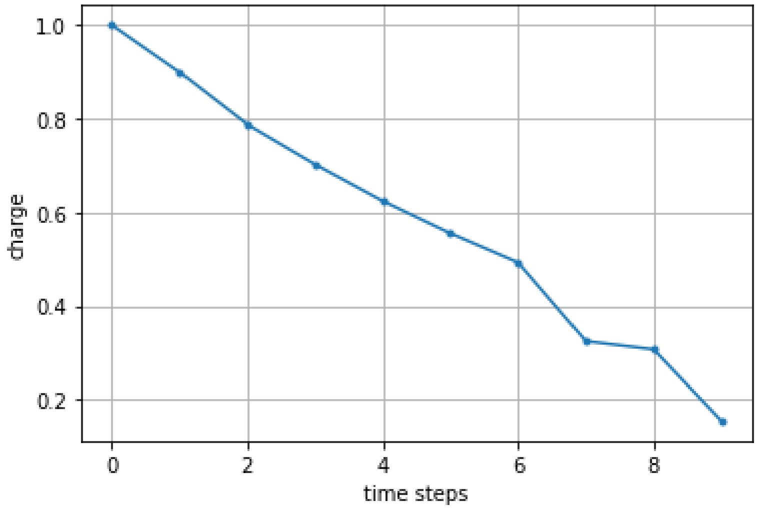
In [ ]:
```
# importing nessery libraries:
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
n_steps = 9

d = 1 # dynamic

# j0 = 0.1, σv = 0.05, and σw = 0.1
σv2 = 0.05**2
σw2 = 0.1**2

q0 = 1  # q(0) = 1
uncertainty_Vr0 = 0

true_state = np.random.normal(1,0)
```

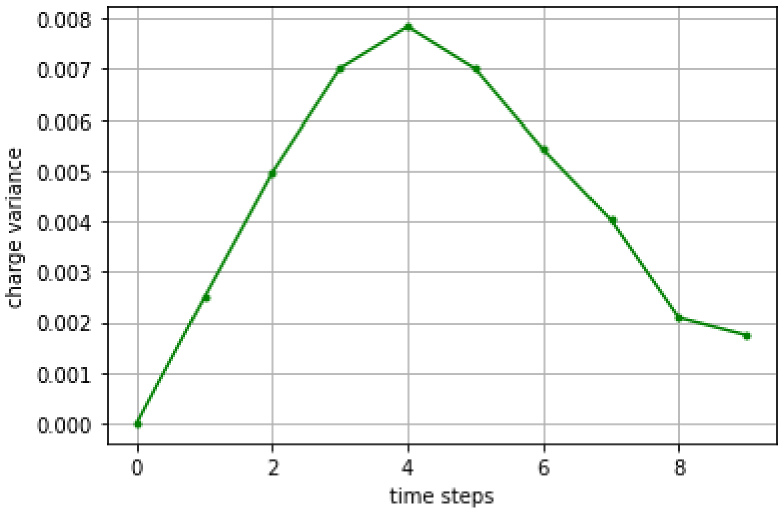
In [ ]:
```
charges = np.zeros([n_steps+1,1]) # to store the charges.
charges[0,0] = q0

uncertainty = np.zeros([n_steps+1,1])
uncertainty[0,0] = uncertainty_Vr0

measurments_ = np.zeros([n_steps+1,1]) # to store measurments.
```

In [ ]:
```
for k in np.arange(1,n_steps+1):

    noise = np.random.normal(0, 0.05)

    true_state = true_state - noise - 0.1

    w = np.random.normal(0, 0.1)

    measurments_[k,0] = 4 + ((true_state)-1)**3 + w


    Kp = q0 - 0.1
    K_uncertainty = d*uncertainty_Vr0*d + σv2

    H = 3*(Kp-1)**2
    K = K_uncertainty*H*1/(H*K_uncertainty*H + σw2)

    AVGs = 4 + ((Kp)-1)**3

    q0 = Kp + K*(measurments_[k,0]-AVGs)
    uncertainty_Vr0 = (1 - K*H)*K_uncertainty*(1 - K*H) + K*σw2*K

    charges[k,0] = q0
    uncertainty[k,0] = uncertainty_Vr0
```

In [ ]:
```
plt.plot(charges[:,0],'.-')
plt.xlabel('time steps')
plt.ylabel('charge')
plt.grid(True)
```



In [ ]:
```
plt.plot(uncertainty[:,0],'g.-')
plt.xlabel('time steps')
plt.ylabel('charge variance')
plt.grid(True)
```

