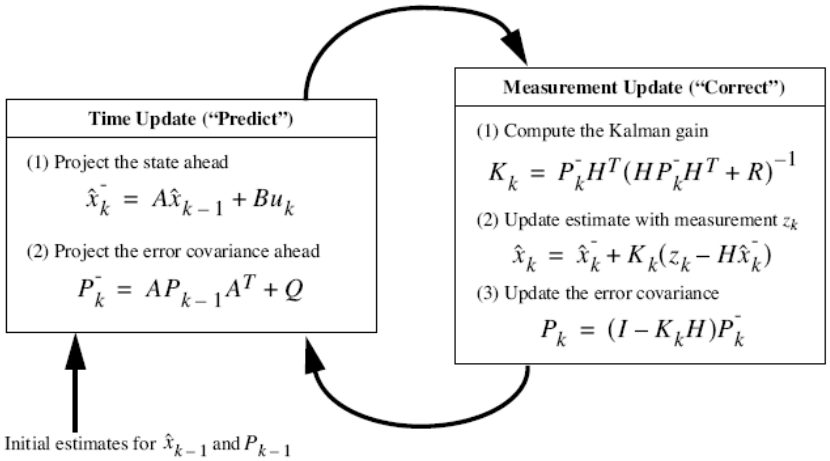
בס"ד

תשובות מטלה 2 – קורס אלגוריתמי ניווט

בפתרון מטלה זו נשתמש בנוסחאות הבאות:



1. יש צורך לתכנן מסנן קלמן עבור מערכת בעלת **שני משתנים** – מיקום ומהירות. המיקום והמהירות הם רק בציר X. מודל המערכת הינו **מהירות קבועה** עם $\Delta t = 1s$. חוסר הדיוק הראשוני במיקום הינו 2 מטרים (מתפלג נורמלאית) וחוסר הדיוק הראשוני במהירות הינו 1.2 מטר לשניה (מתפלג נורמאלית גם). ישנו חיישן המודד את **המיקום** בציר X בלבד (ללא המהירות). החיישן מודד את המיקום ברגל (רגל אחת שווה 0.3048 מטר). דיוק החיישן הינו גאוסי עם סטיית תקן של 0.5 רגל. הניחוש הראשוני של מצב המערכת הינו 8 מטרים למיקום והניחוש הראשוני למהירות הינו 5 מטרים לשנייה. ניתן להניח שמטריצת Q שווה 0.

א. צריך לרשום את מטריצת F, P, H. לכן:

```
F matrix is:
 [[1. 1.]
 [0. 1.]]

P matrix is:
 [[4.   0.  ]
 [0.   1.44]]

State Vector is:
 [[8]
 [5]]
```

ב. צריך לרשום את וקטור המצב החדש ומטריצת P החדשה לאחר שנייה אחת. לכן, ע"פ הנוסחאות הנ"ל:

```
P matrix is:
 [[5.44 1.44]
 [1.44 1.44]]

State Vector is:
 [[13.]
 [ 5.]]
```

ג. מייד לאחר התזוזה, שהחיישן דיווח שהמערכת נמצאת במיקום של 43 רגל, צריך לחשב את וקטור המצב (X ו P) ואת הגבר קלמן החדש לאחר שלב ה-**UPDATE.**
לכן, נקבל:

```
P matrix is:
 [[0.02312702 0.00612186]
 [0.00612186 1.06044402]]

State Vector is:
 [[13.10594766]
 [ 5.02804497]]

New Kalman Gain (K) is:
 [[0.30350421]
 [0.08033935]]
```

2. נניח שהחיישן מדווח גם על מיקום (ברגל-FEET) וגם על מהירות (ביחידות של מטר לשנייה). **סטיית הת**קן של המיקום היא 0.5 רגל **וסטיית התקן** של המהירות הינה 4 מטר\לשניה (שימו לב ששיערוך המהירות הינו גרוע ביותר).

א. צריך לחזור על שאלה 1 סעיף א עם הנתונים החדשים. לכן, נקבל:

```
F matrix is:
 [[1. 1.]
 [0. 1.]]

P matrix is:
 [[5.44 1.44]
 [1.44 1.44]]

H matrix is:
 [[3.2808399 0.        ]
 [0.        1.        ]]
```

ב. בהנחה שהחיישן דיווח שהמערכת נמצאת במיקום של 43 רגל, והמהירות הינה 4 מטר לשנייה, צריך לחשב את וקטור המצב (X וP) ואת הגבר קלמן החדש לאחר שלב ה**UPDATE**. לכן, נקבל:

```
P matrix is:
 [[0.02312482 0.00574134]
 [0.00574134 0.99452888]]

State Vector is:
 [[13.10557877]
 [ 4.96414369]]

New Kalman Gain (K) is:
 [[0.30347538 0.00035883]
 [0.07534561 0.06215806]]
```

בס"ד

3. **צריך לכתוב תוכנית בפיתון שמקבלת את הפרמטרים הבאים:**

המטריצות\וקטורים: P, F, H

ניחוש ראשוני : X

מדידה Z

ומחשבת ומחזירה את וקטור המצב החדש (X) ואת חוסר הוודאות החדש (P).

מימוש אפשרי לתוכנית:

```
import numpy as np
from numpy.linalg import inv

def kalman(h,f,p,x,z,r):
    Q_matrix = 0
    I = np.matrix([[1,0],[0,1]])
    # Calc X
    x = f*x
    # Calc P
    p = f*p*f.T + Q_matrix
    # Update & print
    S = h*p*h.T + r
    S = inv(S)
    K = p*h.T*S
    x = x + K*(z-h*x)
    p = (I - K*h)*p
    print("After Update:")
    print("\nP matrix is:\n", pMatrix)
    print("\nState Vector is:\n", x)
    print("\nNew Kalman Gain (K) is:\n", K)

if __name__ == "__main__":
    # H matrix
    factor = 1./0.3048
    hMatrix = np.matrix([factor,0])
    # F matrix
    deltaT = 1.0
    fMatrix = np.matrix([[1,deltaT],[0,1]])
    # P matrix
    stdPos = 2
    stdVel = 1.2
    pMatrix = np.matrix([[stdPos**2,0],[0,stdVel**2]])
    # X matrix
    initialPos_guess = 8
    initialVelocity_guess = 5
    xMatrix = np.matrix([[initialPos_guess],[initialVelocity_guess]])
    # Z matrix
    zMatrix = np.matrix([43])
    # R matrix
    std_sensor = 0.5
    rMatrix = np.matrix([std_sensor**2])
    kalman(hMatrix,fMatrix,pMatrix,xMatrix,zMatrix,rMatrix)
```

4. צריך להשתמש בקוד הפיתון שכתבנו ולשנות אותו כך שיתאים לבעיה הבאה:

מסנן קלמן המודד מיקום ומהירות בשני צירים $\begin{pmatrix} Px \\ Py \\ Vx \\ Vy \end{pmatrix}$. יש חיישן המודד את רק את המיקום $\begin{pmatrix} Px \\ Py \end{pmatrix}$. סטיית התקן של שגיאת החיישן הינה 6 מטר לכל ציר. בנוסף לוקטור המצב, למערכת יש תאוצה קבועה בציר X ותאוצה קבועה בציר Y - $ax = 5\frac{m}{s^2}$ $ay = 15\frac{m}{s^2}$. חוסר הוודאות הראשוני במיקום הינו 7 מטרים לציר X ו-7 מטרים לציר Y. חוסר הוודאות הראשוני במהירות הינו 100 מטרים לשנייה. ניתן להניח ניחוש ראשוני של מהירות כאפס. הניחוש הראשוני של המערכת הינו x=200, y=150.

החיישן מקבל את המדידות הבאות (זמן הדגימה של המערכת הינו שנייה אחת)

X [m]	Y [m]	t
240	204	1
284	267	2
334	344	3
390	437	4
450	544	5
516	667	6

צריך לחשב מהו וקטור המצב לאחר כל המדידות האלו. לכן, נקבל:

```
State Vector is:
 [[515.30128021]
 [667.82815584]
 [ 67.6560699 ]
 [130.62964398]]

P matrix is:
 [[17.059979    0.          4.06497694  0.        ]
 [ 0.         17.059979    0.          4.06497694]
 [ 4.06497694  0.          1.41055982  0.        ]
 [ 0.          4.06497694  0.          1.41055982]]
```

(מימוש אפשרי לתוכנית המתבקשת בשאלה 4 נמצא בסוף הקובץ).

בס"ד

קוד עבור שאלה 4:

```
import numpy as np
from numpy.linalg import inv

def kalman(h,f,p,x,z,r,b,u):
    Q_matrix = 0
    # Calc X
    x = f*x+b*u
    # Calc P
    p = f*p*f.T + Q_matrix
    # Update
    S = h*p*h.T + r
    S = inv(S)
    K = p*h.T*S
    x = x + K*(z-h*x)
    p = p-K*h*p
    return x,p

if __name__ == "__main__":
    # H matrix
    factor = 1.0
    hM = np.matrix([[factor,0,0,0],[0,factor,0,0]])
    # P matrix
    stdPos = 7
    velocityPos = 100
    pM = np.matrix([[stdPos**2,0,0,0],[0,stdPos**2,0,0],[0,0,velocityPos**2,0],[0,0,0,velocityPos**2]])
    # X matrix
    initialXPos_guess = 200
    initialYPos_guess = 150
    initialVelocity_guess = 0
    xM = np.matrix([[initialXPos_guess],[initialYPos_guess],[initialVelocity_guess],[initialVelocity_guess]])
    # R matrix
    std_sensor = 6
    rM = np.matrix([[std_sensor**2,0],[0,std_sensor**2]])
    # U matrix
    uM = np.matrix([[5],[15]])
    # F matrix
    deltaT = 1
    fM = np.matrix([[1,0,deltaT,0],[0,1,0,deltaT],[0,0,1,0],[0,0,0,1]])
    # B
    bM = np.matrix([[1/2*deltaT**2, 0], # after multiply with U matrix: ax*(1/2)t^2.
                    [0,1/2*deltaT**2], # after multiply with U matrix: ay*(1/2)t^2.
                    [deltaT,0],        # after multiply with U matrix: ax*t.
                    [0,deltaT]])       # after multiply with U matrix: ay*t.
    # set the measurements from the sensor for iteration
    measurements = [[240,204],[284,267],[334,344],[390,437],[450,544],[516,667]]
    for i in range(0,6):
        # Update Z matrix
        zM = np.matrix([[measurements[i][0]],[measurements[i][1]]])
        # Call kalman method
        xM,pM = kalman(hM,fM,pM,xM,zM,rM,bM,uM)
        print("---------------------------------")
        print("Iteration number", i+1)
        print("---------------------------------")
        print("State Vector is:\n", xM)
        print("\nP matrix is:\n", pM)
```