```python
################################################################################
# IMPORTS
################################################################################
import numpy as np
import os
import matplotlib.pyplot as plt
import matplotlib as mpl
from kalman import kalman_filter
##############################################################################
# Variable declaration
##############################################################################
isPlotReqd = True
isPlotPdf = True
##############################################################################
# Settings for plot
##############################################################################
if (True == isPlotReqd):
    if (True == isPlotPdf):
        mpl.use('pdf')
        fig_width  = 3.487
        fig_height = fig_width / 1.618
        rcParams = {
            'font.family': 'serif',
            'font.serif': 'Times New Roman',
            'text.usetex': True,
            'xtick.labelsize': 8,
            'ytick.labelsize': 8,
            'axes.labelsize': 8,
            'legend.fontsize': 8,
            'figure.figsize': [fig_width, fig_height]
          }
        plt.rcParams.update(rcParams)
##############################################################################
# CODE STARTS HERE
##############################################################################
##############################################################################
# Parameters
##############################################################################
x_0 = np.array([[650.], [250.]])
x_0_hat = np.array([[600.], [200.]])
P_0_hat = np.array([[500., 0.], [0., 200.]])
Q = np.array([[0., 0.], [0., 10.]])
H = np.array([[1, 0]])
R = 10.
F = np.array([[0.5, 2],[0, 1]])
G = np.zeros(x_0_hat.shape)
##############################################################################
# Time steps and process noise
##############################################################################
num_step = 10
n_historical_data = 1000
t = np.linspace(0, n_historical_data, num=n_historical_data+1)
w_f = np.zeros(t.shape)
w_f[1:] = np.sqrt(10.) * np.random.normal(0, 1, (t.shape[0]-1,))
w_k = np.vstack((np.zeros(t.shape), w_f))
##############################################################################
# True population and generate measurements
##############################################################################
x_true = np.zeros((x_0.shape[0], t.shape[0]))
x_true[:, 0] = x_0[:, 0]
y_k = np.zeros((1, t.shape[0]))
```

```python
# We will generate history for 1000 steps
for step in range(n_historical_data):
    x_true[:, step+1] = F @ x_true[:, step] + w_k[:, step]
    y_k[0, step+1] = H @ x_true[:, step+1] + np.sqrt(R) * np.random.normal()

################################################################################
# Kalman filter prediction
################################################################################
kf = kalman_filter(init_state=x_0_hat, init_est_err=P_0_hat, Q=Q, R=R, F=F, G=G, H=H, y_k=y_k, num_step=num_step)
kf.run_kalman()
P_k = kf.get_est_error_cov()
K_k = kf.get_kalman_gains()
x_k_0 = kf.get_predicted_state(0)
x_k_1 = kf.get_predicted_state(1)
#######################################################################
# Plot of true and estimated population
#######################################################################
if (True == isPlotReqd):
    ###########################################################################
    # Configure axis and grid
    ###########################################################################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    fig.subplots_adjust(left=.15, bottom=.16, right=.99, top=.97)

    ax.set_axisbelow(True)
    ax.minorticks_on()
    ax.grid(which='major', linestyle='-', linewidth='0.5')
    ax.grid(which='minor', linestyle="-.", linewidth='0.5')
    ax.plot(t[0:num_step+1], x_true[0, 0:num_step+1], label='True population')
    ax.plot(t[0:num_step+1], x_k_0, label='estimated population')

    ax.set_xlabel(r'time', fontsize=8)
    ax.set_ylabel(r'population', fontsize=8)

    plt.legend()
    if (True == isPlotPdf):
        if not os.path.exists('./generatedPlots'):
            os.makedirs('generatedPlots')
        fig.savefig('./generatedPlots/q4_population.pdf')
    else:
        plt.show()
#######################################################################
# Plot of true and estimated food supply
#######################################################################
if (True == isPlotReqd):
    ###########################################################################
