COMP0130: ROBOT VISION AND NAVIGATION

Workshop 2: Aircraft Navigation using GNSS and Kalman Filtering

Paul Groves

Aim

The aim of this workshop is to apply Kalman filtering to GNSS navigation using Matlab. First, a theoretical design problem is presented to help test your understanding.

Introduction

An unmanned air vehicle (UAV) is flying over the Amazon jungle in Brazil, conducting an aerial survey to detect unauthorised logging. GNSS pseudo-range and pseudo-range rate measurements are logged every second. Your task is to compute a continuous GNSS navigation solution using Kalman filtering. Step by step instructions, including all equations to implement, are provided. Answers will appear on Moodle at the end of the workshop.

Data Files Supplied

You have been provided on Moodle with three data files, all in comma-separated variable (CSV) format:

- In the file Workshop2_Pseudo_ranges.csv, the first column contains the time in seconds and the first row contains the satellite numbers. The remaining rows and columns contain the corresponding pseudo-range measurements in metres.
- In the file Workshop2_Pseudo_range_rates.csv, the first column contains the time in seconds and the first row contains the satellite numbers. The remaining rows and columns contain the corresponding pseudo-range rate measurements in metres per second.
- In the file Workshop2_GNSS_Pos_ECEF.csv, the first column contains the time in seconds and the remaining three columns contain the Cartesian Earth-centred Earth-fixed (ECEF) position solution computed using the pseudo-range measurements in the first file.

Task 0: Theoretical Design Problem

A Kalman filter estimates six states using a local-tangent-plane coordinate system. These are north position, east position, north velocity, east velocity, north acceleration, and east acceleration.

a) Determine the exact transition matrix as a function of the state propagation interval, τ_s.

b) Use this to determine the position error that arises from ignoring the acceleration if the propagation interval is 0.5s and the horizontal acceleration is 4 m s^{-2} north.

c) If the measurement vector comprises north position, east position, north velocity and east velocity, what is the measurement matrix?

Task 1A: Basic Kalman Filter First Epoch

For task 1, you will take the simple 2-state position and velocity Kalman filter from Lecture 4 and extend it to three dimensions. The measurements input to the Kalman filter will comprise the least-squares GNSS position solution at each epoch.

For task 1A, compute the position and velocity at time 0 using the first set of measurements in Workshop2_GNSS_Pos_ECEF.csv by following the steps below.

For task 1, you will implement a 6-state Kalman filter estimating Cartesian ECEF position (m) and velocity (m/s).The state vector is thus

$$\mathbf{x} = \begin{pmatrix} \mathbf{r}_{ea}^{e} \\ \mathbf{v}_{ea}^{e} \end{pmatrix} = \begin{pmatrix} x_{ea}^{e} \\ y_{ea}^{e} \\ z_{ea}^{e} \\ v_{ea,x}^{e} \\ v_{ea,y}^{e} \\ v_{ea,z}^{e} \end{pmatrix} \tag{1}$$

a) Initialise the Kalman filter state vector estimate using an estimated position of (2447019, -5884199, -284783) m and an estimated velocity of (184, 77, 0) m/s. Initialise the error covariance to give initial state uncertainties of 10m for each component of position and 5 m/s for each component of velocity. Thus,

$$\hat{\mathbf{x}}_{0}^{+} = \begin{pmatrix} 2447019 \\ -5884199 \\ -284783 \\ 184 \\ 77 \\ 0 \end{pmatrix} \quad \mathbf{P}_{0}^{+} = \begin{pmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 25 \end{pmatrix} \tag{2}$$

b) (Kalman filter Step 1) Compute the transition matrix using

$$\mathbf{\Phi}_{k-1} = \begin{pmatrix} \mathbf{I}_3 & \tau_s \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{pmatrix}, \tag{3}$$

where the propagation interval, τ_s, is 1s in this case.

c) (Step 2) Compute the system noise covariance matrix using

$$\mathbf{Q}_{k-1} = \begin{pmatrix} \frac{1}{3} S_a \tau_s^3 \mathbf{I}_3 & \frac{1}{2} S_a \tau_s^2 \mathbf{I}_3 \\ \frac{1}{2} S_a \tau_s^2 \mathbf{I}_3 & S_a \tau_s \mathbf{I}_3 \end{pmatrix} \tag{4}$$

where the acceleration power spectral density (PSD), S_a^e, is 5 m^2s^{-3}.

d) (Step 3) Use the transition matrix to propagate the state estimates:

$$\hat{\mathbf{x}}_{k}^{-} = \mathbf{\Phi}_{k-1} \hat{\mathbf{x}}_{k-1}^{+}. \tag{5}$$

e) (Step 4) Then use this to propagate the error covariance matrix:

$$\mathbf{P}_{k}^{-} = \mathbf{\Phi}_{k-1} \mathbf{P}_{k-1}^{+} \mathbf{\Phi}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1} \tag{6}$$

f) (Step 5) Formulate the measurement matrix:

$$\mathbf{H}_{k} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{7}$$

g) (Step 6) Compute the measurement noise covariance matrix assuming each component of the position measurement has an error standard deviation of σ_x = 2.5 m Thus,

$$\mathbf{R}_k = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_x^2 & 0 \\ 0 & 0 & \sigma_x^2 \end{pmatrix} = \begin{pmatrix} 6.25 & 0 & 0 \\ 0 & 6.25 & 0 \\ 0 & 0 & 6.25 \end{pmatrix}. \tag{8}$$

h) (Step 7) Compute the Kalman gain matrix using

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} + \mathbf{R}_k \right)^{-1} \tag{9}$$

i) (Step 8) Formulate the measurement innovation vector, $\delta\mathbf{z}_k^-$, using

$$\delta\mathbf{z}^- = \tilde{\mathbf{r}}_{ea,k}^e - \hat{\mathbf{r}}_{ea,k}^{e-} = \begin{pmatrix} \tilde{x}_{ea,k}^e - \hat{x}_{ea,k}^{e-} \\ \tilde{y}_{ea,k}^e - \hat{y}_{ea,k}^{e-} \\ \tilde{z}_{ea,k}^e - \hat{z}_{ea,k}^{e-} \end{pmatrix} \tag{10}$$

where $\tilde{\mathbf{r}}_{ea,k}^e$ is the GNSS least-squares position solution and $\hat{\mathbf{r}}_{ea,k}^{e-}$ is the position predicted by the Kalman filter, i.e. the first 3 components of the propagated state vector, $\hat{\mathbf{x}}_k^-$.

j) (Step 9) Update the state estimates using

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \delta\mathbf{z}_k^- \tag{11}$$

k) (Step 10) Update the error covariance matrix using

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^- \quad . \tag{12}$$

l) Convert this Cartesian ECEF position solution to latitude, longitude and height using the Matlab function pv_ECEF_to_NED.m (supplied on Moodle). Note that the Matlab function outputs latitude and longitude in radians. Use the same Matlab function to convert the velocity solution from ECEF resolving axes to north, east and down.

Task 1B: Basic Kalman Filter Multiple Epochs

Now use the same method to compute the filtered position and velocity at all of the epochs in Workshop2_GNSS_Pos_ECEF.csv. Remember that $\hat{\mathbf{x}}_{k-1}^+$ and $\mathbf{P}_{k-1}^+$ for the current epoch are the same as $\hat{\mathbf{x}}_k^+$ and $\mathbf{P}_k^+$ for the previous epoch.

Task 1C: Basic Kalman Filter State Uncertainties (OPTIONAL)

By taking the square roots of the 1st and 4th diagonal elements of the error covariance matrix, $\mathbf{P}_k^+$, examine how the uncertainty of the *x*-direction position and velocity states vary over time. Note that they converge to constant values after a few seconds. Repeat this exercise with the measurement update (steps 5 to 10) disabled and observe how the uncertainties increase over time in the absence of new measurement information.

Task 2A: GNSS Kalman Filter First Epoch

We now move on to a proper GNSS Kalman filter that determines the position and velocity directly from the pseudo-range and pseudo-range rate measurements. For task 2A, compute the position and velocity at time 0 using the first set of measurements in Workshop2_Pseudo_ranges.csv and Workshop2_Pseudo_range_rates.csv by following the steps below.

For task 2 you need to extend the Kalman filter from task 1 to 8 states by adding clock offset (m) and clock drift (m/s). The state vector is thus

$$\mathbf{x} = \begin{pmatrix} \mathbf{r}_{ea}^{e} \\ \mathbf{v}_{ea}^{e} \\ \delta\rho_{c}^{a} \\ \delta\dot{\rho}_{c}^{a} \end{pmatrix} \tag{13}$$

a) Initialise the Kalman filter state vector estimate and error covariance matrix using the Matlab function Initialise_GNSS_KF.m (supplied on Moodle). Note that the initial state uncertainties are 10m for position and clock offset and 0.1 m/s for velocity and clock drift.

b) (Kalman filter Step 1) Compute the transition matrix using

$$\mathbf{\Phi}_{k-1} = \begin{pmatrix} \mathbf{I}_3 & \tau_s \mathbf{I}_3 & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} \\ \mathbf{0}_{1,3} & \mathbf{0}_{1,3} & 1 & \tau_s \\ \mathbf{0}_{1,3} & \mathbf{0}_{1,3} & 0 & 1 \end{pmatrix}, \tag{14}$$

where the propagation interval, τ_s, is 1s in this case.

c) (Step 2) Compute the system noise covariance matrix using

$$\mathbf{Q}_{k-1} = \begin{pmatrix} \frac{1}{3} S_a \tau_s^3 \mathbf{I}_3 & \frac{1}{2} S_a \tau_s^2 \mathbf{I}_3 & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} \\ \frac{1}{2} S_a \tau_s^2 \mathbf{I}_3 & S_a \tau_s \mathbf{I}_3 & \mathbf{0}_{3,1} & \mathbf{0}_{3,1} \\ \mathbf{0}_{1,3} & \mathbf{0}_{1,3} & S_{c\phi}^a \tau_s + \frac{1}{3} S_{cf}^a \tau_s^3 & \frac{1}{2} S_{cf}^a \tau_s^2 \\ \mathbf{0}_{1,3} & \mathbf{0}_{1,3} & \frac{1}{2} S_{cf}^a \tau_s^2 & S_{cf}^a \tau_s \end{pmatrix} \tag{15}$$

where the acceleration power spectral density (PSD), S_a^e, is 5 m^{2}s^{-3}, the clock phase PSD, $S_{c\phi}^a$, is 0.01 m^{2}s^{-3} and the clock frequency PSD, S_{cf}^a, is 0.04 m^{2}s^{-1}.

d) (Step 3) Use the transition matrix to propagate the state estimates:

$$\hat{\mathbf{x}}_k^- = \mathbf{\Phi}_{k-1} \hat{\mathbf{x}}_{k-1}^+ . \tag{16}$$

e) (Step 4) Then use this to propagate the error covariance matrix:

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{\Phi}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1} \tag{17}$$

f) Predict the ranges from the approximate user position to each satellite using

$$\hat{r}_{aj}^- = \sqrt{\left[\mathbf{C}_e^I\left(\hat{r}_{aj}^-\right)\hat{\mathbf{r}}_{ej}^e - \hat{\mathbf{r}}_{ea}^{e-}\right]^{\mathrm{T}}\left[\mathbf{C}_e^I\left(\hat{r}_{aj}^-\right)\hat{\mathbf{r}}_{ej}^e - \hat{\mathbf{r}}_{ea}^{e-}\right]} \tag{18}$$

where $\hat{\mathbf{r}}_{ej}^e$ is the Cartesian ECEF position of satellite j, $\hat{\mathbf{r}}_{ea}^{e-}$ is the predicted Cartesian ECEF user position and $\mathbf{C}_e^I$ is the Sagnac effect compensation matrix, given by

$$\mathbf{C}_e^I(r_{aj}) \approx \begin{pmatrix} 1 & \omega_{ie} r_{aj}/c & 0 \\ -\omega_{ie} r_{aj}/c & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{19}$$

where the Earth rotation rate, ω_{ie}, is 7.292115×10^{-5} rad s^{-1} and the speed of light, c, is 299792458 m s^{-1}. The recursion is resolved by initially computing the range with the Sagnac effect compensation matrix set to the identity matrix, then using this range to compute the Sagnac effect compensation matrix and then recomputing the range.

g) Compute the Cartesian ECEF positions of the satellites at time 0 using the Matlab function Satellite_position_and_velocity.m (supplied on Moodle). Note that this function needs the satellite numbers given in the first row of the pseudo-ranges file. Then, Compute the line-of-sight unit vector from the approximate user position to each satellite using

$$\mathbf{u}_{aj}^{e} = \frac{\mathbf{C}_e^I\left(\hat{r}_{aj}^-\right)\hat{\mathbf{r}}_{ej}^e - \hat{\mathbf{r}}_{ea}^{e-}}{\hat{r}_{aj}^-}. \tag{20}$$

You can also use the following approximation. However, it will give you slightly different results from those in the answer sheet:

$$\mathbf{u}_{aj}^{e} \approx \frac{\hat{\mathbf{r}}_{ej}^e - \hat{\mathbf{r}}_{ea}^{e-}}{\hat{r}_{aj}^-}. \tag{21}$$

h) Predict the range rates from the approximate user position to each satellite using

$$\hat{\dot{r}}_{aj}^- = \hat{\mathbf{u}}_{aj}^{e-\mathrm{T}}\left[\mathbf{C}_e^I\left(\hat{r}_{aj}^-\right)\left(\hat{\mathbf{v}}_{ej}^e + \boldsymbol{\Omega}_{ie}^e\hat{\mathbf{r}}_{ej}^e\right) - \left(\hat{\mathbf{v}}_{ea}^{e-} + \boldsymbol{\Omega}_{ie}^e\hat{\mathbf{r}}_{ea,k}^{e-}\right)\right] \tag{22}$$

where $\hat{\mathbf{v}}_{ej}^e$ is the Cartesian ECEF velocity of satellite j, $\hat{\mathbf{v}}_{ea}^{e-}$ is the predicted Cartesian ECEF user velocity and $\boldsymbol{\Omega}_{ie}^e$ is the skew symmetric matrix of the Earth rotation rate vector. It is calculated for you in Define_constants.m.

i) (Step 5) Compute the measurement matrix using

$$\mathbf{H}_k = \begin{pmatrix} -u_{a4,x}^e & -u_{a4,y}^e & -u_{a4,z}^e & 0 & 0 & 0 & 1 & 0 \\ -u_{a5,x}^e & -u_{a5,y}^e & -u_{a5,z}^e & 0 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -u_{a30,x}^e & -u_{a30,y}^e & -u_{a30,z}^e & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -u_{a4,x}^e & -u_{a4,y}^e & -u_{a4,z}^e & 0 & 1 \\ 0 & 0 & 0 & -u_{a5,x}^e & -u_{a5,y}^e & -u_{a5,z}^e & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -u_{a30,x}^e & -u_{a30,y}^e & -u_{a30,z}^e & 0 & 1 \end{pmatrix} \tag{23}$$

j) (Step 6) Compute the measurement noise covariance matrix assuming all pseudo-range measurements have an error standard deviation of 10m and all pseudo-range rate measurements have an error standard deviation of 0.05 m/s. Thus,

$$\mathbf{R}_k = \begin{pmatrix} \sigma_\rho^2 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_\rho^2 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_\rho^2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \sigma_r^2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \sigma_r^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & \sigma_r^2 \end{pmatrix} \tag{24}$$

where σ_ρ = 10 m and σ_r = 0.05 m/s.

k) (Step 7) Compute the Kalman gain matrix using

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} + \mathbf{R}_k \right)^{-1} \tag{25}$$

l) (Step 8) Formulate the measurement innovation vector, $\delta\mathbf{z}_k^-$, using

$$\delta\mathbf{z}^- = \begin{pmatrix} \tilde{\rho}_a^4 - \hat{r}_{a4}^- - \delta\hat{\rho}_c^{a-} \\ \tilde{\rho}_a^5 - \hat{r}_{a5}^- - \delta\hat{\rho}_c^{a-} \\ \vdots \\ \tilde{\rho}_a^{30} - \hat{r}_{a30}^- - \delta\hat{\rho}_c^{a-} \\ \tilde{\dot{\rho}}_a^4 - \hat{\dot{r}}_{a4}^- - \delta\hat{\dot{\rho}}_c^{a-} \\ \tilde{\dot{\rho}}_a^5 - \hat{\dot{r}}_{a5}^- - \delta\hat{\dot{\rho}}_c^{a-} \\ \vdots \\ \tilde{\dot{\rho}}_a^{30} - \hat{\dot{r}}_{a30}^- - \delta\hat{\dot{\rho}}_c^{a-} \end{pmatrix} \tag{26}$$

where $\tilde{\rho}_a^j$ is the measured pseudo-range from satellite *j* to the user antenna, $\tilde{\dot{\rho}}_a^j$ is the measured pseudo-range rate from satellite *j* to the user antenna, $\delta\hat{\rho}_c^{a-}$ is the propagated receiver clock offset estimate, and $\delta\hat{\dot{\rho}}_c^{a-}$ is the propagated receiver clock drift estimate.

m) (Step 9) Update the state estimates using

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\delta\mathbf{z}_k^- \tag{27}$$

n) (Step 10) Update the error covariance matrix using

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\,\mathbf{P}_k^- \;. \tag{28}$$

o) Convert this Cartesian ECEF position solution to latitude, longitude and height using the Matlab function pv_ECEF_to_NED.m (supplied on Moodle). Note that the Matlab function outputs latitude and longitude in radians. Use the same Matlab function to convert the velocity solution from ECEF resolving axes to north, east and down.

Task 2B: GNSS Kalman Filter Multiple Epochs

Now use the same method to compute the filtered position and velocity at all of the epochs in Workshop2_Pseudo_ranges.csv and Workshop2_Pseudo_range_rates.csv. Remember that $\hat{\mathbf{x}}_{k-1}^+$ and $\mathbf{P}_{k-1}^+$ for the current epoch are the same as $\hat{\mathbf{x}}_k^+$ and $\mathbf{P}_k^+$ for the previous epoch.