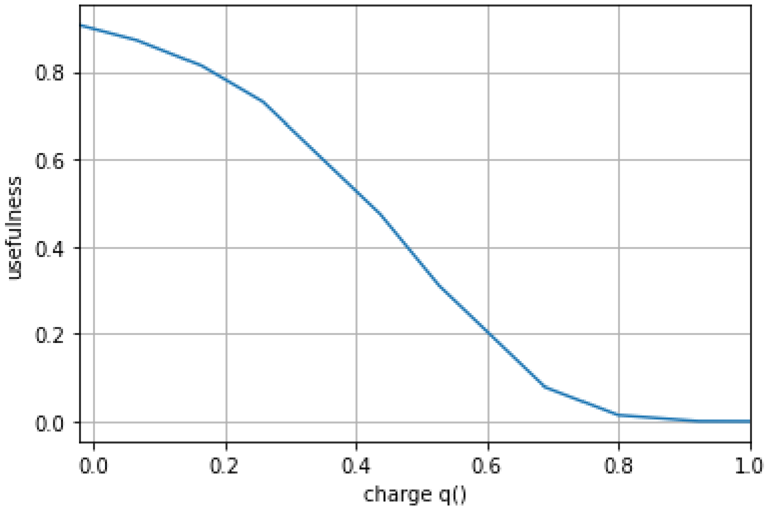


In [ ]:
```
def q(x, u, v):
    return  x - u - v

q0 = 1
σw = u = 0.1

time_steps = np.arange(n_steps+1)
Xs = np.zeros(n_steps+1)
Hs = np.zeros(n_steps+1)
Us = np.zeros(n_steps+1)

Xs[0] = q0
Hs[0] = (q0-1)**2
Us[0]=(Hs[0]**2*0.1)/(Hs[0]**2*0.1+σw)
```

In [ ]:
```
for k in range(n_steps):
        v = np.random.normal(0,0.05)
        Xs[k+1] = q(Xs[k], u, v)
        Hs[k+1] = 3*(Xs[k+1]-1)**2
        Us[k+1]=(Hs[k+1]**2*0.1)/(Hs[k+1]**2*0.1+σw)
```

In [ ]:
```
plt.plot(Xs, Us)
plt.xlim([Xs[-1], Xs[0]])
plt.xlabel('charge q()')
```

```
plt.ylabel('usefulness')
plt.grid(True)
```

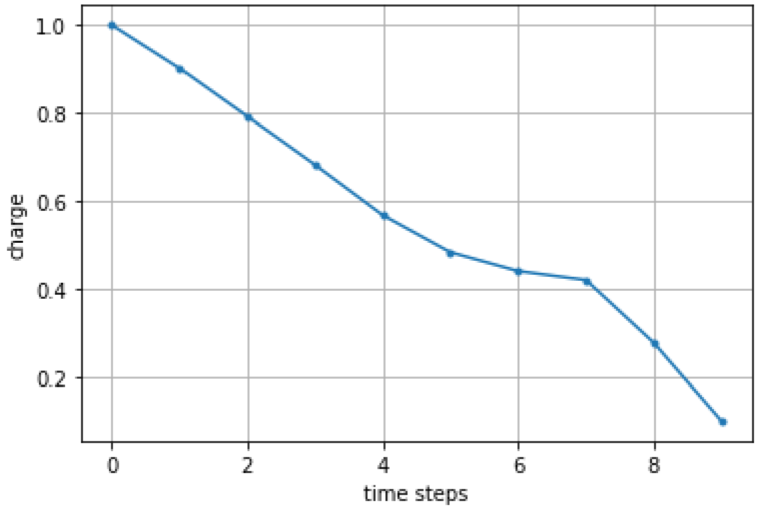


In [ ]:
```
q0 = 1  # q(0) = 1
uncertainty_Vr0 = 0

states = np.zeros([n_steps+1,1]) # to store the history of the true states.
true_state = np.random.normal(1,0)
states[0,0] = true_state
```

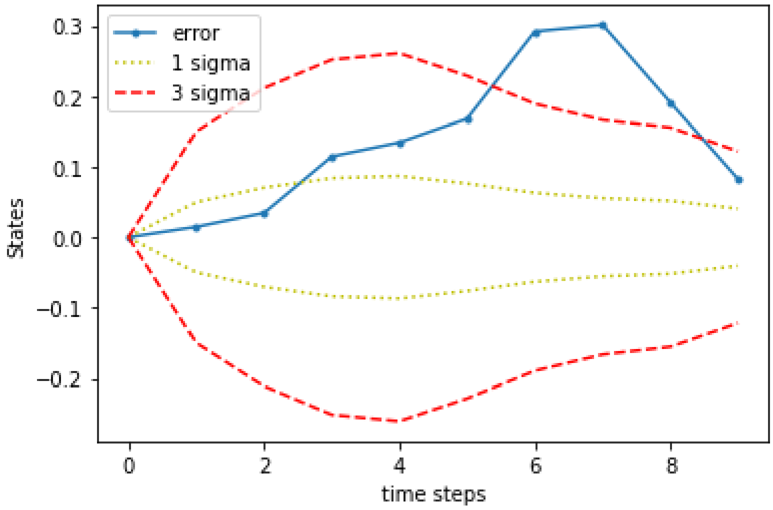
In [ ]:
```
measurments = [4.21, 3.83, 3.92, 3.89, 3.88, 3.89, 3.91, 3.57, 3.21]
```

In [ ]:
```
for k in np.arange(1,n_steps+1):

    noise = np.random.normal(0, 0.05)

    true_state = true_state - noise - 0.1

    w = np.random.normal(0, 0.1)

    measurments_[k,0] = measurments[k-1]

    Kp = q0 - 0.1
    K_uncertainty = d*uncertainty_Vr0*d + σv2

    H = 3*(Kp-1)**2
    K = K_uncertainty*H*1/(H*K_uncertainty*H + σw2)

    AVGs = 4 + ((Kp)-1)**3

    q0 = Kp + K*(measurments_[k,0]-AVGs)
    uncertainty_Vr0 = (1 - K*H)*K_uncertainty*(1 - K*H) + K*σw2*K

    charges[k,0] = q0
    uncertainty[k,0] = uncertainty_Vr0
    states[k,0] = true_state
```

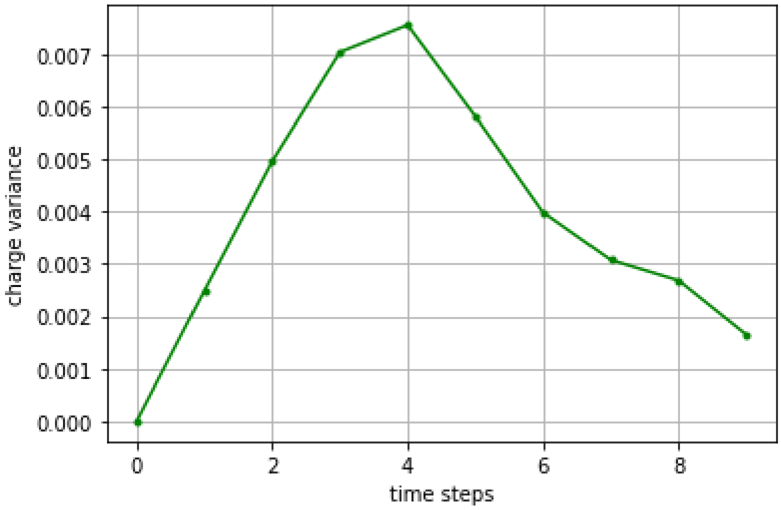
In [ ]:
```
plt.plot(charges[:,0],'.-')
plt.xlabel('time steps')
plt.ylabel('charge')
plt.grid(True)
```



In [ ]:
```
plt.plot(charges[:,0]-states[:,0],'.-',label="error")
plt.plot(np.sqrt(uncertainty[:,0]),'y:',label="1 sigma")
plt.plot(-np.sqrt(uncertainty[:,0]),'y:',)
plt.plot(3*np.sqrt(uncertainty[:,0]),'r--',label="3 sigma")
plt.plot(-3*np.sqrt(uncertainty[:,0]),'r--',)
plt.xlabel('time steps')
plt.ylabel('States')
plt.legend()
```



In [ ]:
```
plt.plot(uncertainty[:,0],'g.-')
plt.xlabel('time steps')
plt.ylabel('charge variance')
plt.grid(True)
```



In [ ]:
```
# since...
j0=0.1
sigma=0.05
# then...
mean_q=1-9*j0
var_q=9*sigma**2

print(f'the mean if I did not have the voltage measurements: {round(mean_q,4)}')
print(f'the variance if I did not have the voltage measurements: {round(var_q,4)}'
```

```
the mean if I did not have the voltage measurements: 0.1
the variance if I did not have the voltage measurements: 0.0225
```

In [ ]:
```
plt.plot(measurments_[:,0],'-',label="z")
plt.plot([0] + measurments,'yo',label="z_m")

plt.xlabel('time steps')
plt.ylabel('measurements')
plt.legend();
```