    # Configure axis and grid
    ###########################################################################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    fig.subplots_adjust(left=.15, bottom=.16, right=.99, top=.97)

    ax.set_axisbelow(True)
    ax.minorticks_on()
    ax.grid(which='major', linestyle='-', linewidth='0.5')
    ax.grid(which='minor', linestyle="-.", linewidth='0.5')
    ax.plot(t[0:num_step+1], x_true[1, 0:num_step+1], label='True food supply')
    ax.plot(t[0:num_step+1], x_k_1, label='estimated food supply')

```

```
    ax.set_xlabel(r'time', fontsize=8)
    ax.set_ylabel(r'food supply', fontsize=8)

    plt.legend()
    if (True == isPlotPdf):
        if not os.path.exists('./generatedPlots'):
            os.makedirs('generatedPlots')
        fig.savefig('./generatedPlots/q4_food.pdf')
    else:
        plt.show()
########################################################################
# Plot of std dev of estimation error
########################################################################
if (True == isPlotReqd):
    ############################################################################
    # Configure axis and grid
    ############################################################################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    fig.subplots_adjust(left=.15, bottom=.16, right=.99, top=.97)

    ax.set_axisbelow(True)
    ax.minorticks_on()
    ax.grid(which='major', linestyle='-', linewidth='0.5')
    ax.grid(which='minor', linestyle="-.", linewidth='0.5')

    ax.plot(t[0:num_step+1], np.sqrt(P_k[0, 0, :]), color='g', label='Population estimation error std')
    ax.plot(t[0:num_step+1], -np.sqrt(P_k[0, 0, :]), color='g')
    ax.plot(t[0:num_step+1], np.sqrt(P_k[1, 1, :]), color='b', label='food estimation error std')
    ax.plot(t[0:num_step+1], -np.sqrt(P_k[1, 1, :]), color='b')

    ax.set_xlabel(r'time', fontsize=8)
    ax.set_ylabel(r'standard deviation', fontsize=8)

    plt.legend()
    if (True == isPlotPdf):
        if not os.path.exists('./generatedPlots'):
            os.makedirs('generatedPlots')
        fig.savefig('./generatedPlots/q4_std_dev.pdf')
    else:
        plt.show()
########################################################################
# Plot of Kalman gains
########################################################################
if (True == isPlotReqd):
    ############################################################################
    # Configure axis and grid
    ############################################################################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    fig.subplots_adjust(left=.15, bottom=.16, right=.99, top=.97)

    ax.set_axisbelow(True)
    ax.minorticks_on()
    ax.grid(which='major', linestyle='-', linewidth='0.5')
    ax.grid(which='minor', linestyle="-.", linewidth='0.5')

    ax.plot(t[0:num_step+1], K_k[0, :], label='Kalman gain for Population')
    ax.plot(t[0:num_step+1], K_k[1, :], label='Kalman gain for food')

    ax.set_xlabel(r'time', fontsize=8)
```

```python
    ax.set_ylabel(r'Kalman gains', fontsize=8)

    plt.legend()
    if (True == isPlotPdf):
        if not os.path.exists('./generatedPlots'):
            os.makedirs('generatedPlots')
        fig.savefig('./generatedPlots/q4_kal_gain.pdf')
    else:
        plt.show()
################################################################################
# (b) Theoretical estimation standard dev from covariance analysis
################################################################################
# cov_ana_x_0_hat = None
n_steps_steady_state = 1000 # Just to calculate theoretical steady state value
cov_ana_P_0_hat = np.zeros(P_0_hat.shape)
P_k_theoretical = kf.calculate_theoretical_cov(x_0=None, P_0=cov_ana_P_0_hat, num_step=n_steps_steady_state)
################################################################################
# Plot of std dev of estimation error and theoretical value
################################################################################
if (True == isPlotReqd):
    ################################################################################
    # Configure axis and grid
    ################################################################################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    fig.subplots_adjust(left=.15, bottom=.16, right=.99, top=.97)

    ax.set_axisbelow(True)
    ax.minorticks_on()
    ax.grid(which='major', linestyle='-', linewidth='0.5')
    ax.grid(which='minor', linestyle='-.', linewidth='0.5')

    ax.plot(t[0:num_step+1], np.sqrt(P_k[0, 0, :]), color='g', label='Population est s.d.')
    ax.plot(t[0:num_step+1], -np.sqrt(P_k[0, 0, :]), color='g')
    ax.hlines(np.sqrt(P_k_theoretical[0, 0, -1]), t[0], t[num_step],
              color='r', label='Population theoretical ss s.d.')
    ax.hlines(-np.sqrt(P_k_theoretical[0, 0, -1]), t[0], t[num_step], color='r')
    ax.plot(t[0:num_step+1], np.sqrt(P_k[1, 1, :]), color='b', label='food est s.d.')
    ax.plot(t[0:num_step+1], -np.sqrt(P_k[1, 1, :]), color='b')
    ax.hlines(np.sqrt(P_k_theoretical[1, 1, -1]), t[0], t[num_step],
              color='m', label='food theoretical ss s.d.')
    ax.hlines(-np.sqrt(P_k_theoretical[1, 1, -1]), t[0], t[num_step], color='m')

    ax.set_xlabel(r'time', fontsize=8)
    ax.set_ylabel(r'standard deviation', fontsize=8)

    plt.legend()
    if (True == isPlotPdf):
        if not os.path.exists('./generatedPlots'):
            os.makedirs('generatedPlots')
        fig.savefig('./generatedPlots/q4_std_dev_theoretical.pdf')
    else:
        plt.show()
################################################################################
# (c) Theoretical estimation standard dev from covariance analysis
################################################################################
kf.num_step = 1000
kf.run_kalman()
P_k = kf.get_est_error_cov()
P_k_theoretical = kf.calculate_theoretical_cov(x_0=None, P_0=cov_ana_P_0_hat
```

```
# t = np.linspace(0, kf.num_step, num=kf.num_step+1)
#########################################################################
# Plot of std dev of estimation error and theoretical value
#########################################################################
if (True == isPlotReqd):
    #########################################################################
    # Configure axis and grid
    #########################################################################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    fig.subplots_adjust(left=.15, bottom=.16, right=.99, top=.97)

    ax.set_axisbelow(True)
    ax.minorticks_on()
    ax.grid(which='major', linestyle='-', linewidth='0.5')
    ax.grid(which='minor', linestyle='-.', linewidth='0.5')

    ax.plot(t, np.sqrt(P_k[0, 0, :]), color='g', label='Population est s.d.')
    ax.plot(t, -np.sqrt(P_k[0, 0, :]), color='g')
    ax.hlines(np.sqrt(P_k_theoretical[0, 0, -1]), t[0], t[-1],
              color='r', label='Population theoretical ss s.d.')
    ax.hlines(-np.sqrt(P_k_theoretical[0, 0, -1]), t[0], t[-1], color='r')
    ax.plot(t, np.sqrt(P_k[1, 1, :]), color='b', label='food est s.d.')
    ax.plot(t, -np.sqrt(P_k[1, 1, :]), color='b')
    ax.hlines(np.sqrt(P_k_theoretical[1, 1, -1]), t[0], t[-1],
              color='m', label='Food theoretical ss s.d.')
    ax.hlines(-np.sqrt(P_k_theoretical[1, 1, -1]), t[0], t[-1], color='m')

    ax.set_xlabel(r'time', fontsize=8)
    ax.set_ylabel(r'standard deviation', fontsize=8)

    plt.legend()
    if (True == isPlotPdf):
        if not os.path.exists('./generatedPlots'):
            os.makedirs('generatedPlots')
        fig.savefig('./generatedPlots/q4_std_dev_theoretical_1000.pdf')
    else:
        plt.show()
```