In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg as sla
fig_size = (12,8)
```

Homework 10

Eugeniu Vezeteu - 886240

$$x_k = tanh(x_{k-1}) + q_{k-1}$$
$$y_k = sin(x_k) + r_k$$

Where $x_0 = N(0,1)$, $q_{k-1} = N(0,1)$ and $r_k = N(0,1)$

a)Simulate 100 steps of states and measurements from the model. Plot the data.

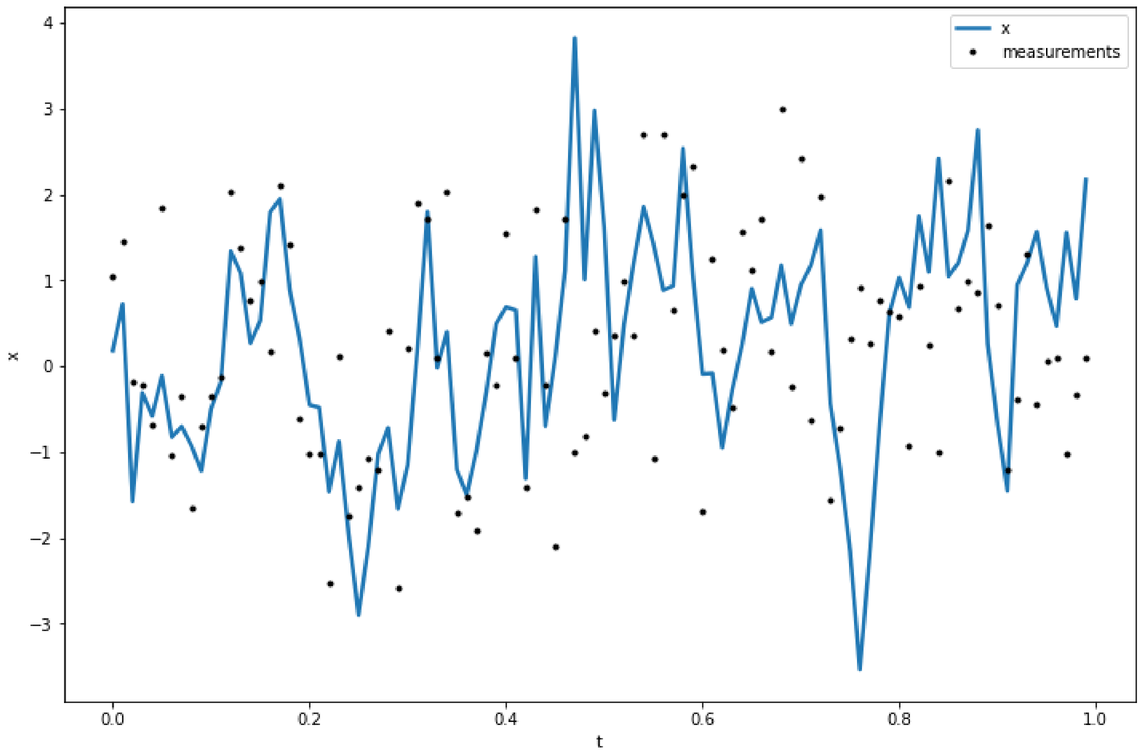
In [2]:

```
def propagate_state(x_init, length):
    x = np.zeros((length, x_init.shape[0]))
    x[0] = x_init
    for i in range(length-1):
        q = np.random.normal(loc=0, scale=1, size=1)
        x[i+1] = np.tanh(x[i]) + q
    return x

dt = 0.01
n = 100

x_init = np.random.normal(loc=0,scale=1.0,size=1)
t = np.arange(n)*dt
r = np.random.normal(loc=0,scale=1.0,size=n)
x = propagate_state(x_init,t.shape[0]).squeeze(-1)          #state
y = np.sin(x) + r                                          #measurements
print('x:{},  y:{}'.format(np.shape(x), np.shape(y)))

plt.figure(figsize=fig_size)
plt.plot(t,x,linewidth=2.5, label='x')
plt.plot(t,y,'ok',markersize=3,alpha=1, label='measurements')
plt.xlabel('t')
plt.ylabel('x')
plt.legend()
```

```
x:(100,),  y:(100,)
```

Out[2]:

```
<matplotlib.legend.Legend at 0x7f50b4e4c8d0>
```



Derive the necessary derivatives and check that they are correct by using numerical finite differences.

The derivative of measurement model is $\frac{\partial h}{\partial x} = \frac{\partial sin(x)}{\partial x} = cos(x)$.

The derivative of the motion model is $\frac{\partial tanh(x)}{\partial x}$, which is:

$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, So we have:$
$\frac{\partial}{\partial x} tanh(x) = \frac{(e^x+e^{-x})(e^x+e^{-x})-(e^x-e^{-x})(e^x-e^{-x})}{(e^x+e^{-x})^2} = 1 - \frac{(e^x-e^{-x})^2}{(e^x+e^{-x})^2} = 1 - tanh^2(x)$ $

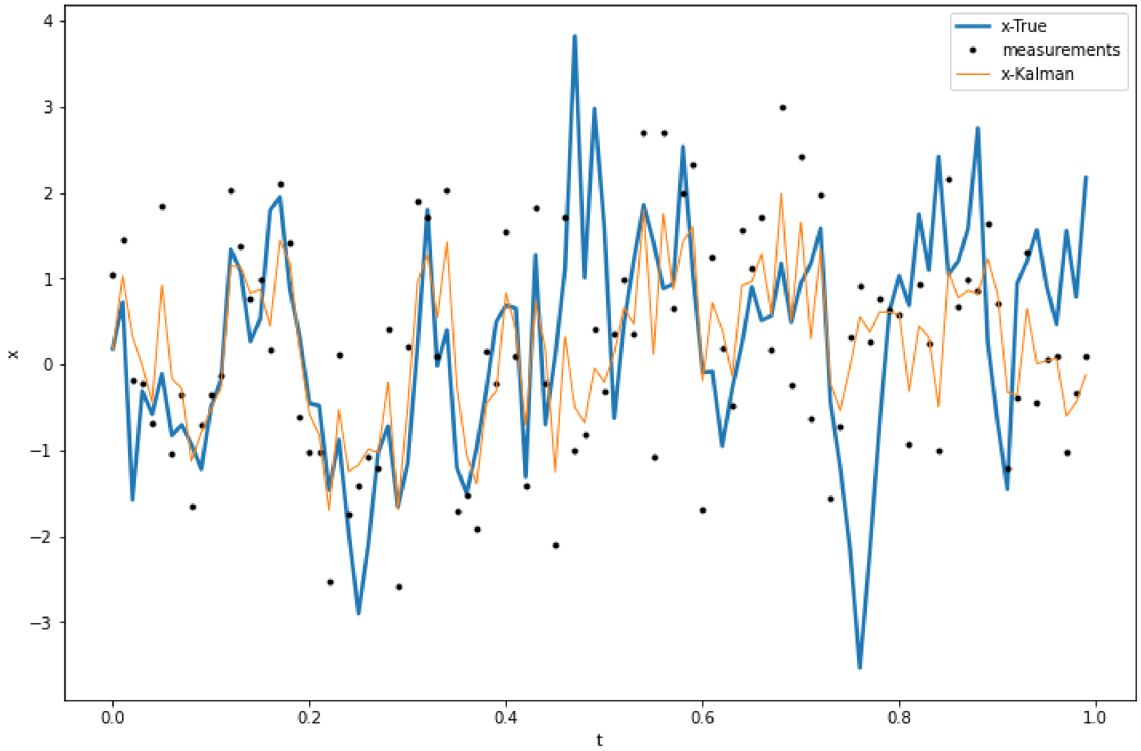
In [3]:

```
def df_dx(x):
    return np.array([1-(np.tanh(x)**2)])

def dh_dx(x):
    return np.array([np.cos(x)])
```

c)Implement and run an EKF for the model. Plot the results.

In [4]:

```
Q = 1.    # process variance
R = 1**2  # measurement variance

P=np.zeros(n)
xhat=np.zeros(n)
xhat[0] = x_init
P[0] = 1.0

def f(x):
    return np.tanh(x)

def h(x):
    return np.sin(x)

for i in range(1,n):
    x_ = f(xhat[i-1])
    F_x    = df_dx(xhat[i-1])
    P_     = F_x*P[i-1]*F_x.T + Q

    H_x    = dh_dx(x_)
    S      = H_x*P_*H_x.T + R
    K_tran = sla.solve(S,H_x@P_)

    P[i] = P_ - K_tran.T*S*K_tran
    xhat[i] = x_ + K_tran.T*(y[i]-h(x_))

plt.figure(figsize=fig_size)
plt.plot(t,x,linewidth=2.5, label='x-True')
plt.plot(t,y,'ok',markersize=3,alpha=1, label='measurements')
plt.plot(t,xhat,linewidth=1, label='x-Kalman')

plt.legend()
plt.xlabel('t')
plt.ylabel('x')

plt.show()
```



In [ ]: