INTERNATION UNIVERSITY LIAISON INDONESIA (IULI)

BACHELOR'S THESIS

AIRCRAFT FLIGHT PATH RECONSTRUCTION BASED ON ADS-B DATA USING KALMAN FILTER

By

Yazfan Tabah Tahta Bagaskara

11201501006

Presented to the Faculty of Engineering

In Partial Fulfilment Of the Requirements for the Degree of

SARJANA TEKNIK

In

AVIATION ENGINEERING

FACULTY OF ENGINEERING

BSD City 15345

Indonesia

September 2020

APPROVAL PAGE

AIRCRAFT FLIGHT PATH RECONSTRUCTION BASED ON ADS-B DATA USING KALMAN FILTER

YAZFAN TABAH TAHTA BAGASKARA
11201501006
Presented to the Faculty of Engineering
In Partial Fulfillment of the Requirements for the Degree of

SARJANA TEKNIK
In
AVIATION ENGINEERING

FACULTY OF ENGINEERING

Triwanto Simanjuntak, PhD _____
Thesis Advisor Date

Dr. Ir. Prianggada Indra Tanaya, M.M.E _____
Dean of Faculty of Engineering Date

STATEMENT BY THE AUTHOR

I hereby declare that this submission is my own work and to the best of my knowledge, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at any educational institution, except where due acknowledgement is made in the thesis.

Yazfan Tabah Tahta Bagaskara _____ _____

Student Date

ABSTRACT

Aircraft Flight Path Reconstruction Based on ADS-B Data Using Kalman Filter

by

Yazfan Tabah Tahta Bagaskara

Triwanto Simanjuntak, PhD, Advisor

In this thesis, the Kalman filter algorithm was used to perform flight path reconstruction. The flight path reconstruction was based on the ADS-B data set that has been obtained from opensky-network.org. Other methods, such as interpolation calculation, also can be used to perform the flight path reconstruction. However, in using interpolation methods, the ADS-B data inaccuracy is not taken into account. Since the position information in the ADS-B data has an error that are indicated by NIC (Navigation Integrity Code). So, the result of the flight path reconstruction based on interpolation calculations are theoretically inaccurate. There were 3 interpolation methods that were explored in this research; they were linear interpolation, spline interpolation, and PCHIP interpolation. These methods were used as preliminary tools to construct the flight path and to see how they might diverge from the Kalman Filter's results. To minimize the ADS-B data inaccuracy, this research used the Kalman filter method; in this research, the Kalman filter calculation was based only on the kinematic equation. The Kalman filter results were compared with the original data in order to observe how the inaccuracy could deteriorate the aircraft trajectory.

Keyword: *ADS-B, Interpolation, Kalman Filter, Flight Path, Navigation Integrity Code (NIC)*

ACKNOWLEDGEMENTS

Thank you God Almighty, for the strength, knowledge, ability, and opportunity to undertake and complete this research study. Without his presence, this wonderful achievement would not be possible.

Throughout the writing process of this thesis report, I have received a great amount of supports and assistance. I would like to thank my thesis supervisor and advisor, Triwanto Simanjuntak, Ph.D.

I would also like to thank my parents for the wise counsel and support.

Special thanks to I Gede Suryadharma Susila, Ebro Haninditho, and all my family members of aviation engineering batch 2015 for all the help and support in the making of this thesis.

And finally, for everyone that I have not mentioned yet.

Thank you for all the encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
ADSB	**A**utomatic **D**ependent **S**urveillance **B**roadcast
FPR	**F**light **P**ath **R**econstruction
ATC	**A**ir **T**raffic **C**ontroller
FIR	**F**light **I**nformation **R**egion
FISB	**F**light **I**nformation **S**ervice **B**roadcast
TISB	**T**raffic **I**nformation **S**ervice **B**roadcast
UAT	**U**niversa **A**ccess **T**ransceiver
NIC	**N**avigation**I**ntegrity **C**ode
NACp	**N**avigation**A**curacy **C**ode **P**osition
NUC	**N**avigation**U**Uncertainty **C**ode
SIL	**S**urveillance**I**n- tegrity **L**evel
MSE	**M**ean**S**quare **E**rror

Dedicated to my parents

CHAPTER 1
INTRODUCTION

1.1 Background

By definition, flight path is a set of corridors that connects one location to another location. In general, we can define the flight path as a route that the aircraft took to get from one point to another point. The corridors that the aircraft used is made by a set of geographical coordinates. These coordinates are usually based on satellite navigational systems or other ground-based radio transmitters navigational system. To monitor the aircraft during flight, we can track it by using a method called flight tracking. Flight tracking is a method of aircraft monitoring used by surveillance users such as an ATC (Air Traffic Controller) to monitor the aircraft during flight by using a surveillance system. The tracking of a flight can be done in real-time or reconstruct by using historical data sets that contain the aircraft condition(i.e., latitude and longitude coordinates). There are several systems that can be used in flight tracking, such as using a radar-based monitoring technique, which uses a principle of a radar radio wave to determine the distance between the aircraft and the radar source to obtain the position of the aircraft. Then there is a satellite-based tracking system that uses GNSS (Global Navigation Satellite System) to determine the aircraft's exact coordinate, which is called ADS-B. The ADS-B (Automatic Dependent Surveillance-Broadcast) satellite-based tracking system is integrated into surveillance technology that monitors aircraft conditions and trajectories. The ADS-B system can track the aircraft in real-time and record all the information that the system received. The recorded data that the ADS-B receives can be used as source data in FPR (Flight Path Reconstruction). FPR results can be used to test the reliability of the data and to check post-flight data. FPR utilizes the aircraft sensors data to determine the position, velocity, and altitude of an aircraft to perform those tasks. In reality, the sensor only consists of

the estimation of the said information. A kinematic model of the aircraft is needed to obtain more accurate information on the aircraft state. The idea was to combine the kinematic model with the complete flight data to recreate the state at an exact timestamp. The reconstruct data then can be used to analyze system identification, dynamic analysis, or control algorithm assessment(Göttlicher & Holzapfel, 2016). By knowing the accurate state of the aircraft at a certain point in time, the data can be used as the tools to help the investigation of an aircraft accident or incident.

1.2 Problem Statement

All the systems used for aircraft tracking have a time when the system needs to update the aircraft information. If we view these exact timestamps, we will see information unavailability. This period is called intermission/interruption periods. For example, in a radar-based tracking system, the intermission period can be caused by the latency between the time radio wave is sent to detect the aircraft and when the radio wave is received after it bounces back after detecting an aircraft. For the ADS-B system, the intermission periods were caused by the time the system needs to update the aircraft information, which occurs because the ADS-B did not continuously update the aircraft information. However, there is a gap between each data transmission. In optimal condition, this period lasts for half a second ($0.5s$). In conclusion, during the intermission periods, every data set collected by a different system has a point where there is data have unavailability. The ADS-B also has inaccuracy when the system is determining the aircraft position. The ADS-B commonly have a quality indicator value that was informing the level of the data accuracy. The quality indicator value can be indicated by several types, which were described in a regulation standard DO-260/A/B. In DO-260/A/B, the quality indicator can be shown as NUC (Navigation Uncertainty Code) or NACp (Navigation Accuracy Code for Position) or NIC (Navigation Integrity Code) or SIL (Surveillance Integrity Level). The value of one of these codes will be describing the accuracy of the ADS-B data. This research used the NIC value, the quality indicator code indicating the accuracy radius of the ADS-B data. The accuracy radius is used for determining the vertical and horizontal position accuracy of the aircraft.

1.3 Research Purpose

To overcome the unavailability during the intermission period and inaccuracy of the aircraft position, a flight path estimation need to be generated. This problem can be resolve by using flight path reconstruction methods or FPR. Flight path reconstruction is a method that uses a historical data set of an aircraft condition to reconstruct the past aircraft trajectory. Flightpath reconstruction can be done by using a mathematical computation.

The result of the flight path reconstruction was expected:

- To be used as a tool in aircraft incident/accident.
- To be used as a post-flight analysis data based on the flight trajectory
- To able to producing an estimated aircraft trajectory that had minimal noise or inaccuracy

1.4 Research Scope

This research only used a historical data set that has been gathered by an ADS-B system. This historical data set was obtained from opensky-network.org. The downloaded data set is in the form of a CSV (Comma-Separated Values) file. This research used the data set from 03 March 2020. Due to the format of the opensky-network.org, the data was separated hourly; because of that and hardware limitation, this research used only the first 10 hours of 03 March 2020 data set, which have a total around 3.097 Gb of data. The reason that the opensky-network.org data set is chosen because the data set offers almost all data parameters required in the calculation: timestamp, position (Latitude, Longitude, and altitude), velocity, vertical rate, and magnetic heading. However, the data set that has been obtained from opensky-network.org did not have a quality indicator value. So, to overcome this, the quality indicator is generated randomly using Python programming language. This research utilized python programming language as a tool for performing the FPR. The FPR that was conducted in this research was based on two methods, which are interpolation calculation and Kalman filter calculation.

1.5 Research Approach

The data analysis for this research is conducted by comparing the result from Kalman filter calculation and interpolation calculation with actual ADS-B data to see if there is a deviation between the original aircraft trajectory and aircraft trajectory based on interpolation and Kalman filter calculation. The python programming language is used as a tool for completing the data analysis. The data comparison is based on the MSE(Mean Square Error) calculation. The MSE calculation was intended to show the data difference between the estimated data (Result of interpolation and Kalman filter calculation) and actual data.

The steps to perform this research are:

1. Data set acquisition
2. Performing data filtration
3. Performing data adjustment so that the data parameters are compatible with one another
4. Performing the interpolation and Kalman filter calculation
5. Analyzing the result of the Kalman filter calculation and interpolation calculation

CHAPTER 2
LITERATURE REVIEW

In general, the ADS-B system can be divided into two categories. They are ADS-B Out and ADS-B In. The ADS-B out is acted as the system's broadcaster, and the ADS-B In is acted as the receiver. As the receiver, the ADS-B In is only receiving the data parameters that the ADS-B out sends, including aircraft velocity, aircraft identification, surface position, airborne position, TCAS, emergency status, and operational status. In the ADS-B system, there is a periodical interval between each data cycle (broadcast and receive). This means that during the interval, an estimated aircraft position between interruption can be made by considering this situation as a regression problem with using the known aircraft condition when before and after the intermission period.

In order to generate those flight paths, a method of flight path generation needs to be chosen. In this research, there are two methods that were used, which were Kalman filter and interpolation. Kalman filter is an algorithm that generates an estimate of previously unknown variables that used a combination of estimation generated by a calculation and observation of measurement over a period of time (*Introduction to Kalman Filter and Its Applications | IntechOpen*, 2018). In this research, the Kalman filter calculation was done by using Python programming language. The Kalman filter calculation is based on the kinematic calculation of the aircraft trajectory. The other method used in this research is interpolation; this is a method of generating an estimation of a previously unknown variable using a known data point from a historical data set(*Types of Interpolation - Advantages and Disadvantages*, 2019). In this research, these can be achieved by using two know data points to generate the aircraft's position during the interruption periods. Both the Kalman filter and the interpolation method had its advantages and disadvantages. The first one is that the interpolation methods have more straightforward calculations because this method is generally only a form of mathematical

approximation calculation. On the other hand, the Kalman filter methods calculation is more complex in comparison with the interpolation calculation, because in this calculation, many factors that did not consider in the interpolation are considered here. Consequently, the interpolation result did not minimize the noise of the system contrarily; one of the main ideas of the Kalman filter calculation is to minimize the noise of the system.

2.1 Flight Tracking

Flight tracking is a method of aircraft monitoring that involves a system used to observe aircraft during flight or on the ground. Flight tracking is essential for ATC (Air Traffic Controller) to managing airspace. The tracking system will help the ATC control aircraft separation, control aircraft flight path, and guide aircraft movement or other navigational needs of the aircraft. The ATC usually used a radar system as a tool to track and controlling aircraft movement. Radar or radio detection and ranging that used in UHF (Ultra High Frequency) or microwave part of the RF (Radio Frequency) spectrum to detect an object position (*What is radar (radio detection and ranging)? - Definition from WhatIs.com*, 2005). In general, Radar work by sending an electromagnetic signal through its transmitter to detect an object. When the signal hits an object, the signal will be bounced back. Then the reflected signal will be detected by the radar receiver, which is then processed to determine the object's geographical position. The process used the time taken by the signal to travel from the Radar source to the object and return back to determining the distance between the Radar and the object. For determining the target's location is measured in angle, from the direction of the maximum amplitude echo signal to the antenna(*RADAR - Introduction of RADAR Systems, Types and Applications*, 2013). Another system that can be used to tracking aircraft is ADS-B or Automatic Dependent Surveillance-Broadcast. The ADS-B is a satellite-based monitoring system that uses GNSS (Global Navigation Satellite System) to determining aircraft position. In recent years, ADS-B has been chosen to be the replacement for Radar as the tracking or monitoring system for aircraft. The reason mainly because ADS-B is considered to be more reliable and accurate than Radar.

2.2 Automatic Dependent Surveillance Broadcast

Automatic Dependent Surveillance-Broadcast or ADS-B is a type of monitoring/navigation system used in air transportation. The system's primary purpose is to track aircraft movement that periodically updated every 0.5 seconds in optimal condition, or in other words, the most optimal interruption period is 0.5 seconds (*Implementasi Automatic Dependent Surveillance Broadcast (ADS-B) di Indonesia | Nurhayati | WARTA ARDHIA*, 2014). The system can determine the aircraft's location by using GNSS (Global Navigation Satellite System) such as GPS (Global Positioning System). The system also sends other information that correlates with the aircraft's state, such as aircraft identification number (hex identification number), altitude, heading, and speed. This aircraft information will be transmitted to other aircraft that are also equipped with the ADS-B system and satellite or ground receiver, in which the data will be relayed to ATC (Air Traffic Control).

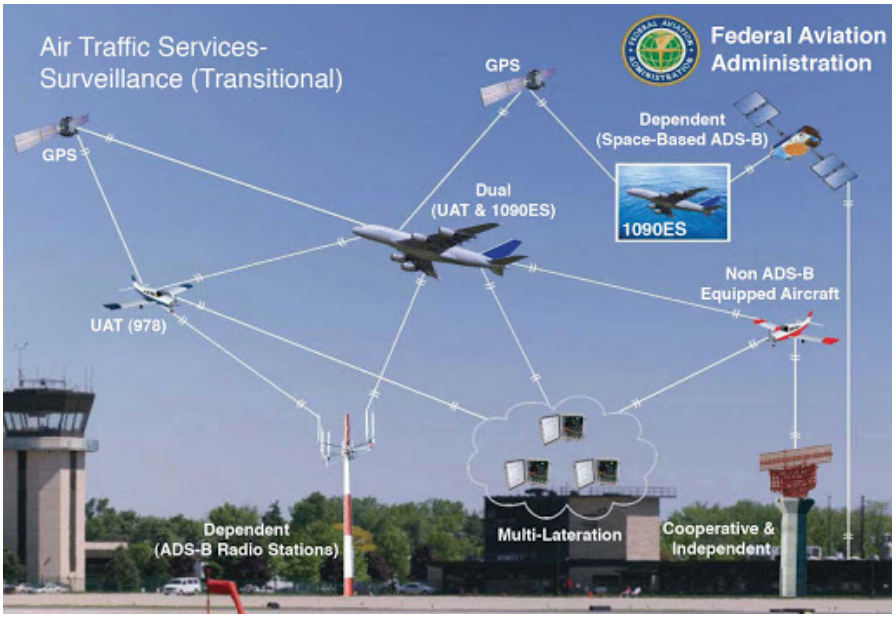

FIGURE 2.1: Schematic of ADS-B system (Calderwood, 2016)

Ideally, the ADS-B is an accurate system that can track an aircraft's trajectory and store that information every 0.5 seconds. However, in real-world implementation, the system cannot perform this perfectly. The system has noise or inaccuracy that can affect the accuracy of the system. Then the system may have a "signal

disturbance" that may affect the data transmission, which can prolong the interruption period. The noise/inaccuracy and prolong interruption time of the ADS-B system may vary between each time step. The magnitude of the noise was identified in the ADS-B message in the form of a quality indicator code. The quality indicator can be defined into several forms of codes; it depends on which version of the DO-260 standard the ADS-B system used. The variation of the quality indicator are:

- NUC = Navigation Uncertainty Code or
- NIC = Navigation Integrity Code
- SIL = Surveillance Integrity Level

Note: In this research, the only quality indicator variant that used is the NIC version. The reason for this is because it is offering much higher error detection, which is up to 30 km, and according to the FAA AC 20-165 mentioned that, NIC is used by surveillance users such as ATC to determine if the data has an acceptable level of integrity. The minimal level requirement of the integrity level is NIC 8. The NIC is also chosen.

There are several factors that can affect the NIC level ("PAPRUsersGuide", 2020):

- Loss of GNSS services
- Antenna masking that caused by aircraft maneuver
- aircraft flying at the edge of ADS-B coverage
- Component/software issue
- Poor ADS-B signal

The NIC is divided into 12 codes, ranging from code 0 to code 11. Each code number is representing a different range of horizontal position accuracy. The following table 2.1 is the description of accuracy for code("PAPRUsersGuide", 2020).

For the system's vertical accuracy, the accuracy magnitude only available for NIC values greater than 8; in other words, the accuracy magnitude available in NIC levels 9 to 11. The magnitude of the vertical NIC can be seen on table 2.2.

NIC Value	Accuracy Value
0	Unknown
1	<37.04 Km
2	<14.81 Km
3	<7.40 Km
4	<3.70 Km
5	<1852 m
6	<926 m
7	<370.4 m
8	<185.2 m
9	<75 m
10	<25 m
11	<7.5 m

TABLE 2.1: NIC Value Horizontal Accuracy Radius

NIC Value	Accuracy Value
9	<112 m
10	<37.5 m
11	<11 m

TABLE 2.2: NIC Value Veritcal Accuracy Radius

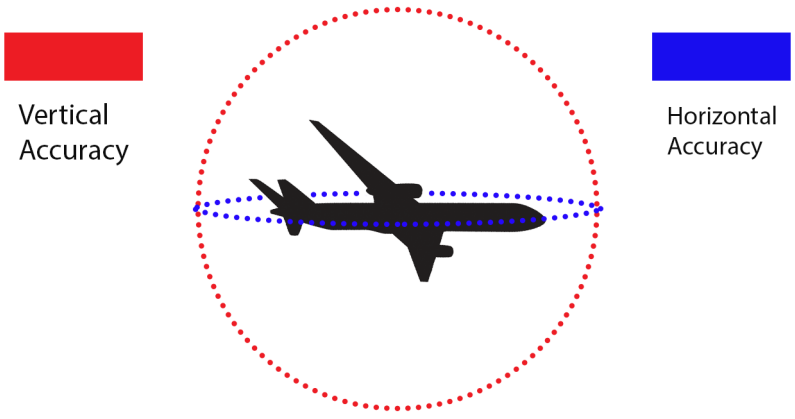

FIGURE 2.2: NIC Horizontal and Vertical Accuracy

2.2.1 Transmission

ADS-B system used mode-s transponder to transmitting/updating aircraft movement that usually operates in 1090 MHz (*Ins and Outs*, 2020). In the US, the FAA (Federal Aviation Administration) want to establish for aircraft that operate below 18,000 feet (5,500 m) transmit the signal 987 MHz. The received ADS-B data can be used alongside the TIS-B (Traffic Information Services Broadcast) and FIS-B (Flight Information Services Broadcast) system. TIS-B is a component of the ADS-B system that provides traffic information for aircraft with ADS-B receiver. TIS-B also has the capability to tracking other aircraft that are not equipped with ADS-B transponder but tracked by other Radar. FIS-B, on the other hand, is a system that provides current airspace information such as graphical weather service and locations of restricted airspace. However, FIS-B information only can be received by aircraft that use 978 MHz UAT (Universal Access Transceiver) (*How Does FIS-B (Flight Information System Broadcast) Work?*, 2019). This process of transmitting and receiving ADS-B information is essentially based on two separate ADS-B systems: ADS-B In and ADS-B Out. ADS-B Out is the part of the ADS-B system that broadcasts aircraft information periodically, such as the aircraft speed, altitude, and location through an onboard transmitter. For the ADS-B In, the system received the other information that will be used for TIS-B and FIS-B system.

2.2.2 ADS-B Implementation

In the time of writing, FAA and EASA regulatory bodies have a mandate that required aircraft to have an ADS-B system installed. Both regulatory bodies want to promote the use of the ADS-B system in other international airspaces, including Asia/Pacific, so they mandate this new regulation. This regulation aimed to obtain the great benefit that the ADS-B system is offering, making the airspace safer and more efficient. In 2010, FAA published a new requirement for aircraft that operate in certain conditions to have ADS-B out technology installed in the aircraft by 1st January 2020. This regulation was based on Title 14 of the US Code of Federal Regulations (14 CFR) sections 91.225 and 91.227. The regulation also required the ADS-B system to operate in US NAS to operate in 1090 and 978 MHz. As of

September 2017, the FAA already deploy an ADS-B ground station to support the ADS-B transmission and collected/relayed the collected ADS-B data.

For Europe, the regulation of ADS-B requirement that fly in the EU (European Union) air space is mandated in Regulation 1028/2014 and Regulation 2017/386. The regulation mandate, aircraft that operate in EU airspace that had criteria of fixed-wing aircraft with maximum take-off mass exceeding 5,700 kg or have maximum cruising true airspeed greater than 250 knots, will be required to have mode-s transponder and ADS-B out system installed by 7th June 2020. The regulation also mandates aircraft that will operate in the trans-Atlantic route to install the ADS-B Out system by 1st January 2020 to fill the FAA requirement fully. For required transmitted ADS-B data is base on EASA AMC 20-24 for ADS-B in Non-Radar Airspace or CS-ACNS for ADS-B out must include:

- Horizontal position of the aircraft (latitude/longitude)
- Barometric altitude of the aircraft (will be the same as for the SSR))
- Quality Indicator
- Aircraft Identification:
  - A unique 24-bit aircraft address
  - Identification of the aircraft
  - Mode A (In the case of CS ACNS for ADS-B Out)
- Emergency Status of the aircraft
- SPI (Special Position Indicator) when selected

In Indonesia, ADS-B research for ADS-B usage begins in 2007 that eventually produces a working ADS-B ground station equipment. The research aims to produce an ADS-B ground station equipment that can control air traffic around the airport and capable of detecting ADS-B out signal within 200 miles. As of 2016, the ADS-B ground station equipment already tested on Soekarno-Hatta international airport, which proving the equipment can detect aircraft within 250 Nm on 29,000 feet ("ADS-B alat navigasi penerbangan", 2016). Indonesia is reported to have

30 ground station that located in Banda Aceh, Matak, Soekarno-Hatta, Pangalan Bun, Cilacap, Palembang, Pekanbaru, Medan, Pontianak, Kintamani Bali, Palu, Ambon, Saumlaki, Alor, Waingapu, Tarakan, Galela, Timika, Kendari, Manado, Natuna, Makasar, Kupang, Sorong, Merauke, Malino Banjarmasin, Balikpapan, Biak, Surabaya, and Semarang (*Implementasi Automatic Dependent Surveillance Broadcast (ADS-B) di Indonesia | Nurhayati | WARTA ARDHIA*, 2014). Those ground stations are integrated into two separate ATM (Air Traffic Management) systems that controlling two different FIR (Flight Information Region), which are JAATS (Jakarta Automated Air Traffic Control System) and MAATS (Makassar Automated Air Traffic Control System). A conference in July 2018, DNP (Direktorat Navigasi Penerbangan) stated that starting in 1st January 2018 aircraft that operate in FL 290 to FL 600 is required to have ADS-B transmitter.

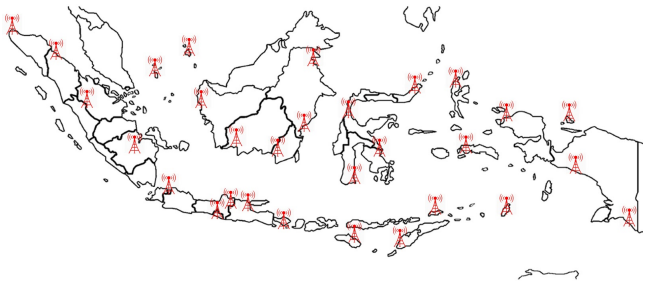

FIGURE 2.3: ADS-B ground station location in Indonesia

2.2.3 ADS-B Benefit

ADS-B is seen as the future of navigation system, which is proven by the United States that already made ADS-B as the primary system for upgrading and modernizing aviation infrastructure and operation. The program is called NextGen National Airspace Strategy that aims to replace the primary surveillance system from Radar to ADS-B. The reason is to improved safety and efficiency.

ADS-B can improve safety by increasing the situational awareness of the pilot. This is caused by the pilot have the ability to see clearer and detailed air traffic

around the aircraft from ADS-B data that the pilot received. The situational awareness of ATC also can be increase because of the same reason. ADS-B can also improve airspace usage efficiency because when the ATM uses the ADS-B system, it can reduce the separation standard between aircraft. The smaller separation can be enabled because ADS-B has a higher accuracy when determining an aircraft position, which leads to ATC can guide/track aircraft more accurately. These improvements were eventually increasing the capacity of an airspace.

2.3 Flight Path Reconstruction

Flightpath reconstruction is a method that uses historical data set to determining aircraft conditions such as aircraft position and speed(Evans, Goodwin, Feik, Martin, & Lozano-Leal, 1985). This was done by using the data from onboard sensors such as inertial and air data sensors. The historical data set can also be obtained by other systems such as ADS-B.

2.4 Interpolation

Interpolation is a data approximation method that will generate a new specific data point between the range of 2 or more known values. For example, a government is commencing a census of their population with ten years interval, however now the government wants to approximate the total population between each year between those two available data. So, those specific data approximation $p_n(x) = f_n$ where pn is interpolation, x is nodes, and f is the mathematical function which then we can use pn to get f value between two point of x.

Note: Interpolation method is only used to approximates value between two points; if the desired value is outside the range, it is called extrapolation.

For solving the interpolation problem, we can use a method of Lagrange interpolation. The method itself has two different approaches: linear interpolation and quadratic interpolation.

For Linear interpolation, the new value of pn will make a straight line between the known $valuex_0$ and x_1. The linear Lagrange polynomial can be express by:

$$p_1 = L_0 f_0 + L_1 f_1 \tag{2.1}$$

Where L is,

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, L_1(x) = \frac{x - x_0}{x_1 - x_0} \tag{2.2}$$

For the quadratic interpolation the new value of pn will made the quadratic line between the known value. The quadratic value of second degree polynomial can be acquired by:

$$p_1 = L_0 f_0 + L_1 f_1 + L_2 f_2 \tag{2.3}$$

Where L is,

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}, L_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \tag{2.4}$$

The interpolation method can be used if the value of pn is expected to unstable or fluctuated, which is called Spline. In general we can say that the generated pn between x_0 and x_1 can be bigger or smaller than the p_1 and p_0.

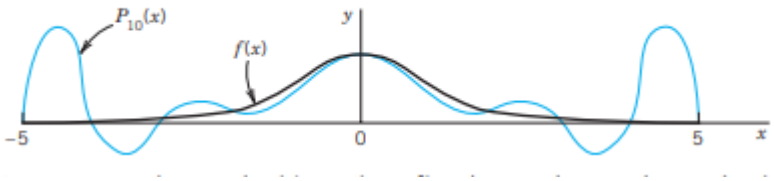

FIGURE 2.4: Spline Interpolation Example (KREYSZIG, 2010)

Based on the figure above, we can see that interpolation can have a more accurate value when it uses the Spline method while approximating fluctuate/unstable data.

Then there is another interpolation method called PCHIP or Piecewise Cubic Hermit interpolation Polynomial. PCHIP interpolation can be interpreted as a

shape-preserving piecewise cubic Hermite polynomial approach that will determine the slop from a function so that the generated value does not overshoot the data values(*pchip function | R Documentation*, 2020). PCHIP interpolation is also the opposite nature of Spline, where the PCHIP will maintaining the monotonicity of x and y value.

2.5 Kalman Filter

Kalman Filter is a type of estimation algorithm that uses a set of equations and a series of measurements observed over time, which have a noise (disturbance) in the measurement that can produce estimates of previously hidden variables. In Kalman Filter, the calculation process can be separated into 3 parts, which are measurement, prediction, and estimation.

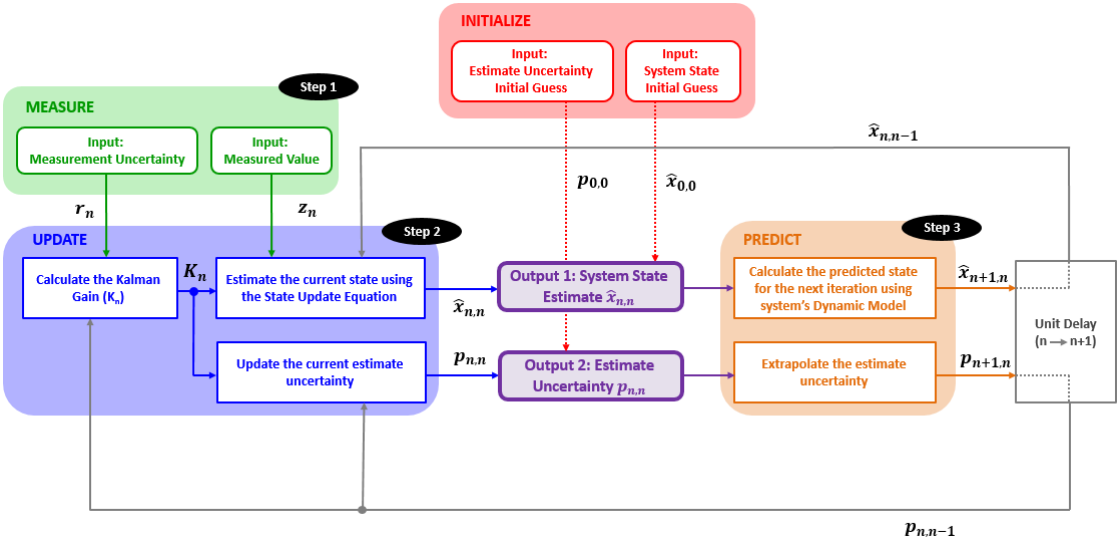

FIGURE 2.5: Kalman Filter Algorithm(Becker (www.kalmanfilter.net), 2018a)

2.5.1 Measurement

First of all, in the Kalman filter, the measurement value that was taken is always considered to have an error. In other words, we can not know the exact value or true value of a measurement. In Kalman filter, every measurement always contains

a random variable resulting in an error in the measurement. These will lead to varying results of the measurement value. However, in Kalman filter we always assume that the measurement always forms a normal distribution. These mean that the true value of the measurement is close to the repeated measurement's mean or average value. Although this condition is not always applied to every system, because in the real world, the measurement tool not always had high accuracy measurement. If the system has high accuracy measurement, then the repeated measurement's mean value is close with the true value, this type of system is called a unbias system. On the other hand, if the system had low accuracy, then the mean value and the true value have an enormous difference; this type of system is called bias system.

In the Kalman filter there 2 types of models that can describe the whole system. First, there is a dynamic model, which mean that the system true value is changing over time, for example, if Kalman filter is used in moving aircraft case. The second one is constant dynamic model, which means the system, in reality, has a constant true value, for example, if we measured the height of a building.

As mentioned previously, in real-world cases, we do not know the true value of a system, and that every measurement always had an error caused by the random variable. The Kalman filter calculation objective is to minimize error or uncertainty in the measurement or referred to as covariance, by doing multiple iterations of the calculation. If the covariance is minimized, then the measured value should be close with the "true value". The initial measurement usually referred to as initial estimate, the estimate is denoted by $\hat{\boldsymbol{x}}_{\boldsymbol{n,n}}$ the first n is for the iteration number of the prediction, and the second n is for the first number of the estimate, so the initial estimate is $\hat{\boldsymbol{x}}_{\mathbf{0,0}}$. The covariance is denoted by $\hat{\boldsymbol{P}}_{\boldsymbol{n,n}}$

2.5.2 Prediction

The usage of the Kalman filter can be divided into two models. The first one is one-dimensional Kalman filter. Which are used for single dimensional measurement; for example, the Kalman filter is used for measuring the temperature of an object. The other model is a multi-dimensional Kalman filter used for the system that multi-dimensional measurement. This model was used in this research because a 3-dimensional system was taken into consideration; that is mean the position,

speed, and acceleration in x, y, z axis. The model needs to be determined because the single-dimensional and multi-dimensional calculations used a different set of equations. The significant difference between the two models is that the equation was calculated in matrix form when calculating the multi-dimensional Kalman filter.

As have been mention in the previous section the first measurement of a system in called initial estimation. This initial estimation is consider to be starting point of the Kalman filter calculation. The initial estimate of a system is also have the covariance which both of this value will be calculated in the prediction state. Then all known value is combine in state extrapolation whihch is:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n+1,n}} = \boldsymbol{F}\hat{\boldsymbol{x}}_{\boldsymbol{n,n}} + \boldsymbol{G}\hat{\boldsymbol{u}}_{\boldsymbol{n,n}} + \boldsymbol{w}_{\boldsymbol{n}} \tag{2.5}$$

Where,

$\hat{\boldsymbol{x}}_{\boldsymbol{n+1,n}}$ is the predicted system vector at time step n+1

$\hat{\boldsymbol{x}}_{\boldsymbol{n,n}}$ is the current estimate system vector/initial estimate (State matrix)

$\hat{\boldsymbol{u}}_{\boldsymbol{n,n}}$ is the control/input variable which is a measurable input to the system

$\boldsymbol{w}_{\boldsymbol{n}}$ is process noise, which is an un-measurable input that affects the state

$\boldsymbol{F}$ is state transition matrix

$\boldsymbol{G}$ is a control matrix or input transition matrix (mapping control to state variable)

In the changing dynamic model, the $\hat{\boldsymbol{u}}$ is a variable that affecting the state matrix. For example, an aircraft that moves in x, y, z axis, and has speed in all of those axes will have an input variable in the form of acceleration. Then the F and G is a transition matrix that converts the variable of state and input matrix to compatible value so both matrices can be calculated. If the Kalman filter is used for a constant dynamic model the input variable can be considered as 0. In general, the purpose of the state extrapolation equation is to predict the state of the object after a certain time. For example, if we know the position, speed and acceleration of an aircraft, we can predict aircraft position and aircraft speed after a certain time (Δt). In the prediction state, the covariance of a system need also to be predicted by using:

$$\boldsymbol{P}_{\boldsymbol{n+1,n}} = \boldsymbol{F}\boldsymbol{P}_{\boldsymbol{n,n}}\boldsymbol{F}^{\boldsymbol{T}} + \boldsymbol{Q} \tag{2.6}$$

Where,

$\boldsymbol{P_{n,n}}$ is estimate uncertainty (covariance) of the current state

$\boldsymbol{P_{n+1,n}}$ is the updated estimate uncertainty

$\boldsymbol{F}$ is the state transition matrix(Same matrix that used in state extrapolation)

$\boldsymbol{Q}$ is the process noise matrix

In general, the purpose of the equation 2.6 is to readjust the parameter of the covariance matrix ($\boldsymbol{P_{n,n}}$) so it can be calculated in the equation.

2.5.3 Estimation

In the last stage of Kalman filter calculation, we get the prediction value of an object for the next measurement. For example, in moving aircraft, we can get the prediction of the aircraft position after a specific time. However, this prediction still had a noise that made the prediction value not accurate. Previously, this noise is defined as a random variable resulting in an error in the prediction; as mentioned before, Kalman filter's goal is to minimize this error, which will be done in this stage.

The first thing to be done in this stage is to know the "actual" measurement value of the object after a certain time that can be obtained by performing the second measurement. If the second measurement is already obtained, then the measurement needs to be converted with an equation that made the measurement value compatible with the Kalman filter equation. This conversion equation is known as Measurement equation:

$$\boldsymbol{z_n} = \boldsymbol{H}\boldsymbol{x_n} + \boldsymbol{v_n} \tag{2.7}$$

Where,

$\boldsymbol{z_n}$ is the measurement vector

$\boldsymbol{x_n}$ is the system state/measurement value

$\boldsymbol{v_n}$ is random noise

$\boldsymbol{H}$ is the observation matrix

The purpose of the $\boldsymbol{H}$ is to transform the state matrix or the measurement value into the desire compatible matrix that can be calculated with other parameters.

In general the next thing to be done in this phase is to comparing the result of the prediction of the system state $\boldsymbol{x_{n,n}}$ with the "actual" measurement of the system state $\boldsymbol{z_n}$. This comparison will resulting more accurate estimation of the system state. In actuality the comparison need an additional parameter that act as a correctional value that basically tell how much adjustment the system need so it can produce more accurate estimation. This adjustment parameter or commonly called kalman gain can be obtain by using:

$$\boldsymbol{K_n} = \frac{\boldsymbol{P_{n,n-1}H^T}}{\boldsymbol{HP_{n,n-1}H^T + R_n}} \tag{2.8}$$

Where,

$\boldsymbol{K_n}$ is Kalman gain

$\boldsymbol{P_{n,n-1}}$ is the previous covariance of the system

$\boldsymbol{R_n}$ is the measurement uncertainty/error

The Kalman gain is then used as a variable that controls the trade-off between the prediction of future values and the observation of current values. In other words, the Kalman gain tells how much we want to change the estimate by given measurement and prediction. The value of Kalman gain is only range between 0 and 1 or $0 \leq K_n \leq 1$. So the Kalman gain is used in the comparison equation between the prediction and "actual" measurement equation or also known as state update equation, which is:

$$\boldsymbol{\hat{x}_{n,n}} = \boldsymbol{\hat{x}_{n,n-1}} + \boldsymbol{K_n}\left(\boldsymbol{z_n} - \boldsymbol{H\hat{x}_{n,n-1}}\right) \tag{2.9}$$

where, $\boldsymbol{\hat{x}_{n,n}}$ is the new estimated state vector

$\boldsymbol{\hat{x}_{n,n-1}}$ is the current predicted system state vector

$\boldsymbol{K_n}$ is kalman gain

$\boldsymbol{H}$ is the observation matrix

$\boldsymbol{z_n}$ is the measurement vector

Intuitively if we get more accurate value of the state system, the value of the covariance should be decreasing. To obtaining the new estimation value of the covariance we can use the covariance update equation:

$$\boldsymbol{P_{n,n}} = \left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)\boldsymbol{P_{n,n-1}} \tag{2.10}$$

Where,

$\boldsymbol{P_{n,n}}$ is estimate uncertainty (covariance) matrix of the current sate

$\boldsymbol{I}$ is identity matrix

$\boldsymbol{P_{n,n-1}}$ is the previous covariance of the system

$\boldsymbol{K_n}$ is kalman gain

In general, the value of covariance should always be decreasing if the Kalman filter calculation is conducted continuously. Suppose the covariance is continuously decreasing with every calculation iteration. In that case, the value of the Kalman gain should also be decreasing because if the error/covariance is smaller, then the adjustment value(Kalman gain value) that was added in state update equation also becomes smaller.

If the new state estimation and new covariance are have been obtained, both of the values then inserted again to the prediction phase. This loop of prediction and estimation calculation is continuously done until the last measurement value is calculated. This looping calculation can be described as:

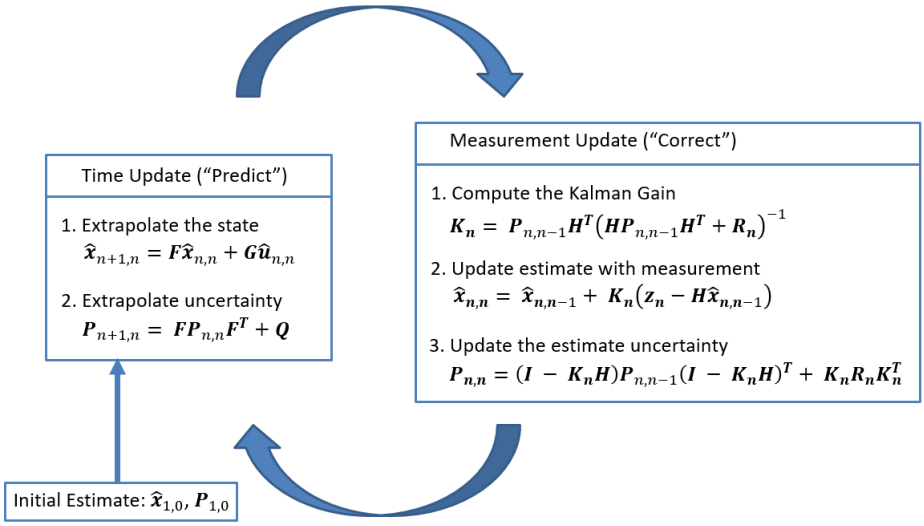

FIGURE 2.6: Kalman filter calculation loop (Becker (www.kalman-filter.net), 2018b)

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Data Acquisition

The data set used in this research was obtained from opensky-network.org. Specifically, this research used the data from March 9th 2020. Although Opensky-network offers other data set from different dates since the website adds new data once a week, this research only used the March 9th 2020 data as the case study. In the Opensky-network website, the March 9th 2020 data set is divided hourly from 00 to 24 based on GMT+0 time zone. This research only used the data from 00 to 10 because of the hardware limitation to store and read the data set. The data set is downloaded in the form of a zip file containing a CSV file of the ADS-B data. The data set from Opensky-network had several parameters, including:

- Time
- Icao24 ID
- lat (Lattitude)
- lon (Longitude)
- Velocity
- Heading
- Vertrate
- Callsign
- Onground
- Alert/SPI
- Squawk
- Baroaltitude
- Geoaltitude
- lastposupdate
- lastcontact

However, in this research, only 9 data parameters were used. Which are:

- Time: The timestamp of the data point is based on GMT+0 timezone, but the timestamp is in form of UNIX timestamp. The timestamp includes the date, month, year, hour, minute, and second description.
- Icao24 ID: Containing a unique 24-bit ICAO transponder ID that is used to differentiate aircraft/airframe.
- lat (Latitude): Geological latitude coordinate that is based on WGS84.
- lon (Longitude): Geological longitude coordinate that is based on WGS84.
- Velocity: Containing the magnitude of aircraft speed over ground in m/s(Meters per Seconds)
- Heading: Containing the direction of movement (track angle) as the clockwise angle from the geographic north.
- Vertrate: Containing the vertical speed of the aircraft in m/s
- Onground: Indicates whether the aircraft is broadcasting surface positions or airborne positions. If the aircraft is on the ground, it will show as True, and if the aircraft is airborne, it will be shown as False.
- Geoaltitude: Containing the altitude that determined using the GNSS (GPS) sensor

After extracting all the CSV files from the zip file, it needed to be merged into 1 file. The reason is to make it easier to process all the data. To merge the CSV files, this research used Python programming language. Several libraries need to be used, which are *os* and *glob* library. The code that used to merge the file are as follow:

```
#Determining the location of the file
path = 'C:\\Users\yttbk\.spyder-py3\Opensky'
extension = 'csv'
os.chdir(path)
#combining the csv file with a loop
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
```

```
combined_csv = pd.concat([pd.read_csv(f) for f in all_filenames ])
#export to csv
combined_csv.to_csv( "combined_csv.csv", index=False,
encoding='utf-8-sig')
```

After the merger, the total data point in the data set is 18855925. However, this file covering all data from all over the world, and both aircraft that still on the ground and airborne. In conducting the FPR with Kalman filter and interpolation, the calculation needs only need one aircraft that already airborne.

To filter the data set to the desire condition, first, the data point that contained on the ground aircraft needs to be removed. To accomplish this, the "onground" parameter needs to be filtered; if the data has a "True" value, the data point was removed. This can be done by using:

```
df[df['onground'].isin([false])]
```

Those codes mean that if the column "onground" did not have "False" value, it was removed. After initiating this code, the total data point decreases to 17117696.

Then to simplified the filtration, the data need to be filtered by region/country. For this research, the aircraft that fly in the US was used. The data set need to be filtered again, so aircraft that did not fly in the US were removed. To accomplish that in Python, the data set can be filtered by the range of the latitude and longitude of the United States of America or filtered by the maximum and minimum of the US's geographical coordinate. The latitude range is from 19.50139 to 64.85694, and the longitude is from -161.75583 to -68.01197(*United States latitude and longitude*, 2015). Then those coordinate range will be used as the base value of the filtration in Python. The code to filter the coordinate range is:

```
df = df[(df['lat'] >= 19.50139) & (df['lat'] <= 64.85694)]
df = df[(df['lon'] >= -161.75583) & (df['lon'] <= -68.01197)]
```

After the filtration, there only 6009740 data point left. From this 6009740, there are 9719 unique aircraft listed on the data set. So, another filtration is needed to filter one unique aircraft. To done that, another line of code is needed to filter one unique ICAO24 ID. These codes are:

3.2 Data analysis

In this research, two types of analysis methods are conducted, which are interpolation and Kalman filter. The analysis aimed to produce the estimated position of the aircraft with reduced noise/inaccuracy and the position of the aircraft between periodical updates or, in other words, interruption/intermission periods. In the interpolation method, there are three different types of interpolation that were conducted in this thesis. The first one is linear interpolation, spline interpolation, and PCHIP (Piecewise Cubic Hermite interpolating Polynomial) interpolation.

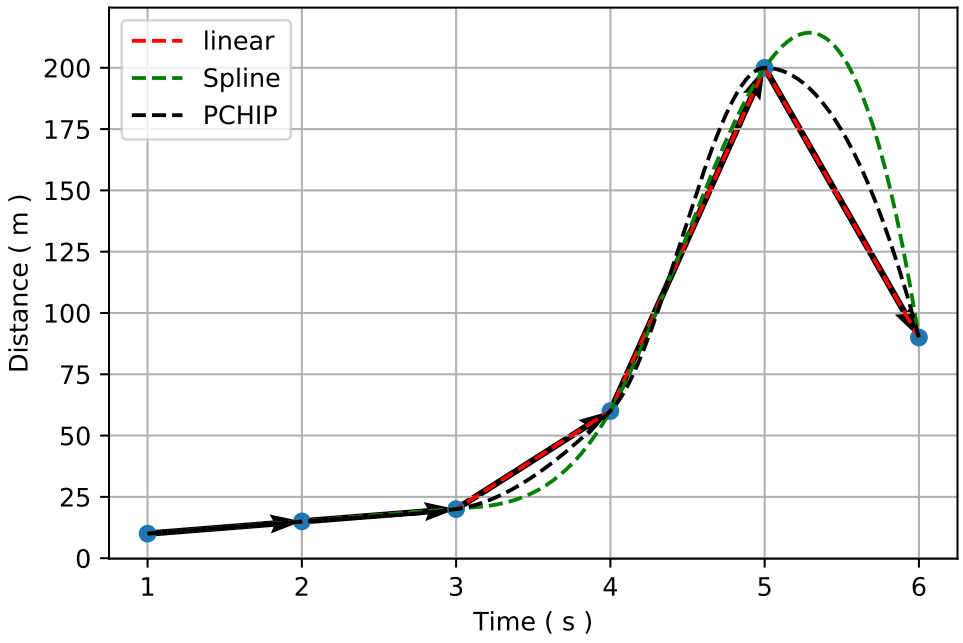

FIGURE 3.1: Example of 3 Differents Interpolation Methods(Aircraft displacement overtime)

Note: The figure 3.1 is an example of data of an aircraft displacement overtime which have been interpolate with 3 different interpolation methods.

3.2.1 Interpolation

Linear interpolation is a type of interpolation that creates a new value, which made a continuous straight line (shortest straight line) between each data point, which

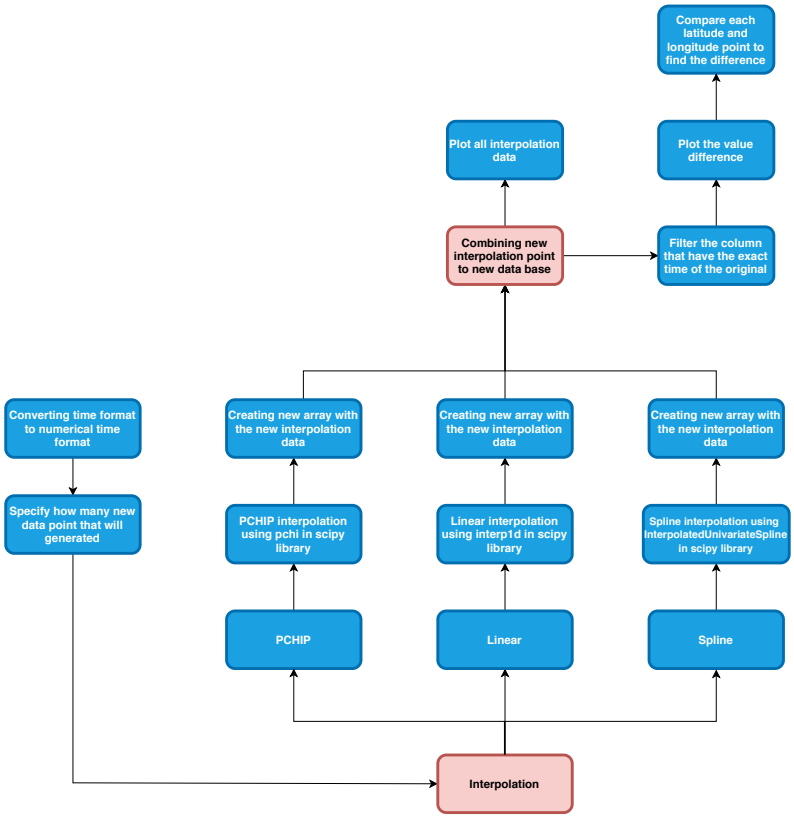

FIGURE 3.2: Algorithm of Trajectory Interpolation

means have a lesser quality of data. In this thesis, the Python programming was used the *scipy* library to produce all three different interpolation approaches. In the *scipy* library, linear interpolation can be immediately performed between the value of latitude and longitude (x and y coordinate, respectively). However, the PCHIP and spline interpolation method can not be performed directly because the *scipy* method needs one of the values of longitude or latitude that represent the x-axis to be a monotonic increasing value. This means that the x-axis value does not have any repetitive value and always increasing value along the x-axis. That is why the *scipy* library cannot directly be implemented because it is impossible to assume an aircraft always has this trajectory pattern.

To overcome this problem, there should be a new value that needs to be introduced, which had the monotonic increasing value and also represent the exact order of the coordinate movement. The value happens to be the timestamp data of the data points which is represented by time description in the data set. The

timestamp value is intended to replace the x value in the interpolation so that one of the axes still maintained a monotonic increasing value. As a result of that, the interpolation calculation was separated into two sections. The first one is for the time and longitude as x and y. The second section is for the time and latitude as x and y. Although the linear interpolation does not need this procedure, it is important to make sure the result is comparable to each other, in order to uniformize the procedure.

As mentioned before, the interpolation needs to be conducted separately, but the code/procedure is almost the same; the difference only occurs when the longitude and latitude value are called. Before using the Scipy library to commencing interpolation between the time and coordinate. The timestamp value format needs to be changed. From the original UNIX time format to conventional format. This format is consists of year, month, date, hour, minute, and second. This was done so that in the analyzation process, the result becomes more readable for human. To perform this process, this research uses the *pandas* library. The overall code to do this is:

```
time_date = pd.to_datetime[df{'time'}]
```

However, the *scipy* library did not recognize the conventional time format. So, the format needs to be converted again to numerical format. From datetime format to numerical format using matplotlib.mdate library.

```
 time_num = mdates.date2num(time_date)}
```

Then the total of new points or nodes needed to be declared, which was made in the interpolation processes. To maximize the detail in the result and the hardware limitation, the total of new points is 10 million. The new point can be made by declaring the start and final value, and then the program will made 10 million values between the start and finish. The start and final value can be determined by the minimal and maximum value of time data. The new point/node can then be made by using NumPy library.

```
timeinterpnum = np.linspace(min(timenum), max(timenum), 10000000)
```

After that, Interpolation can be done between each coordinate of latitude and longitude. All of the Interpolation can be done using Scipy.interpolate library.

For linear interpolation:

```
latiinterplin = interpolate.interp1d(timenum, lati)
latiinterplin = latiinterplin(timeinterpnum)
```

Note: For latitude

```
{longinterplin = interpolate.interp1d(timenum, long)}
{longinterplin = longinterplin(timeinterpnum)}
```

Note: For longitude

For spline interpolation, it needs to replace all the interp1d to InterpolatedUnivariateSpline. The difference between linear and Spline interpolation is that it will create an interpolation of a new point that will be made a cubic line between existing data points (the new point will have value outside the range of the existing data value).

For PCHIP interpolation replace all the inretp1d with pchip. Also, for this interpolation, it will create an interpolation of a new point that will be made into a smooth curve line between existing data points(the new point will have value outside the range of the existing data value). This interpolation approach's general idea is to mimic the real aircraft movement when turning, hence the benefit of a smooth curve interpolation generate.

After all interpolation approach has been done, the result was plotted. To plot the result, *matplotlib* library is used. The plot is made with the original longitude and latitude with the combination of the interpolated longitude and interpolated latitude. For the exact magnitude difference, this research is using MSE(Mean Square Error). The Formula of MSE can be written as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{3.1}$$

3.2.2 Kalman Filter

Before calculating using the Kalman filter. The data parameters must be adjusted. The adjustment must be made because, in Kalman filter, the calculation is

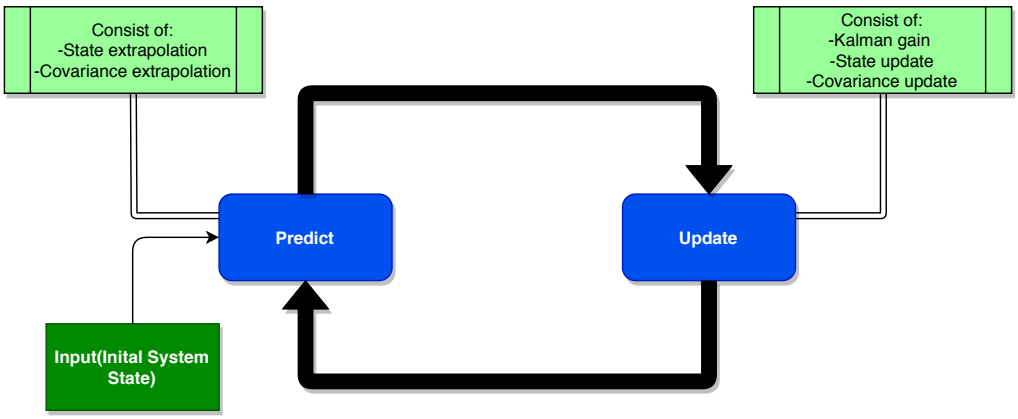

FIGURE 3.3: Algorithm of Kalman Filter Calculation

computed in vector form. In order to fulfill those criteria, the velocity parameter that still in the scalar form must be converted into a 3-dimensional vector form. The vector velocity was consisting of v_x, v_y, and v_z that correspond to velocity in latitude axis, velocity in longitude axis, and the vertical velocity. Although the vertical velocity is obtainable from the ADS-B data's vertrate parameter, the v_x and v_y must be calculated. To converting the scalar form, it will use:

$$v_x = vsin(c)$$

$$v_y = vcos(c)$$

where,

v is velocity

c is the direction of movement of the aircraft as the clockwise angle from the geographic north

In the Python programming language using *numpy* library, the code will be:

```
1 Vx = df['velocity'] * np.cos(df['heading']) #Speed in X axis (m/s)
2 Vy = df['velocity'] * np.sin(df['heading']) #Speed in Y axis (m/s)
3 Vz = df['vertrate'] #Speed in Z axis (m/s)
```

In this Kalman filter calculation, the input variable will be using the acceleration of the aircraft. However, in the ADS-B data, the acceleration data is unavailable. So, an additional calculation is needed. To find the acceleration of the aircraft, we

can use the velocity difference (Δv) and time difference (Δt) between data points. In the mathematical form it will be shown as:

$$a = \frac{v_2 - v_1}{t_2 - t_1} = \frac{\Delta v}{\Delta t}$$

where,
a is acceleration in m/s^2
v is scalar velocity in m/s
t is time in second s
In python programming language the time difference can be find by using:

```
time_diff = (df['time'] - df['time'].shift())
time_diff = pd.Series(pd.to_timedelta(time_diff))
time_diff = time_diff.dt.total_seconds() #time difference in second
df['time_difference'] = time_diff #declaring new data set in the data frame
```

Then after the time difference is obtained, the acceleration can be found and the vector acceleration can be found using the same method as the vector velocity. In python programming language the code will be:

```
Ac = (df['velocity'] - df['velocity'].shift()) / df['time_difference']
Ax = Ac * np.cos(df['heading'])
Ay = Ac * np.sin(df['heading'])
Az = ((df['vertrate'] - df['vertrate'].shift()) / df['time_difference'])
```

Then the coordinate position of the aircraft needs to be changed so that it was compatible with other variables. The coordinate represents the x, y, and z vector, which is latitude, longitude, and altitude were changed into the Earth-center Earth-fixed coordinate system or known as ECEF. The ECEF is a type of conventional terrestrial coordinate system that determines its position by measuring the distance between the position to the center of the earth. The ECEF will be combining the distance to the center of the earth with the degrees of latitude and longitude. Unlike the Geographic coordinate system that determines the position from the degrees displacement from the equator for determining the latitude and degrees displacement from the Greenwich point for determining the longitude.

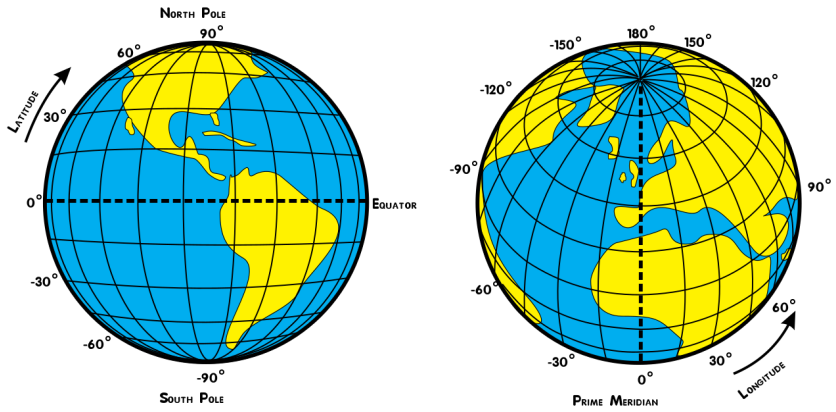

FIGURE 3.4: Geographic coordinate system (*Preprocessing: Calculate the Distance Between Longitude and Latitude Points with a Function | by The Data Detective | Towards Data Science*, 2019)

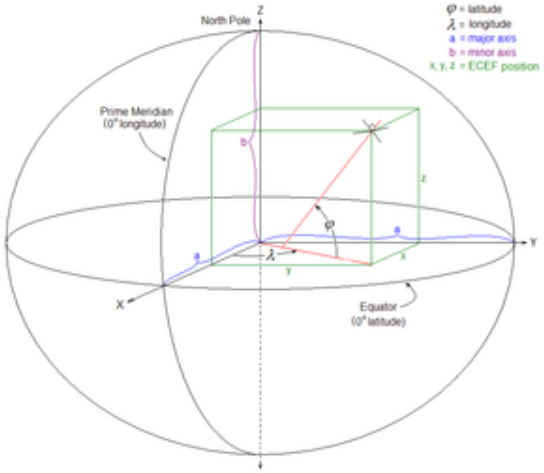

FIGURE 3.5: ECEF coordinate system (*ECEF(Earth-Centered Earth-Fixed Frame) Coordinate*, 2016)

To convert the coordinate system, a python library was used. The library is *pymap3d*. The library only needs input from the original position coordinate, consisting of latitude, longitude, and altitude. By using the geodetic2ecef command in the library, the conversion was initialized. The code is done in a loop so that all coordinate for every timestamp are converted:

```
1 lat = (df3['lat'])
2 lon = (df3['lon'])
```

```
h = df3['geoaltitude']

x_coor,y_coor,z_coor = [], [], []
x_coor,y_coor,z_coor = pm.geodetic2ecef(lat,lon,h)
```

In order to calculate the Kalman Filter with Python programming, this research will use *filterpy* library. In general, the library consisting of 2 step process just like the actual Kalman Filter calculation. The first one is the predicted step, and the second one is the updated step. In the predicted step, the calculation of the state extrapolation and covariance extrapolation was done to produce the future prediction of the aircraft state. Then in the updated stage, the Kalman gain, state update, and covariance update equation were calculated to produce more accurate information of the current aircraft state. However, before the Cartesian variable is inputted in the filterpy, the matrix form for the equation need to be defined.

The *filterpy* library already providing a command that enabling the user to input the parameter separately. However, the *filterpy* can only compute a system that has two dimensions. For example, the position of x and y axis with its speed components. So the calculation will be conducted in 2 separate forms. The first one will be calculating the x and y axis component, and the second will be calculating the z axis component. The x and y component calculation will be referred as *kf1* and the z component calculation will be referred as *kf2*.

The parameter that will be inputted in the first calculation(x and y axis) are:

State system $\boldsymbol{x_n}$ using KalmanFilter.x command:

$$kf1 = \boldsymbol{x_n} = \begin{bmatrix} x \\ y \\ V_x \\ V_y \end{bmatrix} \qquad kf2 = \boldsymbol{x_n} = \begin{bmatrix} z \\ V_z \end{bmatrix} \tag{3.2}$$

Note: x, y and z is the coordinate in ECEF form.

State transition matrix $\boldsymbol{F}$ using KalmanFilter.F command:

$$kf1 = \boldsymbol{F} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad kf2 = \boldsymbol{F} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

Note: dt is the delta time between the current timestamp and future timestamp.

Input variable $\boldsymbol{u}$ using KalmanFilter.u command:

$$kf1 = \boldsymbol{u_n} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \qquad kf2 = \boldsymbol{u_n} = \begin{bmatrix} a_z \end{bmatrix} \tag{3.4}$$

Note: a_x, a_y and a_z is the acceleration parameters.

Input transition variable $\boldsymbol{B}$ using KalmanFilter.B command:

$$kf1 = \boldsymbol{B} = \begin{bmatrix} dt^2/2 & 0 \\ 0 & dt^2/2 \\ dt & 0 \\ 0 & dt \end{bmatrix} \qquad kf2 = \boldsymbol{B} = \begin{bmatrix} dt^2/2 & 0 \\ 0 & dt \end{bmatrix} \tag{3.5}$$

Covariance matrix $\boldsymbol{P_n}$ using KalmanFilter.P command:

$$kf1 = \boldsymbol{P_n} = \begin{bmatrix} VAR(x) & 0 & 0 & 0 \\ 0 & VAR(y) & 0 & 0 \\ 0 & 0 & COV(V_x, V_x) & COV(V_x, V_y) \\ 0 & 0 & COV(V_y, V_x) & COV(V_y, V_y) \end{bmatrix} \tag{3.6}$$

$$kf2 = \boldsymbol{P_n} = \begin{bmatrix} VAR(z) & dt \\ 0 & VAR(V_z) \end{bmatrix}$$

Note: $VAR(x)$, $VAR(y)$ and $VAR(z)$ is the square of GPS accuracy, this research will use 20 meter(*Global Positioning System (GPS) in Aviation | Aircraft Systems*, 2017). The COV section means that the velocity of one vector was multiplied with other vectors. The velocity of every vector is the same, which is 4 knot or 2.05778 m/s(*Civil Aviation Order 108.56 Instrument 2007*, 2007). The VAR means that the noise of a vector does not affect one another, which the opposite of the COV the noise in every vector is effecting others vector. Hence why the position is using COV or, in other words, is covariance except the V_z because it does not calculate with others vector velocity and the position using VAR or variance.

measurement matrix $\boldsymbol{H}$ using KalmanFilter.H command:

$$kf1 = \boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad kf2 = \boldsymbol{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

Measurement covariance matrix $\boldsymbol{R}$ using KalmanFilter.R command:

$$kf1 = \boldsymbol{R_n} = \begin{bmatrix} VAR(x_m) & 0 & 0 & 0 \\ 0 & VAR(y_m) & 0 & 0 \\ 0 & 0 & COV(V_x, V_x) & COV(V_x, V_y) \\ 0 & 0 & COV(V_y, V_x) & COV(V_y, V_y) \end{bmatrix} \tag{3.8}$$

$$kf2 = \boldsymbol{R_n} = \begin{bmatrix} VAR(z_m) & dt \\ 0 & VAR(V_z) \end{bmatrix}$$

Note: x_m, y_m and z_m is based on the accuracy indicator or NIC value. It needs to be noted that these values can fluctuate between each iteration.

For the process noise matrix, $\boldsymbol{Q}$ is not used in the calculation because of the unavailability of the data parameter.

Usually, the ADS-B data have the quality indicator value that consists of NIC,

NACp, or SIL, but the data set provided by OpenSky-network did not have it. So, in this research, the quality indicator is generated by using python code. Considering this research use the NIC value as the bases for the quality indicator, the magnitude of the inaccuracy or noise was based on NIC value.

The NIC value is generated randomly with predetermine probability. Only the maximum value of each NIC value will be used in value generation. So, the probability of each NIC value is:

NIC Value	Probability
0	2.39%
1	0.01%
2	0.01%
3	0.07%
4	0.01%
5	0.01%
6	0.01%
7	0.01%
8	0.08%
9	42.92%
10	50.59%
11	3.93%

TABLE 3.1: NIC Value Probability (Tesi & Pleninger, 2017)

Note: For the NIC 0 the horizontal accuracy is supposed to be unknown, but in this research, the horizontal accuracy of the NIC 0 will be 20 meters. For the vertical accuracy of the NIC value, less than 8 the 20-meter magnitude also will be used.

Then the equation process can be start by initiating an predict and update command in looping sequence for every timestamp:

```
#for kf1
for i in range(count):
pos = Measurement[i]
z = pos
kf1.predict ()
```

```
kf1.update (z)
xs.append (kf1.x[0,0])
ys.append (kf1.x[1,0])
vxs.append (kf1.x[2,0])
vys.append (kf1.x[3,0])
pxs.append (pos[0])
pys.append(pos[1])
```

```
#for kf2
for i in range(count):
pos2 = Measurement2[i]
z = pos2
kf2.predict ()
kf2.update (z)
kzs.append (kf2.x[0,0])
vzs.append (kf2.x[1,0])
pzs.append (pos2[0])
pvz.append(pos2[1])
```

The resulting new estimate coordinate is still in the ECEF form, so it needed to be reconverted to geodetic form. The reason for this change is to compare the original data and new estimated data. To accomplish that task, a method of determining the data difference is needed. This research will be using the MSE (Mean Squared Error) method.

The reconverted coordinate also plotted using *basemap* library. So that the actual aircraft trajectory of estimated data from Kalman filter and original ADS-B data can be visualized in the geodetic format. The original result of the ECEF coordinate also will be plotted with *matplotlib* library, so that the difference between original ECEF ADS-B coordinate and Kalman filter data can be analyzed.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Interpolation Result

As mentioned before, in the previous chapter, the interpolation result was plotted with the original ADS-B data. This research was intended to create four different plots. Three of them contain a single interpolation and original data, and one of them contains all available data (all interpolation results and original ADS-B data). This was intended to visualize the result and visualize the comparison between the interpolation data and original ADS-B data.

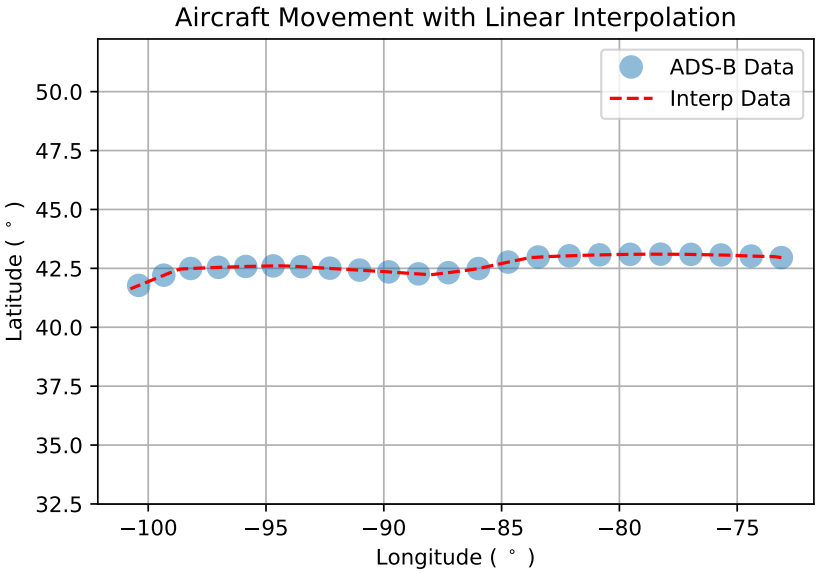

FIGURE 4.1: Linear Interpolation

As mention before the figure 4.1, 4.2, 4.3, and 4.4 is the result of comparison between interpolation data and ADS-B data. Originally the ADS-B data has 1119 timestamp or, in other words, 1119 data points. If all the original ADS-B data

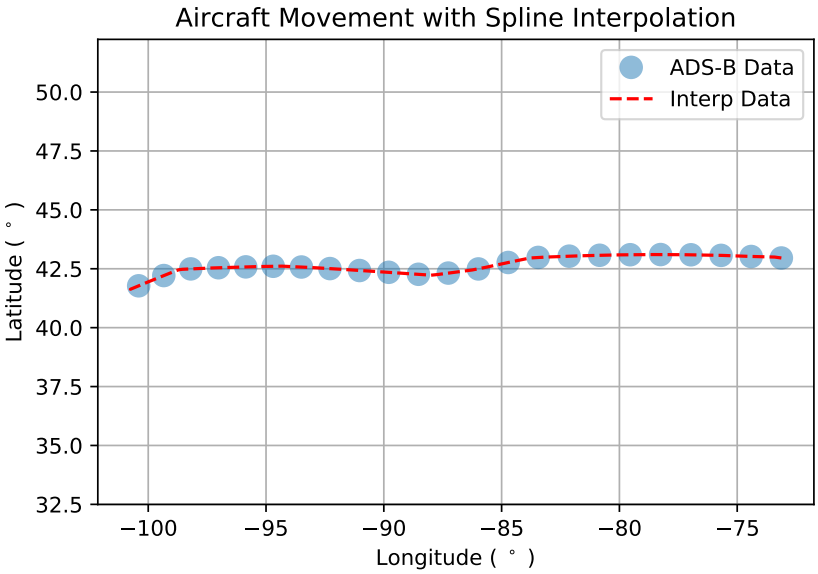

FIGURE 4.2: Spline Interpolation

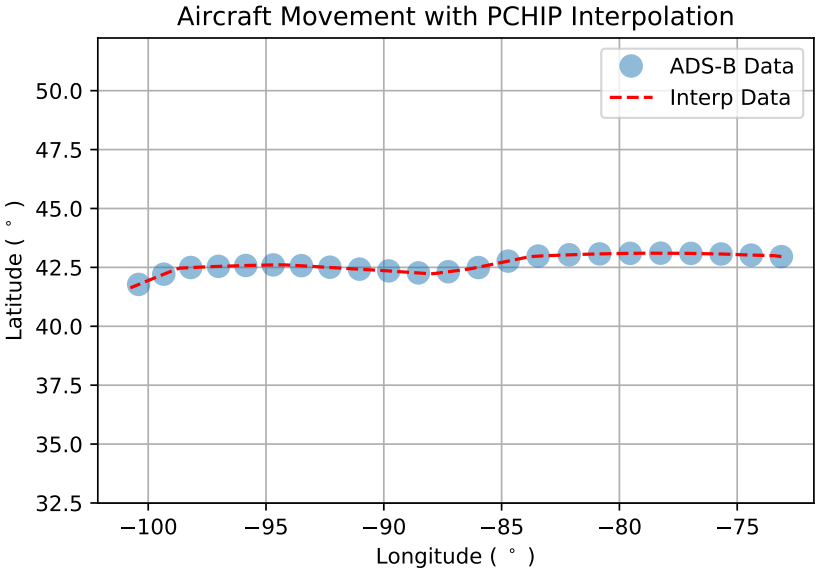

FIGURE 4.3: PCHIP Interpolation

points are used, each data point's separation is unseen able. To overcome this, the plot only shows a data point in 50 data point intervals, or only the 50^{th} data point is shown. To make each data point more visible in the plot, this research uses the scatter plot mode in the *matplotlib* library.

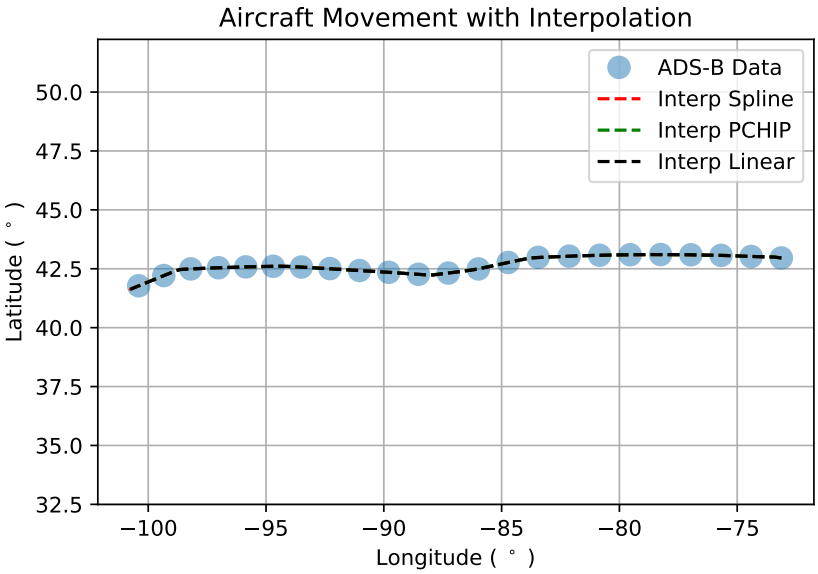

FIGURE 4.4: Combination of All Interpolation

As mentioned in the previous chapter, originally, there is the sum of the total original data point plus 10 million new data points created in interpolation calculation. So, in general, there are 10 million and 1119 data points that were created in the interpolation calculation. This resulted in 10 million new data points are filling in the timestamp that was not covered in the original ADS-B data. As a result of that, the delta time between each data point becomes smaller. In order to visualize the smaller time interval between each data point as seen in figure 4.1, 4.2, 4.3, and 4.4, this research used the line plot when using *matplotlib* library. This was aimed to resemble the continuous aircraft trajectory during flight, or in other words, it will resemble the interpolation data that does not have data point intervals.

As seen in figure 4.4 that the interpolation result between 3 interpolation methods did not have a visual difference as expected to see in the example in the figure 3.1. The reason is that the small periodical separation in the original data is not big enough that eventually made the graph indifference. The average delta time in this data example is 10 seconds.

Then, an analysis of the magnitude difference between interpolation data and ADS-B data is needed. For this analysis, this research will use the MSE method. To

be able to conduct this analysis first, both data need to have matching timestamp value. To do this, the interpolation data need to filter. This aims to remove data points that did not have a matching timestamp with the original ADS-B data. After that is done, then the analysis can begin. The MSE calculation was done in 2 separate forms. The first one is for the MSE between latitude and longitude coordinate of interpolation data and original ADS-B data. The second one is for distance difference in meter unit between each coordinate.

MSE of Coordinate Difference	MSE (degree)
Latitude of Linear Interpolation and ADS-B	0
Latitude of Spline Interpolation and ADS-B	5.46×10^{-28}
Latitude of PCHIP Interpolation and ADS-B	0
Longitude of Linear Interpolation and ADS-B	0
Longitude of Spline Interpolation and ADS-B	5.46×10^{-28}
Longitude of PCHIP Interpolation and ADS-B	0

TABLE 4.1: MSE of Latitude and Longitude

MSE of Distance	MSE (m^2)
ADS-B Data and Linear Interpolation	0
ADS-B Data and Spline Interpolation	2.74×10^{-18}
ADS-B Data and PCHIP Interpolation	0

TABLE 4.2: MSE of Distance

As seen in table 4.1 and 4.2 the value MSE is small. As a result of the MSE is above 10×10^{-14} the error can be ignored. So, in other words, the resulting interpolation coordinate has no difference with the original ADS-B data. However, it needed to remember that in reality, both values still had a noise or inaccuracy because, in the interpolation calculation, the noise and inaccuracy are not taken into consideration during the calculation.

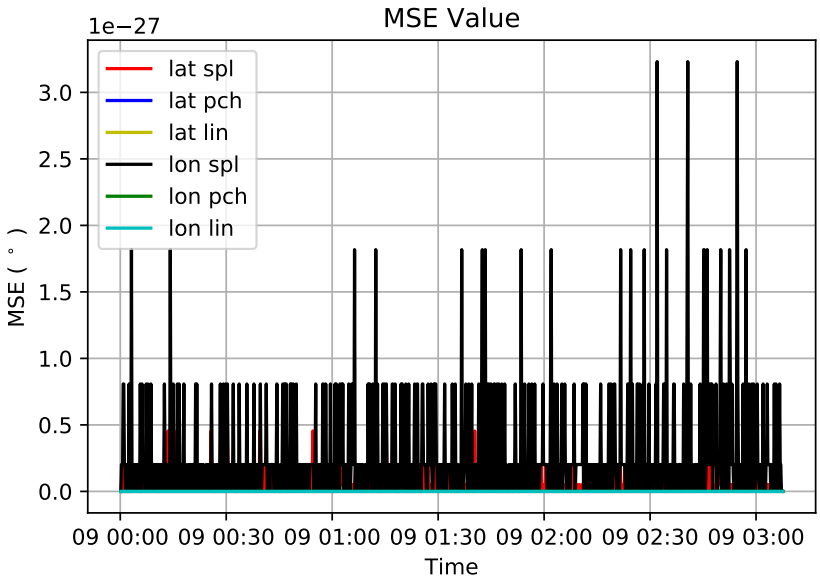

FIGURE 4.5: MSE of Latitude and Longitude Over Time

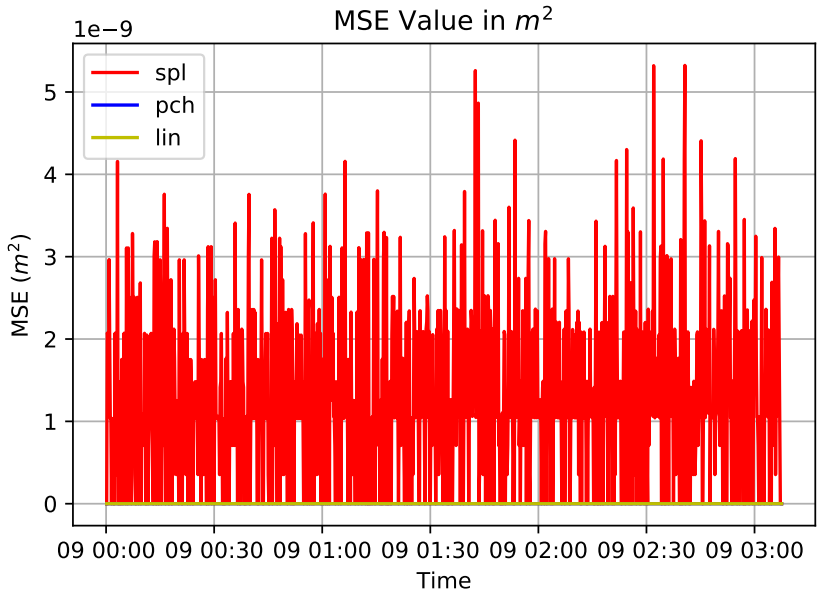

FIGURE 4.6: MSE of Distance Over Time

4.2 Kalman Filter Result

The original result of the Kalman filter data is still in the ECEF format. The Kalman filter calculation result is then plotted into two different graphs: latitude

and longitude comparison between ADS-B data and estimated data(Kalman filter result) and the graph of the comparison altitude over time between ADS-B data and estimated data. To make the data point interval more visible, the graph only displays 1 data point every 50 available data points for both estimated data and the original ADS-B data. Both data was plotted using scatter mode. To make it easier to visualize the difference between ADS-B data and estimated data, the scatter plot's size is increased. As the plot results still use the ECEF form, the latitude, longitude, and altitude are still in vector form using the meter unit.

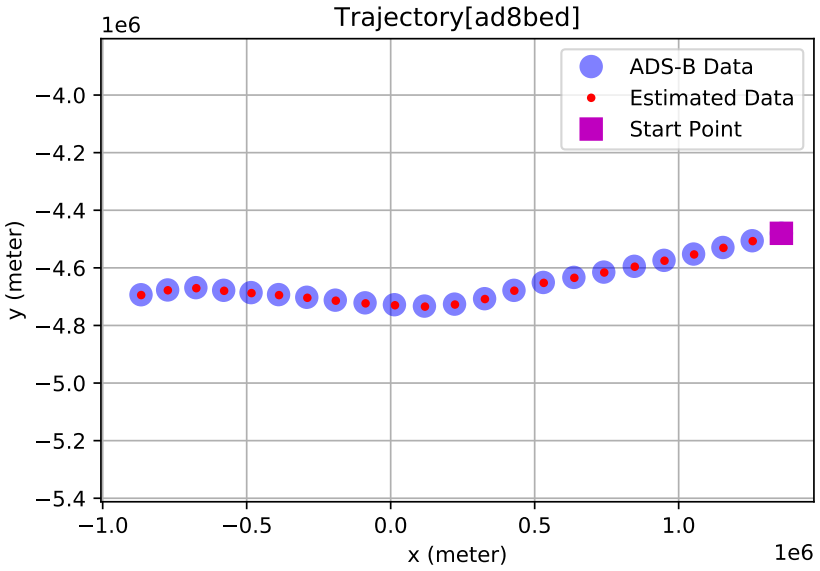

FIGURE 4.7: Latitude and Longitude Trajectory

As seen in figure 4.8 and 4.8, there is a small deviation between the estimated data and ADS-B data. To measure those position difference, MSE calculation is conducted. The result of the MSE calculation are displayed on table 4.3.

The result of MSE in the original coordinate of the latitude and longitude form also needs to be calculated. This was aimed to calculating actual position difference, which was not influenced by other factors. To accomplish that, this research reconverting the coordinate using the *pymap3d* library. After the reconversion, both coordinates can be plotted using *basemap* library so that the visualization of the aircraft trajectory is straightforward. The result of the *basemap* plot and MSE result can be seen in the figure 4.9 and table 4.4.

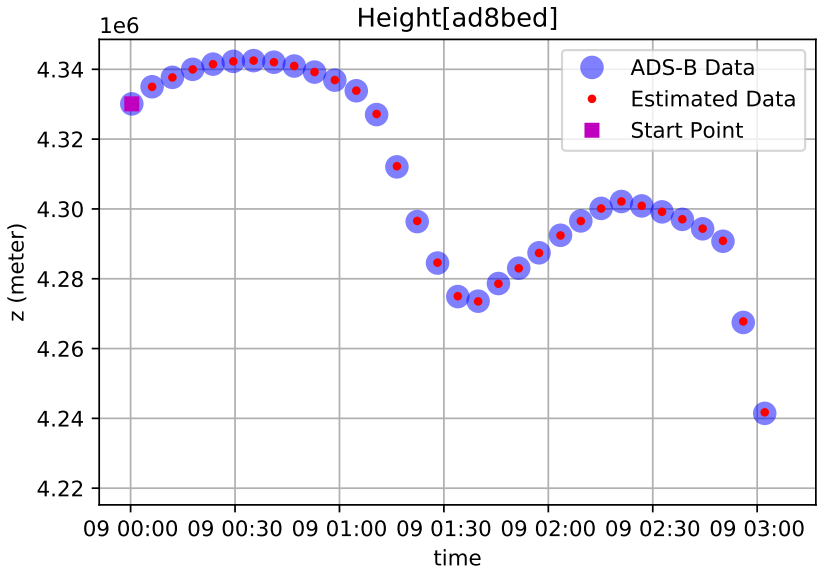

FIGURE 4.8: Altitude Trajectory

MSE of Distance	MSE (m^2)
Latitude of Estimated ECEF and ADS-B ECEF	2.55×10^6
Longitude of Estimated ECEF and ADS-B ECEF	2.35×10^6
Altitude of Estimated ECEF and ADS-B ECEF	2.80×10^3

TABLE 4.3: MSE of ECEF Coordinate

MSE of Distance	MSE (degree)
Latitude of Estimated and ADS-B	9.24×10^{-5}
Longitude of Estimated and ADS-B	3.42×10^{-4}

TABLE 4.4: MSE of Earth Coordinate

To make it easier to assess each data's actual difference, this research implemented another MSE calculation for using the distance using the meter unit. The

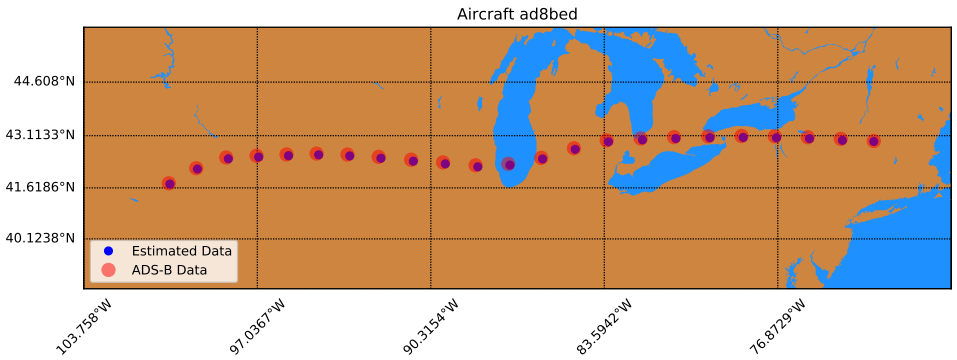

FIGURE 4.9: Aircraft Trajectory

distance between coordinates in each data point can be calculated using *geopy* library and using the *vincenty* command. The result of MSE distance using meter unit can be seen in the tabel 4.5.

MSE of Distance	MSE (m^2)
Difference Between Estimated and ADS-B Data	1.10×10^5

TABLE 4.5: MSE of Earth Coordinate in m^2

It needs to remember that, theoretically, Kalman filter is always assumed to be more accurate. This is because, in the Kalman filter calculation, the noise or inaccuracy indicated by the quality indicator of NIC is minimized. In this case study, most of the generated NIC values were in NIC 8 and NIC 9. So, if most of the NIC value is 8 and 9, the resulting MSE is 1.10×10^5. This research used other levels of NIC to simulate the aircraft's trajectory if another quality indicator value is applied. There are three different values that were used, which are 10 m, 45 m, and 30000 m. The result of the comparison between normal fluctuated noise and the 3 other constant noise can be seen in the figure 4.10, 4.11, and 4.12, and

then table 4.6 is displaying the result as normal case, case 1, case 2, and case 3 for the normal MSE, 10 m MSE, 45 m MSE and 30000 m MSE respectively.

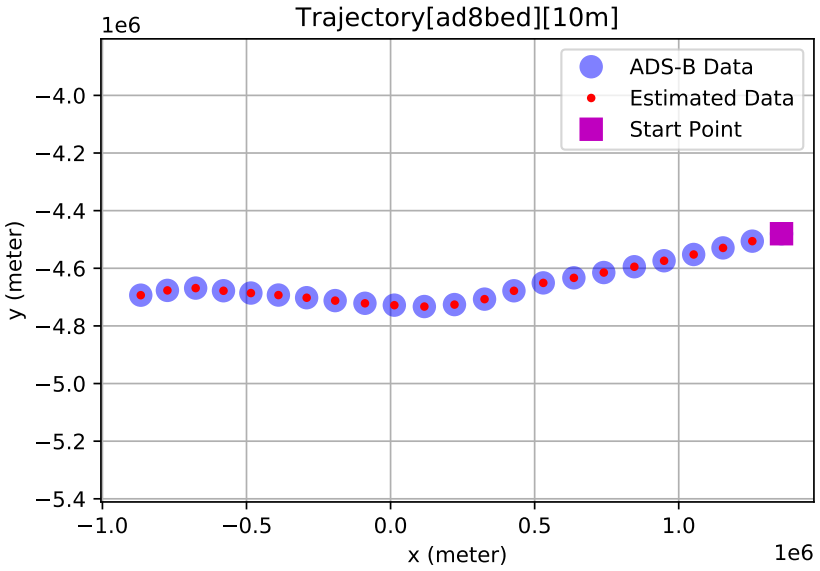

FIGURE 4.10: 10 meter Inaccuracy

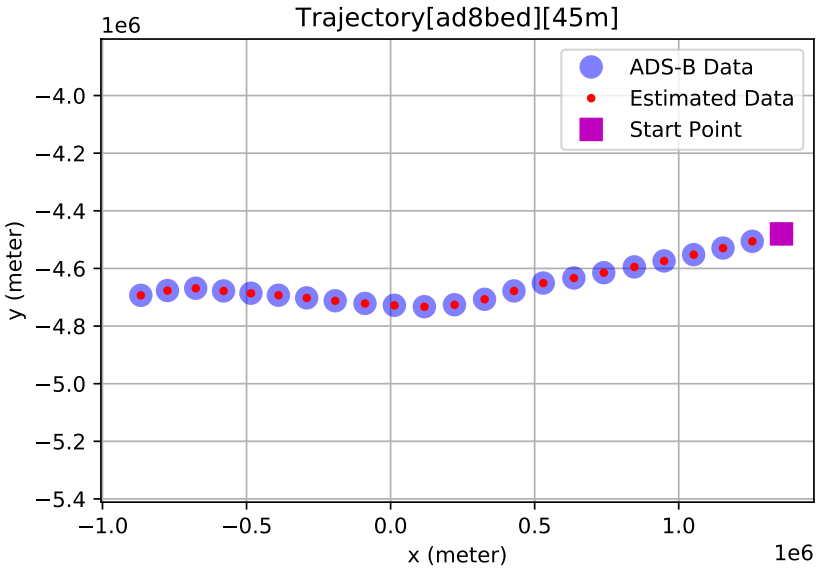

FIGURE 4.11: 45 meter Inaccuracy

In the Kalman filter calculation, determining the aircraft's position between the interruption period is not straight forward as the Interpolation method. To obtain

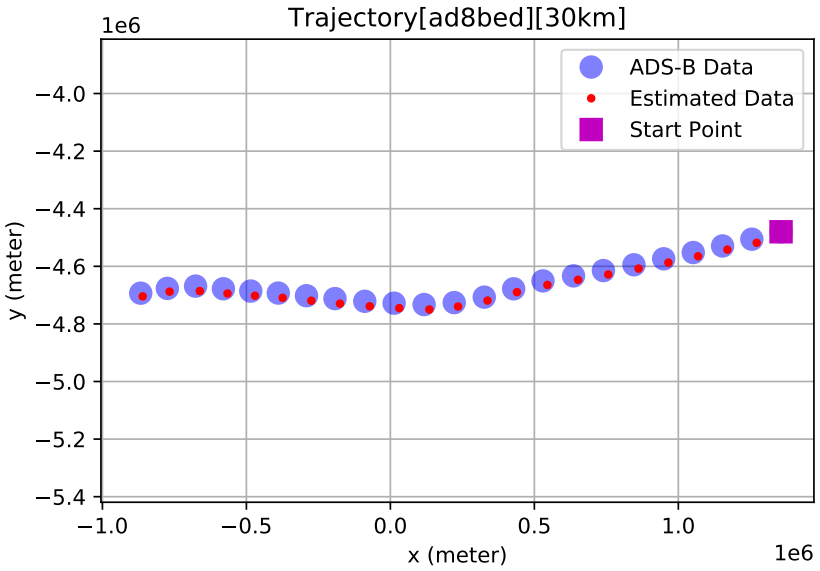

FIGURE 4.12: 30000 meter Inaccuracy

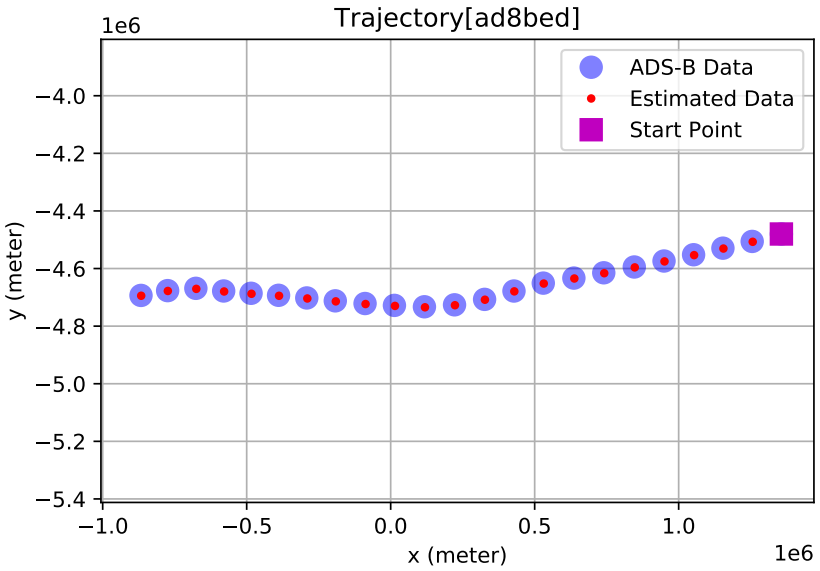

FIGURE 4.13: Normal Inaccuracy

this information, the delta time in calculation needs to be changed manually. For example, in the previous case study, the average interruption period is 10 seconds, and now the FPR needs aircraft position data if the interruption time is 5 seconds. To accomplish that, the *dt* in Kalman filter calculation needs to change manually

Based on ECEF Axis	Normal case MSE(m^2)	Case 1 MSE(m^2)	Case 2 MSE(m^2)	Case 3 MSE(m^2)
X	2.55×10^6	5.42×10^3	6.18×10^4	2.29×10^8
Y	2.35×10^6	4.52×10^3	5.49×10^4	2.12×10^8
Z	8.11×10^4	4.14×10^2	2.80×10^3	4.79×10^6

TABLE 4.6: MSE Comparison Between Difference Noise

to 5. The result of this method can be seen in figure 4.14.

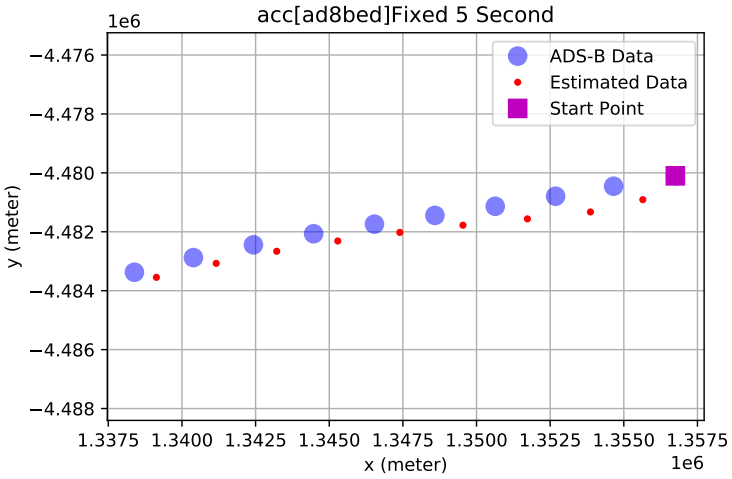

FIGURE 4.14: Fixed 5 Second Delta Time

This research also exploring ten differents aircraft trajectory so that the average of MSE can be obtained. Each MSE of ECEF component was calculated. Then the distance between each coordinate was also calculated the result of the calculation can be seen in table 4.7. However, it is important to mention that the aircraft a4e108, a009a4, ac9ff2, a732d9, a05a01, and a65c29 have a high intermission period, which ranges from 410 to 1040 second. This large intermission period may occur because the aircraft fly in the area that did not have any ADS-B receiver coverage.

4.3 Result Comparison

In this section, the MSE between each method and original ADS-B data will be compared. The comparison is aim to showing the magnitude of the MSE for each method. The comparison can be seen from table 4.8 to table 4.10

A/C Code	x (m^2)	y (m^2)	z (m^2)	Distance (m^2)
ad8bed	2.55×10^6	2.35×10^6	8.11×10^4	1.10×10^5
a4e108	4.14×10^6	2.64×10^6	6.71×10^5	4.26×10^6
a009a4	9.63×10^6	2.51×10^6	3.89×10^4	1.08×10^7
a01c37	1.66×10^6	2.42×10^6	1.06×10^4	1.04×10^5
ac9ff2	8.09×10^7	8.86×10^6	2.34×10^4	3.81×10^8
a732d9	1.29×10^7	1.62×10^6	6.32×10^5	1.50×10^7
a05a01	1.34×10^7	2.64×10^6	1.90×10^5	1.46×10^7
a65c29	1.12×10^7	3.26×10^6	1.08×10^4	1.30×10^7
a149f1	6.53×10^6	5.49×10^6	4.63×10^3	8.51×10^6
a3a8ef	3.41×10^6	3.69×10^6	1.45×10^4	2.75×10^5
Average	1.46×10^7	3.54×10^6	1.67×10^5	4.47×10^7

TABLE 4.7: Other Aircrafts MSE

MSE of Latitude	Kalman (degree)	Linear (degree)	Spline (degree)	PCHIP (degree)
ADS-B	9.24×10^{-5}	0	6.12×10^{-26}	0

TABLE 4.8: MSE of Latitude Result Comparison

MSE of Longitude	Kalman (degree)	Linear (degree)	Spline (degree)	PCHIP (degree)
ADS-B	3.42×10^{-4}	0	2.73×10^{-25}	0

TABLE 4.9: MSE of Longitude Result Comparison

MSE of Distance	Kalman (m^2)	Linear (m^2)	Spline (m^2)	PCHIP (m^2)
ADS-B	1.10×10^5	0	2.74×10^{-18}	0

TABLE 4.10: Distance Difference Comparison in m^2

CHAPTER 5
SUMMARY, CONCLUSION, AND RECOMMENDATION

5.1 Summary

In summary, FPR or flight path reconstruction by using the interpolation method and Kalman filter method have been done. In this research, both approaches were made using Python programming language. It is mainly using the *scipy* library for the interpolation and *filterpy* library for the Kalman filter calculation. The analysis that was conducted in this research is primarily based on the MSE comparison between original ADS-B data and result from both calculation methods. To visualize the result of interpolation and Kalman filter trajectory, this research mainly use the *matplotlib* library's in Python programming language.

5.2 Conclusion

In conclusion, the result of interpolation result virtually did not have any error value, or in other words, the original ADS-B data is similar to the interpolation result. This occurred due to the small delta time between timestamp or small interruption period. As a result of that, the interpolation MSE of distance between coordinates were very small if compared with the original ADS-B data. The exact value of the distance MSE between aircraft trajectory is 0 m^2 for linear interpolation, 2.74×10^{-18} m^2 for spline interpolation, and 0 m^2 for PCHIP interpolation which means there is virtually no MSE because of the small maximum MSE value. However, those calculation results neglected the inaccuracy/noise of the ADS-B system. As a result, the interpolation method became inaccurate.

For the result of Kalman filter, theoretically this method result is the most accurate trajectory. With the assumption of most of the NIC Value is 8 and 9, the MSE between the original data and Kalman filter data is 1.10×10^5 m^2. For various inaccuracy values of 10, 45 ,and 30000 meters the MSE became 5.42×10^3 m^2, 4.52×10^3 m^2, and 4.14×10^2 m^2 for x, y, and z of 10 meters inaccuracy, 6.18×10^4 m^2, 5.49×10^4 m^2, and 2.80e+03 m^2 for x, y, and z of 45 meters inaccuracy and 2.29×10^8 m^2, 2.12×10^8 m^2, and 4.79×10^6 m^2 for x, y, and z of 30000 meters inaccuracy. Then for producing aircraft trajectory between the interruption period, interpolation method was show to be much easier because when using Kalman filter the tools user needs to input the desired delta time manually.

However, it is important to remember that the result of the interpolation calculation methods has 0 or very small MSE value, it does not mean that it was accurate because, in this calculation, the inaccuracy/noise of the ADS-B system did not taken into consideration. While the MSE in Kalman filter did not consider as an error but as a trajectory deviation after noise/inaccuracy of ADS-B data is minimized. In other words, the noise/inaccuracy of the ADS-B data resulting an MSE of 1.10×10^5 m^2 (for aircraft ad8bed).

5.3 Recommendation

As for the recommendation for future research in the same field, this research recommends using an ADS-B data set that has the original quality indicator value and not using a randomly generated version. This is aimed for producing a result that is not influenced by human-generated data. Then, the next recommendation is to develop a new python library for Kalman filter calculation. So, it is not required to perform a data adjustment before calculation the Kalman filter.

References

ADS-B alat navigasi penerbangan. (2016). Retrieved from http://kelembagaan.ristekdikti.go.id/index.php/2016/12/09/ads-b-alat-navigasi-penerbangan/

Becker (www.kalmanfilter.net), A. (2018a). *Online Kalman Filter Tutorial.* Retrieved 2020-08-06, from https://www.kalmanfilter.net/%7D (Library Catalog: www.kalmanfilter.net)

Becker (www.kalmanfilter.net), A. (2018b). *Online Kalman Filter Tutorial.* Retrieved 2020-08-05, from https://www.kalmanfilter.net/%7D (Library Catalog: www.kalmanfilter.net)

Calderwood, D. (2016, March). *LAA and BMAA to approve future ADS-B installations.* Retrieved 2020-08-05, from https://www.flyer.co.uk/laa-and-bmaa-to-approve-future-ads-b-installations/ (Library Catalog: www.flyer.co.uk)

Civil Aviation Order 108.56 Instrument 2007. (2007). Retrieved 2020-08-05, from https://www.legislation.gov.au/Details/F2007L04874/Html/Text

ECEF(Earth-Centered Earth-Fixed Frame) Coordinate. (2016, March). Retrieved 2020-08-05, from https://grandstayner.tistory.com/entry/ECEFEarthCentered-EarthFixed-Frame-Coordinate (Library Catalog: grandstayner.tistory.com Section: IT)

Evans, R. J., Goodwin, G. C., Feik, R. A., Martin, C., & Lozano-Leal, R. (1985, July). Aircraft Flight Data Compatibility Checking Using Maximum Likelihood and Extended Kalman Filter Estimation. *IFAC Proceedings Volumes*, *18*(5), 487–492. Retrieved 2020-08-05, from http://www.sciencedirect.com/science/article/pii/S1474667017606074 doi: 10.1016/S1474-6670(17)60607-4

Global Positioning System (GPS) in Aviation | Aircraft Systems. (2017). Retrieved 2020-08-05, from https://www.aircraftsystemstech.com/2017/

05/global-positioning-system-gps.html

Göttlicher, C., & Holzapfel, F. (2016). FLIGHT PATH RECONSTRUCTION FOR AN UNMANNED AERIAL VEHICLE USING LOW-COST SENSORS. , 10.

How Does FIS-B (Flight Information System Broadcast) Work? (2019). Retrieved 2020-08-05, from https://www.thebalancecareers.com/what-is-fis-b-282560

Implementasi Automatic Dependent Surveillance Broadcast (ADS-B) di Indonesia | Nurhayati | WARTA ARDHIA. (2014). Retrieved 2020-08-05, from https://wartaardhia.com/index.php/wartaardhia/article/view/128/131

Ins and Outs [template]. (2020). Retrieved 2020-08-05, from https://www.faa.gov/nextgen/equipadsb/capabilities/ins_outs/ (Last Modified: 2020-01-02T09:13:02-0500 Library Catalog: www.faa.gov)

Introduction to Kalman Filter and Its Applications | IntechOpen. (2018). Retrieved 2020-08-05, from https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications

KREYSZIG, E. (2010). *advanced engineering mathematics erwin kreyszig.pdf* (10th ed.). John Wiley & Sons.

PAPRUsersGuide. (2020). Retrieved from https://adsbperformance.faa.gov/PAPRUsersGuide.pdf

pchip function | R Documentation. (2020). Retrieved 2020-08-05, from https://www.rdocumentation.org/packages/pracma/versions/1.9.9/topics/pchip

Preprocessing: Calculate the Distance Between Longitude and Latitude Points with a Function | by The Data Detective | Towards Data Science. (2019). Retrieved 2020-08-05, from https://towardsdatascience.com/preprocessing-calculate-the-distance-between-longitude-and-latitude-points-with-a-function-8f748a2eab88

RADAR - Introduction of RADAR Systems, Types and Applications. (2013, September). Retrieved 2020-08-05, from https://www.elprocus.com/radar-basics-types-and-applications/ (Library Catalog: www.elprocus.com)

Tesi, S., & Pleninger, S. (2017). Analysis of Quality Indicators in ADS-B Messages.

MAD - Magazine of Aviation Development, *5*(3), 6–12. Retrieved 2020-08-06, from https://ojs.cvut.cz/ojs/index.php/mad/article/view/4381%7D (Number: 3) doi: 10.14311/MAD.2017.03.01

Types of Interpolation - Advantages and Disadvantages. (2019). Retrieved 2020-08-05, from http://www.gisresources.com/types-interpolation-methods_3/ (Library Catalog: www.gisresources.com)

United States latitude and longitude. (2015). Retrieved 2020-08-05, from https://latitudelongitude.org/us/%7D

What is radar (radio detection and ranging)? - Definition from WhatIs.com. (2005). Retrieved 2020-08-05, from https://searchmobilecomputing.techtarget.com/definition/radar (Library Catalog: searchmobilecomputing.techtarget.com)

Appendices

Appendix A: Python Codes

```
import pandas as pd
import numpy as np
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
from filterpy.kalman import KalmanFilter
import pymap3d as pm
from geopy.distance import vincenty


"""
Inputing Data
"""
#Reading Data
df = pd.read_csv('US_ad8bed.csv')
df['time'] = pd.to_datetime(df['time'],unit='s')
df['heading'] = np.deg2rad(df['heading'])

#Only include Aircraft on Air
#df2 = df.drop(df.index[805:2849]) #US_a4e108
df2 = df.drop(df.index[1122:3330]) #US_bed
df3 = df2[df2['onground'].isin([False])]



"""
Converting Magnitude to Cartesian
"""
```

```
#Convert Geodetic to ECEF
lat = (df3['lat'])
lon = (df3['lon'])
h = df3['geoaltitude']

x_coor,y_coor,z_coor = [], [], []
x_coor,y_coor,z_coor = pm.geodetic2ecef(lat,lon,h)

coord_value = {"X_Coordinate" : x_coor,
               "Y_Coordinate" : y_coor, "Z_Coordinate" : z_coor}
Cart_coord = pd.DataFrame(coord_value)



#Converting velocity

Vx = df3['velocity'] * np.cos(df3['heading']) #Speed in X axis (m/s)
Vy = df3['velocity'] * np.sin(df3['heading']) #Speed in Y axis (m/s)
Vz = df3['vertrate'] #Speed in Z axis (m/s)

#Defining the acceleartion

time_diff = (df3['time'] - df3['time'].shift())
time_diff = pd.Series(pd.to_timedelta(time_diff))
time_diff = time_diff.dt.total_seconds() #time difference in second
df3['time_difference'] = time_diff #declaring new data set in the data frame

Ac = (df3['velocity'] - df3['velocity'].shift()) /
df3['time_difference'] #Acceleration scalar (m/s^2)
Ax = Ac * np.cos(df3['heading'])  #Acceleration in X axis (m/s^2)
Ay = Ac * np.sin(df3['heading'])  #Acceleration in Y axis (m/s^2)
Az = ((df3['vertrate'] - df3['vertrate'].shift()) /
      df3['time_difference']) #Acceleration in z axis (m/s^2)
```

```
61
62
63
64  #Generate Dummy Data for Measurement Uncertainty
65  HACC = np.array([20, 37040, 14816, 7408, 3704, 1111.2,
66                   926 ,370.4 ,185.2 ,75 ,25, 7.5]) #Horizonta Magnitude
67  VAAC = np.array([20, 20, 20 , 20, 20, 20,
68                   20 ,20 ,20 ,112 ,37.5, 11])
69  new_data = len(df3['icao24'])
70  df3['HACC'] = np.random.choice(HACC, new_data,
71     p= [0.0239,0.0001, 0.0001, 0.0007,0.0001 ,
72         0.0001, 0.0001, 0.0001, 0.0008, 0.4290,
73         0.5057, 0.0393]) #For Horizontal
74
75  df3['VACC'] = np.random.choice(VAAC, new_data,
76     p= [0.0239,0.0001, 0.0001, 0.0007,0.0001 ,
77         0.0001, 0.0001, 0.0001, 0.0008, 0.4290, 0.5057, 0.0393]) #For Vertical
78
79
80
81  """
82  Kalman Filter(with acceleration)
83  """
84  #Compiling Kalman Filter Data set
85  all_value = {'time' : df3['time'], 'X_position' : Cart_coord['X_Coordinate'],
86               'Y_position' : Cart_coord['Y_Coordinate'],
87               'Z_position' : Cart_coord['Z_Coordinate'],
88               'Velocity' : df3['velocity'],
89               'velocity_x' : Vx, 'velocity_y' : Vy,
90               'velocity_z' : Vz, 'Acceleration' : Ac, 'acceleration_x' : Ax,
91               'acceleration_y' : Ay, 'acceleration_z' : Az,
92               'Delta_time' : df3['time_difference'], 'HACC' : df3['HACC'],
93               'VACC' : df3['VACC'], 'Latitude' : df3['lat'],
```

```
94                'Longitude' : df3['lon']}
95  df_kal = pd.DataFrame(all_value)
96
97  #Defining Kalman Input
98  Measurement = df_kal[['X_position', 'Y_position', 'velocity_x' ,
99                        'velocity_y']].values.tolist()
100 InputVariable = df_kal[['acceleration_x', 'acceleration_y']].values.tolist()
101
102 inp = InputVariable[0] #Initial Input Variable
103
104 init = np.array([Measurement[0]]). T #Initial Position
105
106 dt = df_kal['Delta_time'].iloc[0] #Initial delta time
107
108 varx = df_kal['HACC'].iloc[0]
109
110 vary = df_kal['VACC'].iloc[0]
111
112
113
114 #Defining Kalman Variable
115 f = KalmanFilter(dim_x=4, dim_z=4, dim_u = 2) #Setup kalman filter dimension
116
117 f.x = init #Initial State matrix
118
119 f.u = inp #Input/control variable
120
121
122 f.B = np.array([[(dt**2)/2, 0],
123                 [0, (dt**2)/2],
124                 [dt, 0],
125                 [0, dt]]) #Input/control transition matrix
126
```

```
127 f.F = np.array([[1, 0, dt, 0],
128                 [0, 1, 0, dt],
129                 [0, 0, 1, 0],
130                 [0, 0, 0, 1]])  #State transition matrix
131
132 f.P = np.array([[400, 0, 0, 0],
133                 [0, 400, 0, 0],
134                 [0, 0, 4.25, 4.25],
135                 [0, 0, 4.25, 4.25]]) #Covariance matrix
136
137 f.R = np.array([[varx**2, 0, 0, 0],
138                 [0, varx**2, 0, 0],
139                 [0, 0, 4.25, 0],
140                 [0, 0, 0, 4.25]]) #Measurement Covariance matrix
141
142 #f.R = np.array([[30000, 0, 0, 0],
143 #                [0, 30000, 0, 0],
144 #                [0, 0, 4.25, 0],
145 #                [0, 0, 0, 4.25]]) #Measurement Covariance matrix for 30000 m
146
147 #f.R = np.array([[10, 0, 0, 0],
148 #                [0, 10, 0, 0],
149 #                [0, 0, 4.25, 0],
150 #                [0, 0, 0, 4.25]]) #Measurement Covariance matrix for 10 m
151
152 #f.R = np.array([[45, 0, 0, 0],
153 #                [0, 45, 0, 0],
154 #                [0, 0, 4.25, 0],
155 #                [0, 0, 0, 4.25]]) #Measurement Covariance matrix for 45 m
156
157
158 f.H = np.eye(4, 4)
159
```

```
160 count = len(df_kal['Delta_time'])
161 xs, ys, vxs, vys = [], [], [], []
162 pxs, pys, pvx, pvy = [], [], [], []
163
164 #Initialize Kalman Filter Calculation
165 for i in range(count):
166     dt = df_kal['Delta_time'].iloc[i]
167     varx = df_kal['HACC'].iloc[i]
168     vary = df_kal['VACC'].iloc[i]
169     inp = InputVariable[i]
170     pos = Measurement[i]
171     z = pos
172     f.predict (inp)
173     f.update (z)
174     xs.append (f.x[0,0])
175     ys.append (f.x[1,0])
176     vxs.append (f.x[2,0])
177     vys.append (f.x[3,0])
178     pxs.append (pos[0])
179     pys.append(pos[1])
180
181
182
183 #Error Calculation
184 Rx=  ((np.array(pxs) - np.array(xs))**2)
185 SERx = (np.sum(Rx))/count
186 SERx1 = '{:.2e}'.format(SERx)
187 print("X(lat) MSE = ", SERx1)
188
189 Ry=  ((np.array(pys) - np.array(ys))**2)
190 SERy = (np.sum(Ry))/count
191 SERy1 = '{:.2e}'.format(SERy)
192 print("Y(lon) MSE = ", SERy1)
```

```
193
194 Rxy = np.sqrt(SERy**2 + SERx**2)
195 Rxy1 =  '{:.2e}'.format(Rxy)
196
197
198 #Plotting
199 cbn = plt.figure(1)
200 plt.plot(pxs, pys, 'bo', markevery = 50,markersize = 10,
201          alpha=0.5, label = 'ADS-B Data')
202 plt.plot(xs, ys, 'r.', markevery = 50,  label = 'Estimated Data')
203 plt.plot(pxs[0], pys[0], 'ms', markersize = 10, label = 'Start Point')
204 plt.axis('equal')
205 plt.xlabel('x (meter)')
206 plt.ylabel('y (meter)')
207 plt.title('acc[ad8bed]')
208 plt.grid()
209 plt.legend()
210 plt.ticklabel_format(axis="x", style="sci", scilimits=(0,0))
211 plt.ticklabel_format(axis="y", style="sci", scilimits=(0,0))
212 plt.show()
213
214
215 """
216 Kalman Filter(z axis)
217 """
218 #Compiling Kalman Filter Data set
219 all_value2 = {'time' : df3['time'],
220               'X_position' : Cart_coord['X_Coordinate'],
221              'Y_position' : Cart_coord['Y_Coordinate'],
222              'Z_position' : Cart_coord['Z_Coordinate'],
223              'Velocity' : df3['velocity'],
224              'velocity_x' : Vx, 'velocity_y' : Vy,
225              'velocity_z' : Vz, 'Acceleration' : Ac, 'acceleration_x' : Ax,
```

```
                'acceleration_y' : Ay, 'acceleration_z' : Az,
                'Delta_time' : df3['time_difference'], 'HACC' : df3['HACC'],
                'VACC' : df3['VACC'], 'Latitude' : df3['lat'],
                'Longitude' : df3['lon']}
df_kal3 = pd.DataFrame(all_value2)

#Defining Kalman Input
Measurement2 = df_kal3[['Z_position','velocity_z']].values.tolist()
InputVariable2 = df_kal3[['acceleration_z']].values.tolist()


inp2 = InputVariable2[0]
init2 = np.array([Measurement2[0]]). T
dt = df_kal3['Delta_time'].iloc[0]

varx = df_kal3['HACC'].iloc[0]

vary = df_kal3['VACC'].iloc[0]


##Defining Kalman Variable
f2 = KalmanFilter(dim_x=2, dim_z=2, dim_u = 1) #Setup kalman filter dimension

f2.x = init2 #Initial State matrix

f2.u = inp2 #Input/control variable
#
f2.B = np.array([[(dt**2)/2],
                 [dt]])    #Input/control transition matrix

f2.F = np.array([[1, dt],
                 [0, 1]])  #State transition matrix

```

```
259 f2.P = np.array([[225, 0],
260                  [0, 4.25]])
261
262
263 #f2.R = np.array([[30000, 0],
264 #                  [0, 4.25]])
265
266 #f2.R = np.array([[10, 0],
267 #                  [0, 4.25]])
268
269 #f2.R = np.array([[45, 0],
270 #                  [0, 4.25]])
271
272 f2.R = np.array([[vary**2, 0],
273                  [0, 4.25]])
274
275 f2.H = np.eye(2, 2)
276
277 count = len(df_kal3['Delta_time'])
278 zs, vzs = [], []
279 pzs, pvz = [], []
280
281 #Initialize Kalman Filter Calculation
282 for i in range(count):
283     pos2 = Measurement2[i]
284     dt = df_kal3['Delta_time'].iloc[i]
285     varx = df_kal3['HACC'].iloc[i]
286     vary = df_kal3['VACC'].iloc[i]
287     z = pos2
288     f2.predict (inp2)
289     f2.update (z)
290     zs.append (f2.x[0,0])
291     vzs.append (f2.x[1,0])
```

```
292     pzs.append (pos2[0])
293     pvz.append(pos2[1])
294 
295 #Error Calculation
296 Rz=  ((np.array(pzs) - np.array(zs))**2)
297 SERz = (np.sum(Rz))/count
298 SERz1 = '{:.2e}'.format(SERz)
299 print("Z(height) MSE = ", SERz1)
300 
301 #Plotting
302 cbn = plt.figure(1)
303 plt.plot(df_kal3['time'], pzs, 'bo', markevery = 50,markersize = 10,
304          alpha=0.5, label = 'ADS-B Data')
305 plt.plot(df_kal3['time'], zs, 'r.',
306          markevery = 50,  label = 'Estimated Data')
307 plt.plot(df_kal3['time'].iloc[0], pzs[0], 'ms', markersize = 10,
308          label = 'Start Point')
309 plt.axis('equal')
310 plt.xlabel('x (meter)')
311 plt.ylabel('y (meter)')
312 plt.title('acc[ad8bed]')
313 plt.grid()
314 plt.legend()
315 plt.ticklabel_format(axis="x", style="sci", scilimits=(0,0))
316 plt.ticklabel_format(axis="y", style="sci", scilimits=(0,0))
317 plt.show()
318 
319 """
320 Convert ECEF to Geodetic
321 """
322 new_lat, new_lon, new_h = [], [], []
323 new_lat, new_lon, new_h = pm.ecef2geodetic(xs, ys, zs)
324 
```

```
325
326 comp_pos = {"ADSB_lat" : df_kal3['Latitude'] ,
327              "ADSB_lon" : df_kal3['Longitude'],
328              "KF_lat" : new_lat, "KF_lon" : new_lon}
329 ac_post = pd.DataFrame(comp_pos)
330
331 abn = plt.figure(2)
332 plt.plot( ac_post['ADSB_lon'], ac_post['ADSB_lat'], 'ro',
333          markevery=50,markersize = 10, alpha=0.5, label = "ADS-B Data")
334 plt.plot(ac_post['KF_lon'], ac_post['KF_lat'], 'b.' ,
335          markevery=50 ,label = "Estimated Data")
336 plt.axis('equal')
337 plt.grid()
338 plt.legend()
339 plt.show()
340
341
342
343 """
344 Error Calculation
345 """
346 Rlat=  (ac_post['ADSB_lat'] - ac_post['KF_lat'])**2
347 SERlat = (np.sum(Rlat))/count
348 SERlat1 = '{:.2e}'.format(SERlat)
349 print("Latitude MSE = ", SERlat1)
350
351 Rlon=  (ac_post['ADSB_lon'] - ac_post['KF_lon'])**2
352 SERlon = (np.sum(Rlon))/count
353 SERlon1 = '{:.2e}'.format(SERlon)
354 print("Longitude MSE = ", SERlon1)
355
356 bk = pd.read_csv('InterpolationResult(US_ad8bed).csv')
357 print("MSE of latitude(KalVlin) = ",
```

```
358       '{:.2e}'.format(
359               (np.sum((ac_post['KF_lat'] - bk['lat_lin'])**2)/count)))
360 print("MSE of latitude(KalVspl) = ",
361       ((np.sum((ac_post['KF_lat'] - bk['lat_spl'])**2)/count)))
362 print("MSE of latitude(KalVpch) = ",
363       '{:.2e}'.format((np.sum(
364               (ac_post['KF_lat'] - bk['lat_pch'])**2)/count)))
365 print("MSE of longitude(KalVlin) = ",
366       '{:.2e}'.format(
367               (np.sum((ac_post['KF_lon'] - bk['long_lin'])**2)/count)))
368 print("MSE of longitude(KalVspl) = ",
369       ((np.sum((ac_post['KF_lon'] - bk['long_spl'])**2)/count)))
370 print("MSE of longitude(KalVpch) = ",
371       '{:.2e}'.format(
372               (np.sum((ac_post['KF_lon'] - bk['long_pch'])**2)/count)))
373
374 #Distance Difference Calculation
375
376 alldata = pd.concat([ac_post, bk], axis=1)
377 alldata = alldata.dropna()
378
379
380 def distance_calc_kalVlin (row):
381     start1 = (row['KF_lat'], row['KF_lon'])
382     stop1 = (row['lat_lin'], row['long_lin'])
383
384     return vincenty(start1, stop1).meters
385
386 alldata['distance_kalVlin'] = alldata.apply (lambda row:
387     distance_calc_kalVlin (row), axis=1)
388
389 def distance_calc_kalVspl (row):
390     start2 = (row['KF_lat'], row['KF_lon'])
```

```
391     stop2 = (row['lat_spl'], row['long_spl'])
392 
393     return vincenty(start2, stop2).meters
394 
395 alldata['distance_kalVspl'] = alldata.apply (lambda row:
396     distance_calc_kalVspl (row), axis=1)
397 
398 def distance_calc_kalVpch (row):
399     start3 = (row['KF_lat'], row['KF_lon'])
400     stop3 = (row['lat_pch'], row['long_pch'])
401 
402     return vincenty(start3, stop3).meters
403 
404 alldata['distance_kalVpch'] = alldata.apply (lambda row:
405     distance_calc_kalVpch (row), axis=1)
406 
407 def distance_calc_adsbVkal (row):
408     start4 = (row['KF_lat'], row['KF_lon'])
409     stop4 = (row['ADSB_lat'], row['ADSB_lon'])
410 
411     return vincenty(start4, stop4).meters
412 
413 alldata['distance_adsbVkal'] = alldata.apply (lambda row:
414     distance_calc_adsbVkal (row), axis=1)
415 
416 def distance_calc_adsbVlin (row):
417     start5 = (row['lat_lin'], row['long_lin'])
418     stop5 = (row['ADSB_lat'], row['ADSB_lon'])
419 
420     return vincenty(start5, stop5).meters
421 
422 alldata['distance_adsbVlin'] = alldata.apply (lambda row:
423     distance_calc_adsbVlin (row), axis=1)
```

```
424
425 def distance_calc_adsbVspl (row):
426     start6 = (row['lat_spl'], row['long_spl'])
427     stop6 = (row['ADSB_lat'], row['ADSB_lon'])
428
429     return vincenty(start6, stop6).meters
430
431 alldata['distance_adsbVspl'] = alldata.apply (lambda row:
432     distance_calc_adsbVspl (row), axis=1)
433
434 def distance_calc_adsbVpch (row):
435     start7 = (row['lat_pch'], row['long_pch'])
436     stop7 = (row['ADSB_lat'], row['ADSB_lon'])
437
438     return vincenty(start7, stop7).meters
439
440 alldata['distance_adsbVpch'] = alldata.apply (lambda row:
441     distance_calc_adsbVpch (row), axis=1)
442
443
444 print("MSE of distance(KalVlin) = ",
445       '{:.2e}'.format((np.sum((alldata['distance_kalVlin'])**2)/count)))
446 print("MSE of distance(KalVspl) = ",
447       ((np.sum((alldata['distance_kalVspl'])**2)/count)))
448 print("MSE of distance(KalVpch) = ",
449       '{:.2e}'.format((np.sum((alldata['distance_kalVpch'])**2)/count)))
450 print("MSE of distance(AdsbVKal) = ",
451       '{:.2e}'.format((np.sum((alldata['distance_adsbVkal'])**2)/count)))
452 print("MSE of distance(AdsbVlin) = ",
453       '{:.2e}'.format((np.sum((alldata['distance_adsbVlin'])**2)/count)))
454 print("MSE of distance(AdsbVspl) = ",
455       ((np.sum((alldata['distance_adsbVspl'])**2)/count)))
456 print("MSE of distance(AdsbVpch) = ",
```

```
457         '{:.2e}'.format((np.sum((alldata['distance_adsbVpch'])**2)/count)))
458
459
460 """
461 Basemap
462 """
463
464 #PLOTTING WITH BASEMAP
465 lats1 = ac_post['ADSB_lat'].tolist()
466 lons1 = ac_post['ADSB_lon'].tolist()
467
468 lats2 = ac_post['KF_lat'].tolist()
469 lons2 = ac_post['KF_lon'].tolist()
470
471 # How much to zoom from coordinates (in degrees)
472 zoom_scale = 3
473
474 #Setup the bounding box for the zoom and bounds of the map
475 bbox = [np.min(lats1)-zoom_scale,np.max(lats1)+zoom_scale,
476             np.min(lons1)-zoom_scale,np.max(lons1)+zoom_scale]
477 plt.figure(figsize=(12,6))
478
479 #Define the projection, scale,
480 #the corners of the map, and the resolution.\n",
481 m = Basemap(projection='merc',llcrnrlat=bbox[0],urcrnrlat=bbox[1],
482                 llcrnrlon=bbox[2],
483                 urcrnrlon=bbox[3],lat_ts=10,resolution='i')
484
485 # Draw coastlines and
486 #fill continents and water with color.drawcoastlines()
487 m.fillcontinents(color='peru',lake_color='dodgerblue')
488
489 # draw parallels, meridians, and color boundaries
```

```
m.drawparallels(np.arange(bbox[0],bbox[1],
                          (bbox[1]-bbox[0])/5),labels=[1,0,0,0])
m.drawmeridians(np.arange(bbox[2],bbox[3],
                          (bbox[3]-bbox[2])/5),labels=[0,0,0,1],rotation=45)
m.drawmapboundary(fill_color='dodgerblue')

# build and plot coordinates onto map
x1,y1 = m(lons1,lats1)
x2,y2 = m(lons2, lats2)
m.plot(x2,y2,'bo',color='b', markevery=50 ,label = "Estimated Data")
m.plot(x1,y1,'ro', color='r',
       markevery=50, markersize = 10, alpha=0.5, label = "ADS-B Data")
plt.legend()
plt.title("Aircraft ad8bed"))
plt.show()
```

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 10 18:07:02 2020

@author: yttbk
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from scipy import interpolate
from geopy.distance import vincenty

"""
Inputing Data
"""
#Reading Data
```

```
19 df = pd.read_csv('US_ad8bed.csv')
20 df['time'] = pd.to_datetime(df['time'],unit='s')
21 df['heading'] = np.deg2rad(df['heading'])
22
23 #Only include Aircraft on Air
24 #df2 = df.drop(df.index[805:2849]) #US_a4e108
25 df2 = df.drop(df.index[1122:3330]) #US_bed
26 df3 = df2[df2['onground'].isin([False])]
27
28
29
30
31
32
33 lati = df3['lat']
34 long = df3['lon']
35 time_date = pd.to_datetime(df3['time'])#setting time format
36 time_num = mdates.date2num(time_date)
37
38
39 #Interpolation
40 time_interp_num = np.linspace(min
41                               (time_num),max(time_num),
42                               10000000)#determining Total new point
43 time_interp_num = np.concatenate([time_interp_num,time_num])
44 time_interp_num = np.sort(time_interp_num)
45
46 #convert time date to numerical
47 time_interp_date = mdates.num2date(time_interp_num)
48
49 """
50 Linear
51 """
```

```
52 #Commencing Linear Interpolation
53 lati_interp_lin = interpolate.interp1d(time_num, lati)
54 lati_interp_lin = lati_interp_lin(time_interp_num)
55
56 #Commencing Linear Interpolation
57 long_interp_lin = interpolate.interp1d(time_num, long)
58 long_interp_lin = long_interp_lin(time_interp_num)
59
60
61 """
62 Spline
63 """
64 #Commencing Spline Interpolation
65 lati_interp_spl = interpolate.InterpolatedUnivariateSpline(time_num, lati)
66 lati_interp_spl = lati_interp_spl(time_interp_num)
67
68 #Commencing Spline Interpolation
69 long_interp_spl = interpolate.InterpolatedUnivariateSpline(time_num, long)
70 long_interp_spl = long_interp_spl(time_interp_num)
71
72
73 """
74 PCHIP
75 """
76 #Commencing Linear PCHIP
77 lati_interp_pch = interpolate.pchip(time_num, lati)
78 lati_interp_pch = lati_interp_pch(time_interp_num)
79
80 #Commencing Linear PCHIP
81 long_interp_pch = interpolate.pchip(time_num, long)
82 long_interp_pch = long_interp_pch(time_interp_num)
83
84
```

```
#Sum Square Error
keywords = time_num
all_interp = {'logged_time' : time_interp_num,
              'lat_lin' : lati_interp_lin,
              'lat_spl' : lati_interp_spl, 'lat_pch' : lati_interp_pch,
              'long_lin' : long_interp_lin, 'long_spl' : long_interp_spl,
              'long_pch' : long_interp_pch}
bk1 = pd.DataFrame(all_interp, columns= ['logged_time',
                                         'lat_lin', 'lat_spl',
                                         'lat_pch', 'long_lin', 'long_spl',
                                         'long_pch'])
bk2 = bk1[bk1['logged_time'].isin(keywords)]
bk3 = bk2.drop_duplicates(subset ="logged_time", keep = 'first')

bk3 = bk3.reset_index()
bk3['lat'] = df3['lat']
bk3['lon'] = df3['lon']

#Clculating Distance Difference
def distance_calc_adsbVlin (row):
    start5 = (row['lat_lin'], row['long_lin'])
    stop5 = (row['lat'], row['lon'])

    return vincenty(start5, stop5).meters

bk3['distance_adsbVlin'] = bk3.apply (lambda row:
    distance_calc_adsbVlin (row), axis=1)

def distance_calc_adsbVspl (row):
    start6 = (row['lat_spl'], row['long_spl'])
    stop6 = (row['lat'], row['lon'])

    return vincenty(start6, stop6).meters
```

```
118
119 bk3['distance_adsbVspl'] = bk3.apply (lambda row:
120     distance_calc_adsbVspl (row), axis=1)
121
122 def distance_calc_adsbVpch (row):
123     start7 = (row['lat_pch'], row['long_pch'])
124     stop7 = (row['lat'], row['lon'])
125
126     return vincenty(start7, stop7).meters
127
128 bk3['distance_adsbVpch'] = bk3.apply (lambda row:
129     distance_calc_adsbVpch (row), axis=1)
130
131 e1a = np.array(lati)
132 e1b = np.array(long)
133 e2 = np.array(bk3['lat_spl'])
134 e3 = np.array(bk3['lat_pch'])
135 e4 = np.array(bk3['lat_lin'])
136 e5 = np.array(bk3['long_spl'])
137 e6 = np.array(bk3['long_pch'])
138 e7 = np.array(bk3['long_lin'])
139 count = len(bk3['lat_spl'])
140
141 sse1 = ((e1a - e2) ** 2)
142 tsse1 = np.sum(sse1)/count
143 sse2 = ((e1a - e3) ** 2)
144 tsse2 = np.sum(sse2)/count
145 sse3 = ((e1a - e4) ** 2)
146 tsse3 = np.sum(sse3)/count
147 sse4 = ((e1b - e5) ** 2)
148 tsse4 = np.sum(sse4)/count
149 sse5 = ((e1b - e6) ** 2)
150 tsse5 = np.sum(sse5)/count
```

```
151 sse6 = ((e1b - e7) ** 2)
152 tsse6 = np.sum(sse6)/count
153
154
155 ssem = plt.figure(6)
156 plt.plot(time_date, bk3['distance_adsbVspl'], 'r', label = 'spl')
157 plt.plot(time_date, bk3['distance_adsbVpch'], 'b', label = 'pch')
158 plt.plot(time_date, bk3['distance_adsbVlin'], 'y', label = 'lin')
159 plt.xlabel('Time')
160 plt.ylabel('MSE (meter)')
161 plt.title('MSE Value in Meter')
162 plt.grid()
163 plt.legend()
164 plt.show()
165
166
167 sse = plt.figure(5)
168 plt.plot(time_date, sse1, 'r', label = 'lat spl')
169 plt.plot(time_date, sse2, 'b', label = 'lat pch')
170 plt.plot(time_date, sse3, 'y', label = 'lat lin')
171 plt.plot(time_date, sse4, 'k', label = 'lon spl')
172 plt.plot(time_date, sse5, 'g', label = 'lon pch')
173 plt.plot(time_date, sse6, 'c', label = 'lon lin')
174 plt.xlabel('Time')
175 plt.ylabel('MSE ( $^\circ$ )')
176 plt.title('MSE Value')
177 plt.grid()
178 plt.legend()
179 plt.show()
180
181
182
183 #PLOTING
```

```
"""
Linear
"""
linear = plt.figure(1)
plt.plot(long, lati, 'o', markevery = 50,
         markersize = 10, alpha=0.5, label ='ADS-B Data')
plt.plot(long_interp_lin, lati_interp_lin, 'r-', label = 'Interp Data')
plt.title('Aircraft Movement with Linear Interpolation')
plt.xlabel('Longitude ( $^\circ$ )')
plt.ylabel('Latitude ( $^\circ$ )')
plt.legend()
plt.grid()
plt.axes().set_aspect('equal', 'datalim')
plt.show()



"""
Spline
"""
spl = plt.figure(2)
plt.plot(long, lati, 'o', markevery = 50,
         markersize = 10, alpha=0.5, label ='ADS-B Data')
plt.plot(long_interp_spl, lati_interp_spl, 'r-', label = 'Interp Data')
plt.title('Aircraft Movement with Spline Interpolation')
plt.xlabel('Longitude ( $^\circ$ )')
plt.ylabel('Latitude ( $^\circ$ )')
plt.legend()
plt.grid()
plt.axes().set_aspect('equal', 'datalim')
plt.show()

```

```
217
218
219 """
220 PCHIP
221 """
222 pch = plt.figure(3)
223 plt.plot(long, lati, 'o', markevery = 50,
224          markersize = 10, alpha=0.5, label ='ADS-B Data')
225 plt.plot(long_interp_pch, lati_interp_pch, 'r-', label = 'Interp Data')
226 plt.title('Aircraft Movement with PCHIP Interpolation')
227 plt.xlabel('Longitude ( $^\circ$ )')
228 plt.ylabel('Latitude ( $^\circ$ )')
229 plt.legend()
230 plt.grid()
231 plt.axes().set_aspect('equal', 'datalim')
232 plt.show()
233
234
235
236
237
238 """
239 Combine
240 """
241 cbn = plt.figure(4)
242 plt.plot(long, lati, 'o', markevery = 50,
243          markersize = 10, alpha=0.5, label ='ADS-B Data')
244 plt.plot(long_interp_spl, lati_interp_spl, 'r-', label = 'Interp Spline')
245 plt.plot(long_interp_pch, lati_interp_pch, 'g-', label = 'Interp PCHIP')
246 plt.plot(long_interp_lin, lati_interp_lin, 'k-', label = 'Interp Linear')
247 plt.title('Aircraft Movement with Interpolation')
248 plt.xlabel('Longitude ( $^\circ$ )')
249 plt.ylabel('Latitude ( $^\circ$ )')
```

```
250 plt.legend()
251 plt.grid()
252 plt.axes().set_aspect('equal', 'datalim')
253 plt.show()
```

Turnitin Report

Aircraft Flight Path Reconstruction Based on ADS-B Data Using Kalman Filter

by Yazfan Tabah

Submission date: 20-Sep-2020 06:05AM (UTC+0200)
Submission ID: 1391535109
File name: finalrevision20092020.pdf (1.33M)
Word count: 13115
Character count: 64039

ABSTRACT

Aircraft Flight Path Reconstruction Based on ADS-B Data Using Kalman Filter

by

Yazfan Tabah Tahta Bagaskara

Triwanto Simanjuntak, PhD, Advisor

In this thesis, the Kalman filter algorithm was used to perform flight path reconstruction. The flight path reconstruction was based on the ADS-B data set that has been obtained from opensky-network.org. Other methods, such as interpolation calculation, also can be used to perform the flight path reconstruction. However, in using interpolation methods, the ADS-B data inaccuracy is not taken into account. Since the position information in the ADS-B data has an error that are indicated by NIC (Navigation Integrity Code). So, the result of the flight path reconstruction based on interpolation calculations are theoretically inaccurate. There were 3 interpolation methods that were explored in this research; they were linear interpolation, spline interpolation, and PCHIP interpolation. These methods were used as preliminary tools to construct the flight path and to see how they might diverge from the Kalman Filter's results. To minimize the ADS-B data inaccuracy, this research used the Kalman filter method; in this research, the Kalman filter calculation was based only on the kinematic equation. The Kalman filter results were compared with the original data in order to observe how the inaccuracy could deteriorate the aircraft trajectory.

Keyword: *ADS-B, Interpolation, Kalman Filter, Flight Path, Navigation Integrity Category (NIC)*

CHAPTER 1
INTRODUCTION

1.1 Background

By definition, flight path is a set of corridors that connects one location to another location. In general, we can define the flight path as a route that the aircraft took to get from one point to another point. The corridors that the aircraft used is made by a set of geographical coordinates. These coordinates are usually based on satellite navigational systems or other ground-based radio transmitters navigational system. To monitor the aircraft during flight, we can track it by using a method called flight tracking. Flight tracking is a method of aircraft monitoring used by surveillance users such as an ATC (Air Traffic Controller) to monitor the aircraft during flight by using a surveillance system. The tracking of a flight can be done in real-time or reconstruct by using historical data sets that contain the aircraft condition(i.e., latitude and longitude coordinates). There are several systems that can be used in flight tracking, such as using a radar-based monitoring technique, which uses a principle of a radar radio wave to determine the distance between the aircraft and the radar source to obtain the position of the aircraft. Then there is a satellite-based tracking system that uses GNSS (Global Navigation Satellite System) to determine the aircraft's exact coordinate, which is called ADS-B. The ADS-B (Automatic Dependent Surveillance-Broadcast) satellite-based tracking system is integrated into surveillance technology that monitors aircraft conditions and trajectories. The ADS-B system can track the aircraft in real-time and record all the information that the system received. The recorded data that the ADS-B receives can be used as source data in FPR (Flight Path Reconstruction). FPR results can be used to test the reliability of the data and to check post-flight data. FPR utilizes the aircraft sensors data to determine the position, velocity, and altitude of an aircraft to perform those tasks. In reality, the sensor only consists of

the estimation of the said information. A kinematic model of the aircraft is needed to obtain more accurate information on the aircraft state. The idea was to combine the kinematic model with the complete flight data to recreate the state at an exact timestamp. The reconstruct data then can be used to analyze system identification, dynamic analysis, or control algorithm assessment(Göttlicher & Holzapfel, 2016). By knowing the accurate state of the aircraft at a certain point in time, the data can be used as the tools to help the investigation of an aircraft accident or incident.

1.2 Problem Statement

All the systems used for aircraft tracking have a time when the system needs to update the aircraft information. If we view these exact timestamps, we will see information unavailability. This period is called intermission/interruption periods. For example, in a radar-based tracking system, the intermission period can be caused by the latency between the time radio wave is sent to detect the aircraft and when the radio wave is received after it bounces back after detecting an aircraft. For the ADS-B system, the intermission periods were caused by the time the system needs to update the aircraft information, which occurs because the ADS-B did not continuously update the aircraft information. However, there is a gap between each data transmission. In optimal condition, this period lasts for half a second ($0.5s$). In conclusion, during the intermission periods, every data set collected by a different system has a point where there is data have unavailability. The ADS-B also has inaccuracy when the system is determining the aircraft position. The ADS-B commonly have a quality indicator value that was informing the level of the data accuracy. The quality indicator value can be indicated by several types, which were described in a regulation standard DO-260/A/B. In DO-260/A/B, the quality indicator can be shown as NUC (Navigation Uncertainty Code) or NACp (Navigation Accuracy Code for Position) or NIC (Navigation Integrity Code) or SIL (Surveillance Integrity Level). The value of one of these codes will be describing the accuracy of the ADS-B data. This research used the NIC value, the quality indicator code indicating the accuracy radius of the ADS-B data. The accuracy radius is used for determining the vertical and horizontal position accuracy of the aircraft.

1.3 Research Purpose

To overcome the unavailability during the intermission period and inaccuracy of the aircraft position, a flight path estimation need to be generated. This problem can be resolve by using flight path reconstruction methods or FPR. Flight path reconstruction is a method that uses a historical data set of an aircraft condition to reconstruct the past aircraft trajectory. Flightpath reconstruction can be done by using a mathematical computation.

The result of the flight path reconstruction was expected:

- To be used as a tool in aircraft incident/accident.
- To be used as a post-flight analysis data based on the flight trajectory
- To able to producing an estimated aircraft trajectory that had minimal noise or inaccuracy

1.4 Research Scope

This research only used a historical data set that has been gathered by an ADS-B system. This historical data set was obtained from opensky-network.org. The downloaded data set is in the form of a CSV (Comma-Separated Values) file. This research used the data set from 03 March 2020. Due to the format of the opensky-network.org, the data was separated hourly; because of that and hardware limitation, this research used only the first 10 hours of 03 March 2020 data set, which have a total around 3.097 Gb of data. The reason that the opensky-network.org data set is chosen because the data set offers almost all data parameters required in the calculation: timestamp, position (Latitude, Longitude, and altitude), velocity, vertical rate, and magnetic heading. However, the data set that has been obtained from opensky-network.org did not have a quality indicator value. So, to overcome this, the quality indicator is generated randomly using Python programming language. This research utilized python programming language as a tool for performing the FPR. The FPR that was conducted in this research was based on two methods, which are interpolation calculation and Kalman filter calculation.

1.5 Research Approach

The data analysis for this research is conducted by comparing the result from Kalman filter calculation and interpolation calculation with actual ADS-B data to see if there is a deviation between the original aircraft trajectory and aircraft trajectory based on interpolation and Kalman filter calculation. The python programming language is used as a tool for completing the data analysis. The data comparison is based on the MSE(Mean Square Error) calculation. The MSE calculation was intended to show the data difference between the estimated data (Result of interpolation and Kalman filter calculation) and actual data.

The steps to perform this research are:

1. Data set acquisition
2. Performing data filtration
3. Performing data adjustment so that the data parameters are compatible with one another
4. Performing the interpolation and Kalman filter calculation
5. Analyzing the result of the Kalman filter calculation and interpolation calculation

CHAPTER 2
LITERATURE REVIEW

In general, the ADS-B system can be divided into two categories. They are ADS-B Out and ADS-B In. The ADS-B out is acted as the system's broadcaster, and the ADS-B In is acted as the receiver. As the receiver, the ADS-B In is only receiving the data parameters that the ADS-B out sends, including aircraft velocity, aircraft identification, surface position, airborne position, TCAS, emergency status, and operational status. In the ADS-B system, there is a periodical interval between each data cycle (broadcast and receive). This means that during the interval, an estimated aircraft position between interruption can be made by considering this situation as a regression problem with using the known aircraft condition when before and after the intermission period.

In order to generate those flight paths, a method of flight path generation needs to be chosen. In this research, there are two methods that were used, which were Kalman filter and interpolation. Kalman filter is an algorithm that generates an estimate of previously unknown variables that used a combination of estimation generated by a calculation and observation of measurement over a period of time (*Introduction to Kalman Filter and Its Applications | IntechOpen*, 2018). In this research, the Kalman filter calculation was done by using Python programming language. The Kalman filter calculation is based on the kinematic calculation of the aircraft trajectory. The other method used in this research is interpolation; this is a method of generating an estimation of a previously unknown variable using a known data point from a historical data set(*Types of Interpolation - Advantages and Disadvantages*, 2019). In this research, these can be achieved by using two know data points to generate the aircraft's position during the interruption periods. Both the Kalman filter and the interpolation method had its advantages and disadvantages. The first one is that the interpolation methods have more straightforward calculations because this method is generally only a form of mathematical

approximation calculation. On the other hand, the Kalman filter methods calculation is more complex in comparison with the interpolation calculation, because in this calculation, many factors that did not consider in the interpolation are considered here. Consequently, the interpolation result did not minimize the noise of the system contrarily; one of the main ideas of the Kalman filter calculation is to minimize the noise of the system.

2.1 Flight Tracking

Flight tracking is a method of aircraft monitoring that involves a system used to observe aircraft during flight or on the ground. Flight tracking is essential for ATC (Air Traffic Controller) to managing airspace. The tracking system will help the ATC control aircraft separation, control aircraft flight path, and guide aircraft movement or other navigational needs of the aircraft. The ATC usually used a radar system as a tool to track and controlling aircraft movement. Radar or radio detection and ranging that used in UHF (Ultra High Frequency) or microwave part of the RF (Radio Frequency) spectrum to detect an object position (*What is radar (radio detection and ranging)? - Definition from WhatIs.com*, 2005). In general, Radar work by sending an electromagnetic signal through its transmitter to detect an object. When the signal hits an object, the signal will be bounced back. Then the reflected signal will be detected by the radar receiver, which is then processed to determine the object's geographical position. The process used the time taken by the signal to travel from the Radar source to the object and return back to determining the distance between the Radar and the object. For determining the target's location is measured in angle, from the direction of the maximum amplitude echo signal to the antenna(*RADAR - Introduction of RADAR Systems, Types and Applications*, 2013). Another system that can be used to tracking aircraft is ADS-B or Automatic Dependent Surveillance Broadcast. The ADS-B is a satellite-based monitoring system that uses GNSS (Global Navigation Satellite System) to determining aircraft position. In recent years, ADS-B has been chosen to be the replacement for Radar as the tracking or monitoring system for aircraft. The reason mainly because ADS-B is considered to be more reliable and accurate than Radar.

2.2 Automatic Dependent Surveillance Broadcast

Automatic Dependent Surveillance-Broadcast or ADS-B is a type of monitoring/navigation system used in air transportation. The system's primary purpose is to track aircraft movement that periodically updated every 0.5 seconds in optimal condition, or in other words, the most optimal interruption period is 0.5 seconds (*Implementasi Automatic Dependent Surveillance Broadcast (ADS-B) di Indonesia | Nurhayati | WARTA ARDHIA*, 2014). The system can determine the aircraft's location by using GNSS (Global Navigation Satellite System) such as GPS (Global Positioning System). The system also sends other information that correlates with the aircraft's state, such as aircraft identification number (hex identification number), altitude, heading, and speed. This aircraft information will be transmitted to other aircraft that are also equipped with the ADS-B system and satellite or ground receiver, in which the data will be relayed to ATC (Air Traffic Control).

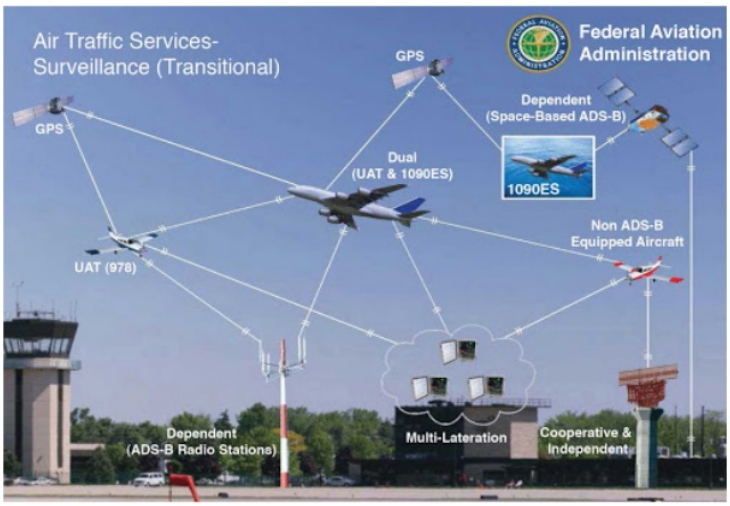

FIGURE 2.1: Schematic of ADS-B system (Calderwood, 2016)

Ideally, the ADS-B is an accurate system that can track an aircraft's trajectory and store that information every 0.5 seconds. However, in real-world implementation, the system cannot perform this perfectly. The system has noise or inaccuracy that can affect the accuracy of the system. Then the system may have a "signal

disturbance" that may affect the data transmission, which can prolong the interruption period. The noise/inaccuracy and prolong interruption time of the ADS-B system may vary between each time step. The magnitude of the noise was identified in the ADS-B message in the form of a quality indicator code. The quality indicator can be defined into several forms of codes; it depends on which version of the DO-260 standard the ADS-B system used. The variation of the quality indicator are:

- NUC = Navigation Uncertainty Code or
- NIC = Navigation Integrity Code
- SIL = Surveillance Integrity Level

Note: In this research, the only quality indicator variant that used is the NIC version. The reason for this is because it is offering much higher error detection, which is up to 30 km, and according to the FAA AC 20-165 mentioned that, NIC is used by surveillance users such as ATC to determine if the data has an acceptable level of integrity. The minimal level requirement of the integrity level is NIC 8. The NIC is also chosen.

There are several factors that can affect the NIC level ("PAPRUsersGuide", 2020):

- Loss of GNSS services
- Antenna masking that caused by aircraft maneuver
- aircraft flying at the edge of ADS-B coverage
- Component/software issue
- Poor ADS-B signal

The NIC is divided into 12 codes, ranging from code 0 to code 11. Each code number is representing a different range of horizontal position accuracy. The following table 2.1 is the description of accuracy for code("PAPRUsersGuide", 2020).

For the system's vertical accuracy, the accuracy magnitude only available for NIC values greater than 8; in other words, the accuracy magnitude available in NIC levels 9 to 11. The magnitude of the vertical NIC can be seen on table 2.2.

NIC Value	Accuracy Value
0	Unknown
1	<37.04 Km
2	<14.81 Km
3	<7.40 Km
4	<3.70 Km
5	<1852 m
6	<926 m
7	<370.4 m
8	<185.2 m
9	<75 m
10	<25 m
11	<7.5 m

TABLE 2.1: NIC Value Horizontal Accuracy Radius

NIC Value	Accuracy Value
9	<112 m
10	<37.5 m
11	<11 m

TABLE 2.2: NIC Value Veritcal Accuracy Radius

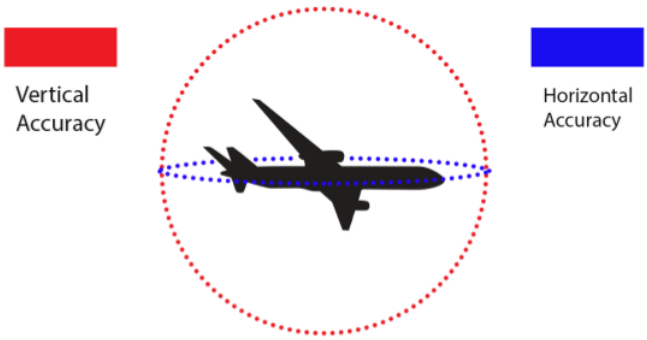

FIGURE 2.2: NIC Horizontal and Vertical Accuracy

2.2.1 Transmission

ADS-B system used mode-s transponder to transmitting/updating aircraft movement that usually operates in 1090 MHz (*Ins and Outs*, 2020). In the US, the FAA (Federal Aviation Administration) want to establish for aircraft that operate below 18,000 feet (5,500 m) transmit the signal 987 MHz. The received ADS-B data can be used alongside the TIS-B (Traffic Information Services Broadcast) and FIS-B (Flight Information Services Broadcast) system. TIS-B is a component of the ADS-B system that provides traffic information for aircraft with ADS-B receiver. TIS-B also has the capability to tracking other aircraft that are not equipped with ADS-B transponder but tracked by other Radar. FIS-B, on the other hand, is a system that provides current airspace information such as graphical weather service and locations of restricted airspace. However, FIS-B information only can be received by aircraft that use 978 MHz UAT (Universal Access Transceiver) (*How Does FIS-B (Flight Information System Broadcast) Work?*, 2019). This process of transmitting and receiving ADS-B information is essentially based on two separate ADS-B systems: ADS-B In and ADS-B Out. ADS-B Out is the part of the ADS-B system that broadcasts aircraft information periodically, such as the aircraft speed, altitude, and location through an onboard transmitter. For the ADS-B In, the system received the other information that will be used for TIS-B and FIS-B system.

2.2.2 ADS-B Implementation

In the time of writing, FAA and EASA regulatory bodies have a mandate that required aircraft to have an ADS-B system installed. Both regulatory bodies want to promote the use of the ADS-B system in other international airspaces, including Asia/Pacific, so they mandate this new regulation. This regulation aimed to obtain the great benefit that the ADS-B system is offering, making the airspace safer and more efficient. In 2010, FAA published a new requirement for aircraft that operate in certain conditions to have ADS-B out technology installed in the aircraft by 1st January 2020. This regulation was based on Title 14 of the US Code of Federal Regulations (14 CFR) sections 91.225 and 91.227. The regulation also required the ADS-B system to operate in US NAS to operate in 1090 and 978 MHz. As of

September 2017, the FAA already deploy an ADS-B ground station to support the ADS-B transmission and collected/relayed the collected ADS-B data.

For Europe, the regulation of ADS-B requirement that fly in the EU (European Union) air space is mandated in Regulation 1028/2014 and Regulation 2017/386. The regulation mandate, aircraft that operate in EU airspace that had criteria of fixed-wing aircraft with maximum take-off mass exceeding 5,700 kg or have maximum cruising true airspeed greater than 250 knots, will be required to have mode-s transponder and ADS-B out system installed by 7th June 2020. The regulation also mandates aircraft that will operate in the trans-Atlantic route to install the ADS-B Out system by 1st January 2020 to fill the FAA requirement fully. For required transmitted ADS-B data is base on EASA AMC 20-24 for ADS-B in Non-Radar Airspace or CS-ACNS for ADS-B out must include:

- Horizontal position of the aircraft (latitude/longitude)
- Barometric altitude of the aircraft (will be the same as for the SSR))
- Quality Indicator
- Aircraft Identification:
  - A unique 24-bit aircraft address
  - Identification of the aircraft
  - Mode A (In the case of CS ACNS for ADS-B Out)
- Emergency Status of the aircraft
- SPI (Special Position Indicator) when selected

In Indonesia, ADS-B research for ADS-B usage begins in 2007 that eventually produces a working ADS-B ground station equipment. The research aims to produce an ADS-B ground station equipment that can control air traffic around the airport and capable of detecting ADS-B out signal within 200 miles. As of 2016, the ADS-B ground station equipment already tested on Soekarno-Hatta international airport, which proving the equipment can detect aircraft within 250 Nm on 29,000 feet ("ADS-B alat navigasi penerbangan", 2016). Indonesia is reported to have

30 ground station that located in Banda Aceh, Matak, Soekarno-Hatta, Pangalan Bun, Cilacap, Palembang, Pekanbaru, Medan, Pontianak, Kintamani Bali, Palu, Ambon, Saumlaki, Alor, Waingapu, Tarakan, Galela, Timika, Kendari, Manado, Natuna, Makasar, Kupang, Sorong, Merauke, Malino Banjarmasin, Balikpapan, Biak, Surabaya, and Semarang (*Implementasi Automatic Dependent Surveillance Broadcast (ADS-B) di Indonesia | Nurhayati | WARTA ARDHIA*, 2014). Those ground stations are integrated into two separate ATM (Air Traffic Management) systems that controlling two different FIR (Flight Information Region), which are JAATS (Jakarta Automated Air Traffic Control System) and MAATS (Makassar Automated Air Traffic Control System). A conference in July 2018, DNP (Direktorat Navigasi Penerbangan) stated that starting in 1st January 2018 aircraft that operate in FL 290 to FL 600 is required to have ADS-B transmitter.

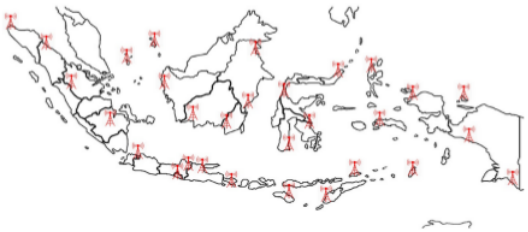

FIGURE 2.3: ADS-B ground station location in Indonesia

2.2.3 ADS-B Benefit

ADS-B is seen as the future of navigation system, which is proven by the United States that already made ADS-B as the primary system for upgrading and modernizing aviation infrastructure and operation. The program is called NextGen National Airspace Strategy that aims to replace the primary surveillance system from Radar to ADS-B. The reason is to improved safety and efficiency.

ADS-B can improve safety by increasing the situational awareness of the pilot. This is caused by the pilot have the ability to see clearer and detailed air traffic

around the aircraft from ADS-B data that the pilot received. The situational awareness of ATC also can be increase because of the same reason. ADS-B can also improve airspace usage efficiency because when the ATM uses the ADS-B system, it can reduce the separation standard between aircraft. The smaller separation can be enabled because ADS-B has a higher accuracy when determining an aircraft position, which leads to ATC can guide/track aircraft more accurately. These improvements were eventually increasing the capacity of an airspace.

2.3 Flight Path Reconstruction

Flightpath reconstruction is a method that uses historical data set to determining aircraft conditions such as aircraft position and speed(Evans, Goodwin, Feik, Martin, & Lozano-Leal, 1985). This was done by using the data from onboard sensors such as inertial and air data sensors. The historical data set can also be obtained by other systems such as ADS-B.

2.4 Interpolation

Interpolation is a data approximation method that will generate a new specific data point between the range of 2 or more known values. For example, a government is commencing a census of their population with ten years interval, however now the government wants to approximate the total population between each year between those two available data. So, those specific data approximation $p_n(x) = f_n$ where pn is interpolation, x is nodes, and f is the mathematical function which then we can use pn to get f value between two point of x.

Note: Interpolation method is only used to approximates value between two points; if the desired value is outside the range, it is called extrapolation.

For solving the interpolation problem, we can use a method of Lagrange interpolation. The method itself has two different approaches: linear interpolation and quadratic interpolation.

For Linear interpolation, the new value of pn will make a straight line between the known $valuex_0$ and x_1. The linear Lagrange polynomial can be express by:

$$p_1 = L_0 f_0 + L_1 f_1 \tag{2.1}$$

Where L is,

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, L_1(x) = \frac{x - x_0}{x_1 - x_0} \tag{2.2}$$

For the quadratic interpolation the new value of pn will made the quadratic line between the known value. The quadratic value of second degree polynomial can be acquired by:

$$p_1 = L_0 f_0 + L_1 f_1 + L_2 f_2 \tag{2.3}$$

Where L is,

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}, L_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \tag{2.4}$$

The interpolation method can be used if the value of pn is expected to unstable or fluctuated, which is called Spline. In general we can say that the generated pn between x_0 and x_1 can be bigger or smaller than the p_1 and p_0.

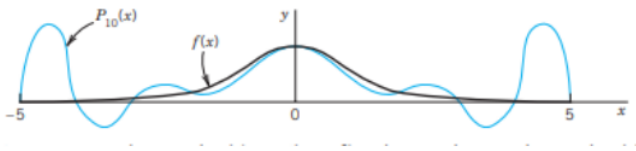

FIGURE 2.4: Spline Interpolation Example (KREYSZIG, 2010)

Based on the figure above, we can see that interpolation can have a more accurate value when it uses the Spline method while approximating fluctuate/unstable data.

Then there is another interpolation method called PCHIP or Piecewise Cubic Hermit interpolation Polynomial. PCHIP interpolation can be interpreted as a

shape-preserving piecewise cubic Hermite polynomial approach that will determine the slop from a function so that the generated value does not overshoot the data values(*pchip function | R Documentation*, 2020). PCHIP interpolation is also the opposite nature of Spline, where the PCHIP will maintaining the monotonicity of x and y value.

2.5 Kalman Filter

Kalman Filter is a type of estimation algorithm that uses a set of equations and a series of measurements observed over time, which have a noise (disturbance) in the measurement that can produce estimates of previously hidden variables. In Kalman Filter, the calculation process can be separated into 3 parts, which are measurement, prediction, and estimation.

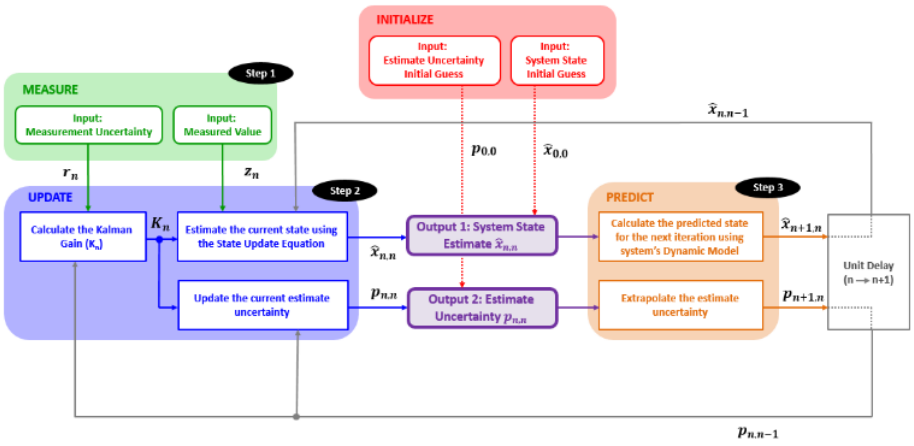

FIGURE 2.5: Kalman Filter Algorithm(Becker (www.kalmanfilter.net), 2018a)

2.5.1 Measurement

First of all, in the Kalman filter, the measurement value that was taken is always considered to have an error. In other words, we can not know the exact value or true value of a measurement. In Kalman filter, every measurement always contains

a random variable resulting in an error in the measurement. These will lead to varying results of the measurement value. However, in Kalman filter we always assume that the measurement always forms a normal distribution. These mean that the true value of the measurement is close to the repeated measurement's mean or average value. Although this condition is not always applied to every system, because in the real world, the measurement tool not always had high accuracy measurement. If the system has high accuracy measurement, then the repeated measurement's mean value is close with the true value, this type of system is called a unbias system. On the other hand, if the system had low accuracy, then the mean value and the true value have an enormous difference; this type of system is called bias system.

In the Kalman filter there 2 types of models that can describe the whole system. First, there is a dynamic model, which mean that the system true value is changing over time, for example, if Kalman filter is used in moving aircraft case. The second one is constant dynamic model, which means the system, in reality, has a constant true value, for example, if we measured the height of a building.

As mentioned previously, in real-world cases, we do not know the true value of a system, and that every measurement always had an error caused by the random variable. The Kalman filter calculation objective is to minimize error or uncertainty in the measurement or referred to as covariance, by doing multiple iterations of the calculation. If the covariance is minimized, then the measured value should be close with the "true value". The initial measurement usually referred to as initial estimate, the estimate is denoted by $\hat{\boldsymbol{x}}_{\boldsymbol{n,n}}$ the first n is for the iteration number of the prediction, and the second n is for the first number of the estimate, so the initial estimate is $\hat{\boldsymbol{x}}_{\boldsymbol{0,0}}$. The covariance is denoted by $\hat{\boldsymbol{P}}_{\boldsymbol{n,n}}$

2.5.2 Prediction

The usage of the Kalman filter can be divided into two models. The first one is one-dimensional Kalman filter. Which are used for single dimensional measurement; for example, the Kalman filter is used for measuring the temperature of an object. The other model is a multi-dimensional Kalman filter used for the system that multi-dimensional measurement. This model was used in this research because a 3-dimensional system was taken into consideration; that is mean the position,

speed, and acceleration in x, y, z axis. The model needs to be determined because the single-dimensional and multi-dimensional calculations used a different set of equations. The significant difference between the two models is that the equation was calculated in matrix form when calculating the multi-dimensional Kalman filter.

As have been mention in the previous section the first measurement of a system in called initial estimation. This initial estimation is consider to be starting point of the Kalman filter calculation. The initial estimate of a system is also have the covariance which both of this value will be calculated in the prediction state. Then all known value is combine in state extrapolation whihch is:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} + \boldsymbol{w}_n \tag{2.5}$$

Where,

$\hat{\boldsymbol{x}}_{n+1,n}$ is the predicted system vector at time step n+1

$\hat{\boldsymbol{x}}_{n,n}$ is the current estimate system vector/initial estimate (State matrix)

$\hat{\boldsymbol{u}}_{n,n}$ is the control/input variable which is a measurable input to the system

$\boldsymbol{w}_n$ is process noise, which is an un-measurable input that affects the state

$\boldsymbol{F}$ is state transition matrix

$\boldsymbol{G}$ is a control matrix or input transition matrix (mapping control to state variable)

In the changing dynamic model, the $\hat{\boldsymbol{u}}$ is a variable that affecting the state matrix. For example, an aircraft that moves in x, y, z axis, and has speed in all of those axes will have an input variable in the form of acceleration. Then the F and G is a transition matrix that converts the variable of state and input matrix to compatible value so both matrices can be calculated. If the Kalman filter is used for a constant dynamic model the input variable can be considered as 0. In general, the purpose of the state extrapolation equation is to predict the state of the object after a certain time. For example, if we know the position, speed and acceleration of an aircraft, we can predict aircraft position and aircraft speed after a certain time (Δt). In the prediction state, the covariance of a system need also to be predicted by using:

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^{\boldsymbol{T}} + \boldsymbol{Q} \tag{2.6}$$

Where,

$\boldsymbol{P_{n,n}}$ is estimate uncertainty (covariance) of the current state

$\boldsymbol{P_{n+1,n}}$ is the updated estimate uncertainty

$\boldsymbol{F}$ is the state transition matrix(Same matrix that used in state extrapolation)

$\boldsymbol{Q}$ is the process noise matrix

In general, the purpose of the equation 2.6 is to readjust the parameter of the covariance matrix ($\boldsymbol{P_{n,n}}$) so it can be calculated in the equation.

2.5.3 Estimation

In the last stage of Kalman filter calculation, we get the prediction value of an object for the next measurement. For example, in moving aircraft, we can get the prediction of the aircraft position after a specific time. However, this prediction still had a noise that made the prediction value not accurate. Previously, this noise is defined as a random variable resulting in an error in the prediction; as mentioned before, Kalman filter's goal is to minimize this error, which will be done in this stage.

The first thing to be done in this stage is to know the "actual" measurement value of the object after a certain time that can be obtained by performing the second measurement. If the second measurement is already obtained, then the measurement needs to be converted with an equation that made the measurement value compatible with the Kalman filter equation. This conversion equation is known as Measurement equation:

$$\boldsymbol{z_n} = \boldsymbol{H}\boldsymbol{x_n} + \boldsymbol{v_n} \tag{2.7}$$

Where,

$\boldsymbol{z_n}$ is the measurement vector

$\boldsymbol{x_n}$ is the system state/measurement value

$\boldsymbol{v_n}$ is random noise

$\boldsymbol{H}$ is the observation matrix

The purpose of the $\boldsymbol{H}$ is to transform the state matrix or the measurement value into the desire compatible matrix that can be calculated with other parameters.

In general the next thing to be done in this phase is to comparing the result of the prediction of the system state $\boldsymbol{x_{n,n}}$ with the "actual" measurement of the system state $\boldsymbol{z_n}$. This comparison will resulting more accurate estimation of the system state. In actuality the comparison need an additional parameter that act as a correctional value that basically tell how much adjustment the system need so it can produce more accurate estimation. This adjustment parameter or commonly called kalman gain can be obtain by using:

$$\boldsymbol{K_n} = \frac{\boldsymbol{P_{n,n-1}H^T}}{\boldsymbol{HP_{n,n-1}H^T + R_n}} \tag{2.8}$$

Where,

$\boldsymbol{K_n}$ is Kalman gain

$\boldsymbol{P_{n,n-1}}$ is the previous covariance of the system

$\boldsymbol{R_n}$ is the measurement uncertainty/error

The Kalman gain is then used as a variable that controls the trade-off between the prediction of future values and the observation of current values. In other words, the Kalman gain tells how much we want to change the estimate by given measurement and prediction. The value of Kalman gain is only range between 0 and 1 or $0 \leq K_n \leq 1$. So the Kalman gain is used in the comparison equation between the prediction and "actual" measurement equation or also known as state update equation, which is:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n,n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n,n-1}} + \boldsymbol{K_n}\left(\boldsymbol{z_n} - \boldsymbol{H}\hat{\boldsymbol{x}}_{\boldsymbol{n,n-1}}\right) \tag{2.9}$$

where, $\hat{\boldsymbol{x}}_{\boldsymbol{n,n}}$ is the new estimated state vector

$\hat{\boldsymbol{x}}_{\boldsymbol{n,n-1}}$ is the current predicted system state vector

$\boldsymbol{K_n}$ is kalman gain

$\boldsymbol{H}$ is the observation matrix

$\boldsymbol{z_n}$ is the measurement vector

Intuitively if we get more accurate value of the state system, the value of the covariance should be decreasing. To obtaining the new estimation value of the covariance we can use the covariance update equation:

$$\boldsymbol{P_{n,n}} = \left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)\boldsymbol{P_{n,n-1}} \tag{2.10}$$

Where,

$\boldsymbol{P_{n,n}}$ is estimate uncertainty (covariance) matrix of the current sate

$\boldsymbol{I}$ is identity matrix

$\boldsymbol{P_{n,n-1}}$ is the previous covariance of the system

$\boldsymbol{K_n}$ is kalman gain

In general, the value of covariance should always be decreasing if the Kalman filter calculation is conducted continuously. Suppose the covariance is continuously decreasing with every calculation iteration. In that case, the value of the Kalman gain should also be decreasing because if the error/covariance is smaller, then the adjustment value(Kalman gain value) that was added in state update equation also becomes smaller.

If the new state estimation and new covariance are have been obtained, both of the values then inserted again to the prediction phase. This loop of prediction and estimation calculation is continuously done until the last measurement value is calculated. This looping calculation can be described as:

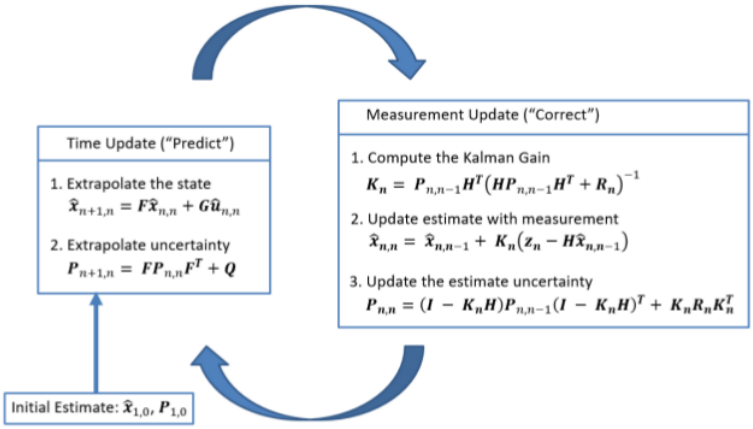

FIGURE 2.6: Kalman filter calculation loop (Becker (www.kalman-filter.net), 2018b)

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Data Acquisition

The data set used in this research was obtained from opensky-network.org. Specifically, this research used the data from March 9th 2020. Although Opensky-network offers other data set from different dates since the website adds new data once a week, this research only used the March 9th 2020 data as the case study. In the Opensky-network website, the March 9th 2020 data set is divided hourly from 00 to 24 based on GMT+0 time zone. This research only used the data from 00 to 10 because of the hardware limitation to store and read the data set. The data set is downloaded in the form of a zip file containing a CSV file of the ADS-B data. The data set from Opensky-network had several parameters, including:

- Time
- Icao24 ID
- lat (Lattitude)
- lon (Longitude)
- Velocity
- Heading
- Vertrate
- Callsign
- Onground
- Alert/SPI
- Squawk
- Baroaltitude
- Geoaltitude
- lastposupdate
- lastcontact

However, in this research, only 9 data parameters were used. Which are:

- Time: The timestamp of the data point is based on GMT+0 timezone, but the timestamp is in form of UNIX timestamp. The timestamp includes the date, month, year, hour, minute, and second description.
- Icao24 ID: Containing a unique 24-bit ICAO transponder ID that is used to differentiate aircraft/airframe.
- lat (Latitude): Geological latitude coordinate that is based on WGS84.
- lon (Longitude): Geological longitude coordinate that is based on WGS84.
- Velocity: Containing the magnitude of aircraft speed over ground in m/s(Meters per Seconds)
- Heading: Containing the direction of movement (track angle) as the clockwise angle from the geographic north.
- Vertrate: Containing the vertical speed of the aircraft in m/s
- Onground: Indicates whether the aircraft is broadcasting surface positions or airborne positions. If the aircraft is on the ground, it will show as True, and if the aircraft is airborne, it will be shown as False.
- Geoaltitude: Containing the altitude that determined using the GNSS (GPS) sensor

After extracting all the CSV files from the zip file, it needed to be merged into 1 file. The reason is to make it easier to process all the data. To merge the CSV files, this research used Python programming language. Several libraries need to be used, which are *os* and *glob* library. The code that used to merge the file are as follow:

```
#Determining the location of the file
path = 'C:\\Users\yttbk\.spyder-py3\Opensky'
extension = 'csv'
os.chdir(path)
#combining the csv file with a loop
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
```

```
combined_csv = pd.concat([pd.read_csv(f) for f in all_filenames ])
#export to csv
combined_csv.to_csv( "combined_csv.csv", index=False,
encoding='utf-8-sig')
```

After the merger, the total data point in the data set is 18855925. However, this file covering all data from all over the world, and both aircraft that still on the ground and airborne. In conducting the FPR with Kalman filter and interpolation, the calculation needs only need one aircraft that already airborne.

To filter the data set to the desire condition, first, the data point that contained on the ground aircraft needs to be removed. To accomplish this, the "onground" parameter needs to be filtered; if the data has a "True" value, the data point was removed. This can be done by using:

```
df[df['onground'].isin([false])]
```

Those codes mean that if the column "onground" did not have "False" value, it was removed. After initiating this code, the total data point decreases to 17117696.

Then to simplified the filtration, the data need to be filtered by region/country. For this research, the aircraft that fly in the US was used. The data set need to be filtered again, so aircraft that did not fly in the US were removed. To accomplish that in Python, the data set can be filtered by the range of the latitude and longitude of the United States of America or filtered by the maximum and minimum of the US's geographical coordinate. The latitude range is from 19.50139 to 64.85694, and the longitude is from -161.75583 to -68.01197(*United States latitude and longitude*, 2015). Then those coordinate range will be used as the base value of the filtration in Python. The code to filter the coordinate range is:

```
df = df[(df['lat'] >= 19.50139) & (df['lat'] <= 64.85694)]
df = df[(df['lon'] >= -161.75583) & (df['lon'] <= -68.01197)]
```

After the filtration, there only 6009740 data point left. From this 6009740, there are 9719 unique aircraft listed on the data set. So, another filtration is needed to filter one unique aircraft. To done that, another line of code is needed to filter one unique ICAO24 ID. These codes are:

3.2 Data analysis

In this research, two types of analysis methods are conducted, which are interpolation and Kalman filter. The analysis aimed to produce the estimated position of the aircraft with reduced noise/inaccuracy and the position of the aircraft between periodical updates or, in other words, interruption/intermission periods. In the interpolation method, there are three different types of interpolation that were conducted in this thesis. The first one is linear interpolation, spline interpolation, and PCHIP (Piecewise Cubic Hermite interpolating Polynomial) interpolation.

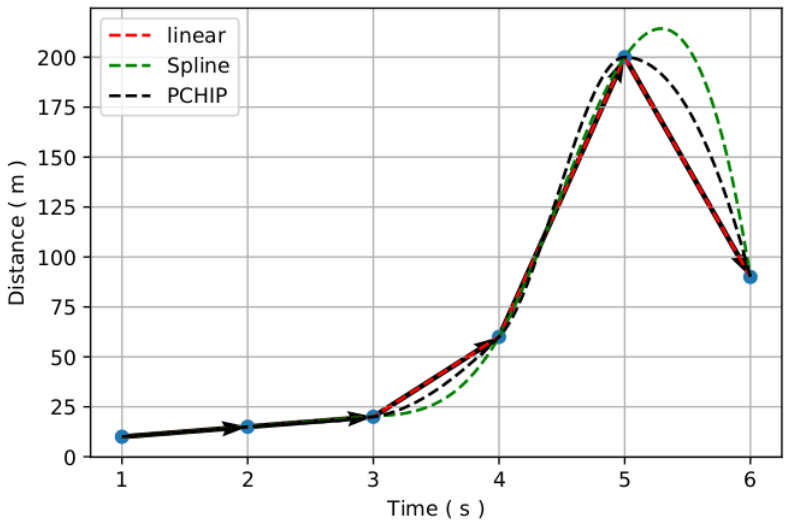

FIGURE 3.1: Example of 3 Differents Interpolation Methods(Aircraft displacement overtime)

Note: The figure 3.1 is an example of data of an aircraft displacement overtime which have been interpolate with 3 different interpolation methods.

3.2.1 Interpolation

Linear interpolation is a type of interpolation that creates a new value, which made a continuous straight line (shortest straight line) between each data point, which

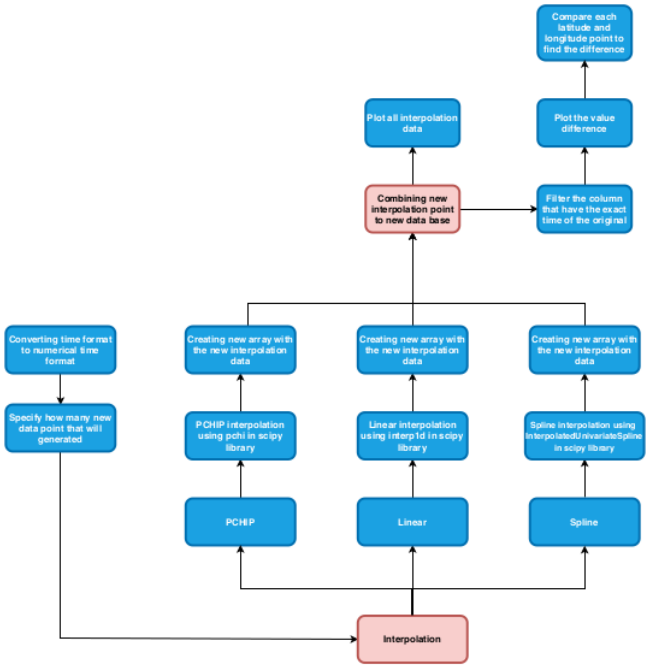

FIGURE 3.2: Algorithm of Trajectory Interpolation

means have a lesser quality of data. In this thesis, the Python programming was used the *scipy* library to produce all three different interpolation approaches. In the *scipy* library, linear interpolation can be immediately performed between the value of latitude and longitude (x and y coordinate, respectively). However, the PCHIP and spline interpolation method can not be performed directly because the *scipy* method needs one of the values of longitude or latitude that represent the x-axis to be a monotonic increasing value. This means that the x-axis value does not have any repetitive value and always increasing value along the x-axis. That is why the *scipy* library cannot directly be implemented because it is impossible to assume an aircraft always has this trajectory pattern.

To overcome this problem, there should be a new value that needs to be introduced, which had the monotonic increasing value and also represent the exact order of the coordinate movement. The value happens to be the timestamp data of the data points which is represented by time description in the data set. The

timestamp value is intended to replace the x value in the interpolation so that one of the axes still maintained a monotonic increasing value. As a result of that, the interpolation calculation was separated into two sections. The first one is for the time and longitude as x and y. The second section is for the time and latitude as x and y. Although the linear interpolation does not need this procedure, it is important to make sure the result is comparable to each other, in order to uniformize the procedure.

As mentioned before, the interpolation needs to be conducted separately, but the code/procedure is almost the same; the difference only occurs when the longitude and latitude value are called. Before using the Scipy library to commencing interpolation between the time and coordinate. The timestamp value format needs to be changed. From the original UNIX time format to conventional format. This format is consists of year, month, date, hour, minute, and second. This was done so that in the analyzation process, the result becomes more readable for human. To perform this process, this research uses the *pandas* library. The overall code to do this is:

```
time_date = pd.to_datetime[df{'time'}]
```

However, the *scipy* library did not recognize the conventional time format. So, the format needs to be converted again to numerical format. From datetime format to numerical format using matplotlib.mdate library.

```
 time_num = mdates.date2num(time_date)}
```

Then the total of new points or nodes needed to be declared, which was made in the interpolation processes. To maximize the detail in the result and the hardware limitation, the total of new points is 10 million. The new point can be made by declaring the start and final value, and then the program will made 10 million values between the start and finish. The start and final value can be determined by the minimal and maximum value of time data. The new point/node can then be made by using NumPy library.

```
timeinterpnum = np.linspace(min(timenum), max(timenum), 10000000)
```

After that, Interpolation can be done between each coordinate of latitude and longitude. All of the Interpolation can be done using Scipy.interpolate library.

For linear interpolation:

```
latiinterplin = interpolate.interp1d(timenum, lati)
latiinterplin = latiinterplin(timeinterpnum)
```

Note: For latitude

```
{longinterplin = interpolate.interp1d(timenum, long)}
{longinterplin = longinterplin(timeinterpnum)}
```

Note: For longitude

For spline interpolation, it needs to replace all the interp1d to InterpolatedUnivariateSpline. The difference between linear and Spline interpolation is that it will create an interpolation of a new point that will be made a cubic line between existing data points (the new point will have value outside the range of the existing data value).

For PCHIP interpolation replace all the inretp1d with pchip. Also, for this interpolation, it will create an interpolation of a new point that will be made into a smooth curve line between existing data points(the new point will have value outside the range of the existing data value). This interpolation approach's general idea is to mimic the real aircraft movement when turning, hence the benefit of a smooth curve interpolation generate.

After all interpolation approach has been done, the result was plotted. To plot the result, *matplotlib* library is used. The plot is made with the original longitude and latitude with the combination of the interpolated longitude and interpolated latitude. For the exact magnitude difference, this research is using MSE(Mean Square Error). The Formula of MSE can be written as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{3.1}$$

3.2.2 Kalman Filter

Before calculating using the Kalman filter. The data parameters must be adjusted. The adjustment must be made because, in Kalman filter, the calculation is

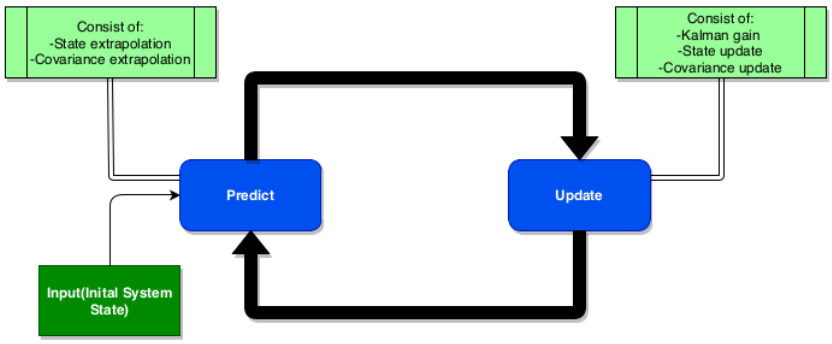

FIGURE 3.3: Algorithm of Kalman Filter Calculation

computed in vector form. In order to fulfill those criteria, the velocity parameter that still in the scalar form must be converted into a 3-dimensional vector form. The vector velocity was consisting of v_x, v_y, and v_z that correspond to velocity in latitude axis, velocity in longitude axis, and the vertical velocity. Although the vertical velocity is obtainable from the ADS-B data's vertrate parameter, the v_x and v_y must be calculated. To converting the scalar form, it will use:

$$v_x = vsin(c)$$

$$v_y = vcos(c)$$

where,

v is velocity

c is the direction of movement of the aircraft as the clockwise angle from the geographic north

In the Python programming language using *numpy* library, the code will be:

```
Vx = df['velocity'] * np.cos(df['heading']) #Speed in X axis (m/s)
Vy = df['velocity'] * np.sin(df['heading']) #Speed in Y axis (m/s)
Vz = df['vertrate'] #Speed in Z axis (m/s)
```

In this Kalman filter calculation, the input variable will be using the acceleration of the aircraft. However, in the ADS-B data, the acceleration data is unavailable. So, an additional calculation is needed. To find the acceleration of the aircraft, we

can use the velocity difference (Δv) and time difference (Δt) between data points. In the mathematical form it will be shown as:

$$a = \frac{v_2 - v_1}{t_2 - t_1} = \frac{\Delta v}{\Delta t}$$

where,

a is acceleration in m/s^2

v is scalar velocity in m/s

t is time in second s

In python programming language the time difference can be find by using:

```
time_diff = (df['time'] - df['time'].shift())
time_diff = pd.Series(pd.to_timedelta(time_diff))
time_diff = time_diff.dt.total_seconds() #time difference in second
df['time_difference'] = time_diff #declaring new data set in the data frame
```

Then after the time difference is obtained, the acceleration can be found and the vector acceleration can be found using the same method as the vector velocity. In python programming language the code will be:

```
Ac = (df['velocity'] - df['velocity'].shift()) / df['time_difference']
Ax = Ac * np.cos(df['heading'])
Ay = Ac * np.sin(df['heading'])
Az = ((df['vertrate'] - df['vertrate'].shift()) / df['time_difference'])
```

Then the coordinate position of the aircraft needs to be changed so that it was compatible with other variables. The coordinate represents the x, y, and z vector, which is latitude, longitude, and altitude were changed into the Earth-center Earth-fixed coordinate system or known as ECEF. The ECEF is a type of conventional terrestrial coordinate system that determines its position by measuring the distance between the position to the center of the earth. The ECEF will be combining the distance to the center of the earth with the degrees of latitude and longitude. Unlike the Geographic coordinate system that determines the position from the degrees displacement from the equator for determining the latitude and degrees displacement from the Greenwich point for determining the longitude.

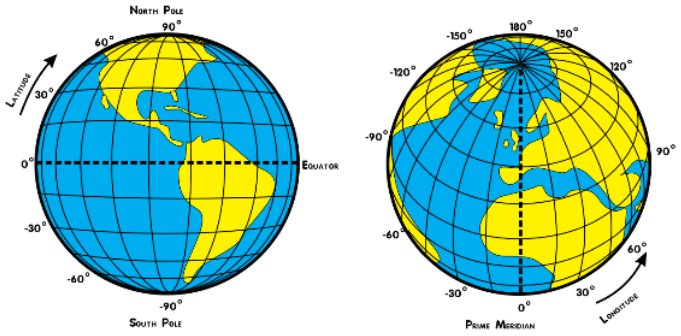

FIGURE 3.4: Geographic coordinate system (*Preprocessing: Calculate the Distance Between Longitude and Latitude Points with a Function | by The Data Detective | Towards Data Science*, 2019)

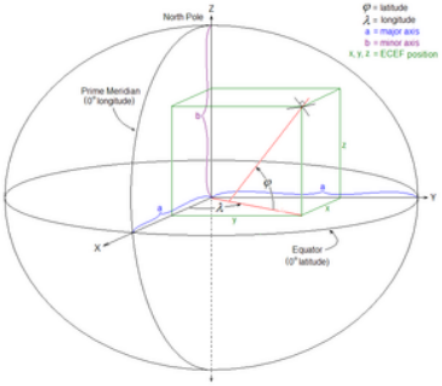

FIGURE 3.5: ECEF coordinate system (*ECEF(Earth-Centered Earth-Fixed Frame) Coordinate*, 2016)

To convert the coordinate system, a python library was used. The library is *pymap3d*. The library only needs input from the original position coordinate, consisting of latitude, longitude, and altitude. By using the geodetic2ecef command in the library, the conversion was initialized. The code is done in a loop so that all coordinate for every timestamp are converted:

```
lat = (df3['lat'])
lon = (df3['lon'])
```

```
h = df3['geoaltitude']

x_coor,y_coor,z_coor = [], [], []
x_coor,y_coor,z_coor = pm.geodetic2ecef(lat,lon,h)
```

In order to calculate the Kalman Filter with Python programming, this research will use *filterpy* library. In general, the library consisting of 2 step process just like the actual Kalman Filter calculation. The first one is the predicted step, and the second one is the updated step. In the predicted step, the calculation of the state extrapolation and covariance extrapolation was done to produce the future prediction of the aircraft state. Then in the updated stage, the Kalman gain, state update, and covariance update equation were calculated to produce more accurate information of the current aircraft state. However, before the Cartesian variable is inputted in the filterpy, the matrix form for the equation need to be defined.

The *filterpy* library already providing a command that enabling the user to input the parameter separately. However, the *filterpy* can only compute a system that has two dimensions. For example, the position of x and y axis with its speed components. So the calculation will be conducted in 2 separate forms. The first one will be calculating the x and y axis component, and the second will be calculating the z axis component. The x and y component calculation will be referred as *kf1* and the z component calculation will be referred as *kf2*.

The parameter that will be inputted in the first calculation(x and y axis) are:

State system $\boldsymbol{x_n}$ using KalmanFilter.x command:

$$kf1 = \boldsymbol{x_n} = \begin{bmatrix} x \\ y \\ V_x \\ V_y \end{bmatrix} \qquad kf2 = \boldsymbol{x_n} = \begin{bmatrix} z \\ V_z \end{bmatrix} \tag{3.2}$$

Note: x, y and z is the coordinate in ECEF form.

State transition matrix $\boldsymbol{F}$ using KalmanFilter.F command:

$$kf1 = \boldsymbol{F} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad kf2 = \boldsymbol{F} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

Note: dt is the delta time between the current timestamp and future timestamp.

Input variable $\boldsymbol{u}$ using KalmanFilter.u command:

$$kf1 = \boldsymbol{u_n} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \qquad kf2 = \boldsymbol{u_n} = \begin{bmatrix} a_z \end{bmatrix} \tag{3.4}$$

Note: a_x, a_y and a_z is the acceleration parameters.

Input transition variable $\boldsymbol{B}$ using KalmanFilter.B command:

$$kf1 = \boldsymbol{B} = \begin{bmatrix} dt^2/2 & 0 \\ 0 & dt^2/2 \\ dt & 0 \\ 0 & dt \end{bmatrix} \qquad kf2 = \boldsymbol{B} = \begin{bmatrix} dt^2/2 & 0 \\ 0 & dt \end{bmatrix} \tag{3.5}$$

Covariance matrix $\boldsymbol{P_n}$ using KalmanFilter.P command:

$$kf1 = \boldsymbol{P_n} = \begin{bmatrix} VAR(x) & 0 & 0 & 0 \\ 0 & VAR(y) & 0 & 0 \\ 0 & 0 & COV(V_x, V_x) & COV(V_x, V_y) \\ 0 & 0 & COV(V_y, V_x) & COV(V_y, V_y) \end{bmatrix} \tag{3.6}$$

$$kf2 = \boldsymbol{P_n} = \begin{bmatrix} VAR(z) & dt \\ 0 & VAR(V_z) \end{bmatrix}$$

Note: $VAR(x)$, $VAR(y)$ and $VAR(z)$ is the square of GPS accuracy, this research will use 20 meter(*Global Positioning System (GPS) in Aviation | Aircraft Systems*, 2017). The COV section means that the velocity of one vector was multiplied with other vectors. The velocity of every vector is the same, which is 4 knot or 2.05778 m/s(*Civil Aviation Order 108.56 Instrument 2007*, 2007). The VAR means that the noise of a vector does not affect one another, which the opposite of the COV the noise in every vector is effecting others vector. Hence why the position is using COV or, in other words, is covariance except the V_z because it does not calculate with others vector velocity and the position using VAR or variance.

measurement matrix $\boldsymbol{H}$ using KalmanFilter.H command:

$$kf1 = \boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad kf2 = \boldsymbol{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

Measurement covariance matrix $\boldsymbol{R}$ using KalmanFilter.R command:

$$kf1 = \boldsymbol{R_n} = \begin{bmatrix} VAR(x_m) & 0 & 0 & 0 \\ 0 & VAR(y_m) & 0 & 0 \\ 0 & 0 & COV(V_x, V_x) & COV(V_x, V_y) \\ 0 & 0 & COV(V_y, V_x) & COV(V_y, V_y) \end{bmatrix} \tag{3.8}$$

$$kf2 = \boldsymbol{R_n} = \begin{bmatrix} VAR(z_m) & dt \\ 0 & VAR(V_z) \end{bmatrix}$$

Note: x_m, y_m and z_m is based on the accuracy indicator or NIC value. It needs to be noted that these values can fluctuate between each iteration.

For the process noise matrix, $\boldsymbol{Q}$ is not used in the calculation because of the unavailability of the data parameter.

Usually, the ADS-B data have the quality indicator value that consists of NIC,

NACp, or SIL, but the data set provided by OpenSky-network did not have it. So, in this research, the quality indicator is generated by using python code. Considering this research use the NIC value as the bases for the quality indicator, the magnitude of the inaccuracy or noise was based on NIC value.

The NIC value is generated randomly with predetermine probability. Only the maximum value of each NIC value will be used in value generation. So, the probability of each NIC value is:

NIC Value	Probability
0	2.39%
1	0.01%
2	0.01%
3	0.07%
4	0.01%
5	0.01%
6	0.01%
7	0.01%
8	0.08%
9	42.92%
10	50.59%
11	3.93%

TABLE 3.1: NIC Value Probability (Tesi & Pleninger, 2017)

Note: For the NIC 0 the horizontal accuracy is supposed to be unknown, but in this research, the horizontal accuracy of the NIC 0 will be 20 meters. For the vertical accuracy of the NIC value, less than 8 the 20-meter magnitude also will be used.

Then the equation process can be start by initiating an predict and update command in looping sequence for every timestamp:

```
#for kf1
for i in range(count):
pos = Measurement[i]
z = pos
kf1.predict ()
```

```
kf1.update (z)
xs.append (kf1.x[0,0])
ys.append (kf1.x[1,0])
vxs.append (kf1.x[2,0])
vys.append (kf1.x[3,0])
pxs.append (pos[0])
pys.append(pos[1])

#for kf2
for i in range(count):
pos2 = Measurement2[i]
z = pos2
kf2.predict ()
kf2.update (z)
kzs.append (kf2.x[0,0])
vzs.append (kf2.x[1,0])
pzs.append (pos2[0])
pvz.append(pos2[1])
```

The resulting new estimate coordinate is still in the ECEF form, so it needed to be reconverted to geodetic form. The reason for this change is to compare the original data and new estimated data. To accomplish that task, a method of determining the data difference is needed. This research will be using the MSE (Mean Squared Error) method.

The reconverted coordinate also plotted using *basemap* library. So that the actual aircraft trajectory of estimated data from Kalman filter and original ADS-B data can be visualized in the geodetic format. The original result of the ECEF coordinate also will be plotted with *matplotlib* library, so that the difference between original ECEF ADS-B coordinate and Kalman filter data can be analyzed.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Interpolation Result

As mentioned before, in the previous chapter, the interpolation result was plotted with the original ADS-B data. This research was intended to create four different plots. Three of them contain a single interpolation and original data, and one of them contains all available data (all interpolation results and original ADS-B data). This was intended to visualize the result and visualize the comparison between the interpolation data and original ADS-B data.

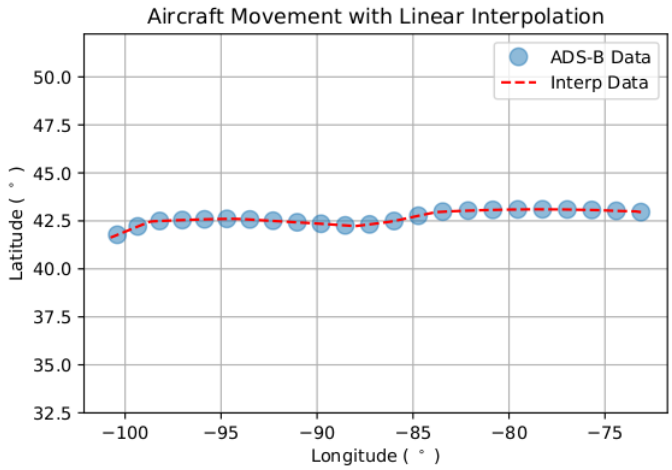

FIGURE 4.1: Linear Interpolation

As mention before the figure 4.1, 4.2, 4.3, and 4.4 is the result of comparison between interpolation data and ADS-B data. Originally the ADS-B data has 1119 timestamp or, in other words, 1119 data points. If all the original ADS-B data

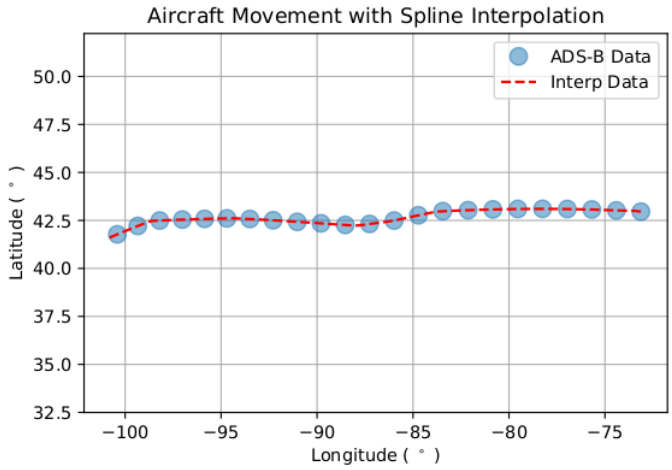

FIGURE 4.2: Spline Interpolation

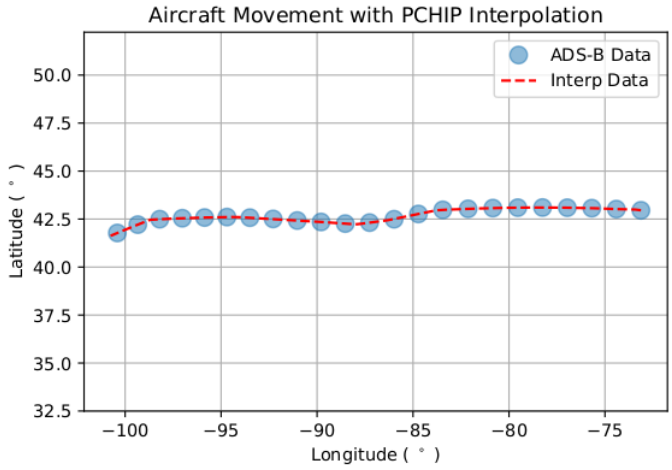

FIGURE 4.3: PCHIP Interpolation

points are used, each data point's separation is unseen able. To overcome this, the plot only shows a data point in 50 data point intervals, or only the 50^{th} data point is shown. To make each data point more visible in the plot, this research uses the scatter plot mode in the *matplotlib* library.

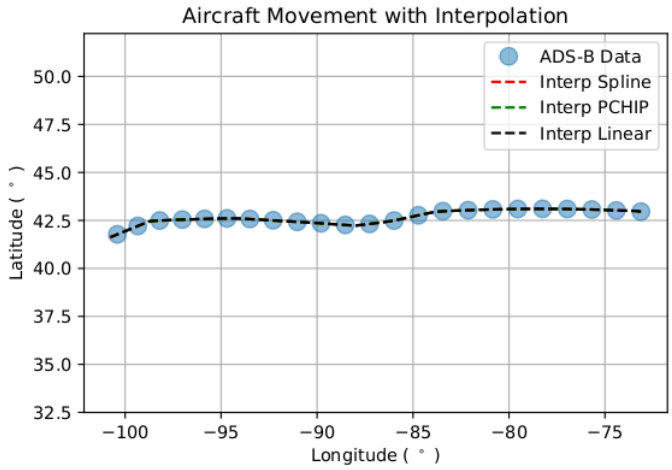

FIGURE 4.4: Combination of All Interpolation

As mentioned in the previous chapter, originally, there is the sum of the total original data point plus 10 million new data points created in interpolation calculation. So, in general, there are 10 million and 1119 data points that were created in the interpolation calculation. This resulted in 10 million new data points are filling in the timestamp that was not covered in the original ADS-B data. As a result of that, the delta time between each data point becomes smaller. In order to visualize the smaller time interval between each data point as seen in figure 4.1, 4.2, 4.3, and 4.4, this research used the line plot when using *matplotlib* library. This was aimed to resemble the continuous aircraft trajectory during flight, or in other words, it will resemble the interpolation data that does not have data point intervals.

As seen in figure 4.4 that the interpolation result between 3 interpolation methods did not have a visual difference as expected to see in the example in the figure 3.1. The reason is that the small periodical separation in the original data is not big enough that eventually made the graph indifference. The average delta time in this data example is 10 seconds.

Then, an analysis of the magnitude difference between interpolation data and ADS-B data is needed. For this analysis, this research will use the MSE method. To

be able to conduct this analysis first, both data need to have matching timestamp value. To do this, the interpolation data need to filter. This aims to remove data points that did not have a matching timestamp with the original ADS-B data. After that is done, then the analysis can begin. The MSE calculation was done in 2 separate forms. The first one is for the MSE between latitude and longitude coordinate of interpolation data and original ADS-B data. The second one is for distance difference in meter unit between each coordinate.

MSE of Coordinate Difference	MSE (degree)
Latitude of Linear Interpolation and ADS-B	0
Latitude of Spline Interpolation and ADS-B	5.46×10^{-28}
Latitude of PCHIP Interpolation and ADS-B	0
Longitude of Linear Interpolation and ADS-B	0
Longitude of Spline Interpolation and ADS-B	5.46×10^{-28}
Longitude of PCHIP Interpolation and ADS-B	0

TABLE 4.1: MSE of Latitude and Longitude

MSE of Distance	MSE (m^2)
ADS-B Data and Linear Interpolation	0
ADS-B Data and Spline Interpolation	2.74×10^{-18}
ADS-B Data and PCHIP Interpolation	0

TABLE 4.2: MSE of Distance

As seen in table 4.1 and 4.2 the value MSE is small. As a result of the MSE is above 10×10^{-14} the error can be ignored. So, in other words, the resulting interpolation coordinate has no difference with the original ADS-B data. However, it needed to remember that in reality, both values still had a noise or inaccuracy because, in the interpolation calculation, the noise and inaccuracy are not taken into consideration during the calculation.

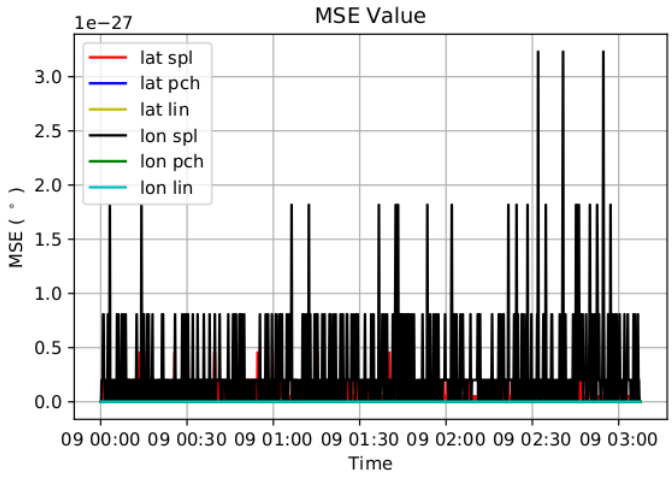

FIGURE 4.5: MSE of Latitude and Longitude Over Time

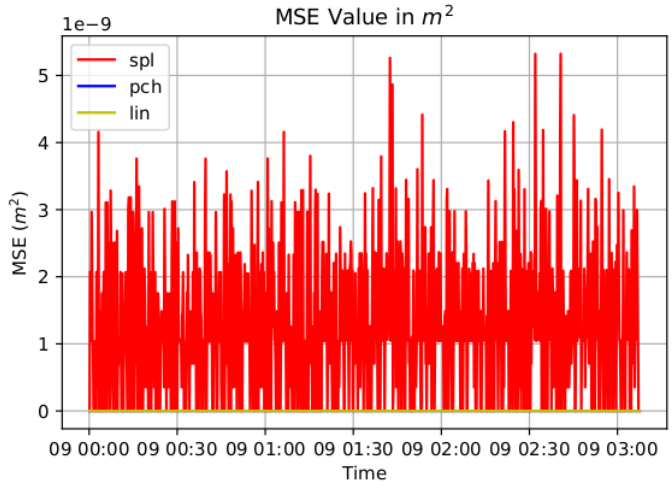

FIGURE 4.6: MSE of Distance Over Time

4.2 Kalman Filter Result

The original result of the Kalman filter data is still in the ECEF format. The Kalman filter calculation result is then plotted into two different graphs: latitude

and longitude comparison between ADS-B data and estimated data(Kalman filter result) and the graph of the comparison altitude over time between ADS-B data and estimated data. To make the data point interval more visible, the graph only displays 1 data point every 50 available data points for both estimated data and the original ADS-B data. Both data was plotted using scatter mode. To make it easier to visualize the difference between ADS-B data and estimated data, the scatter plot's size is increased. As the plot results still use the ECEF form, the latitude, longitude, and altitude are still in vector form using the meter unit.

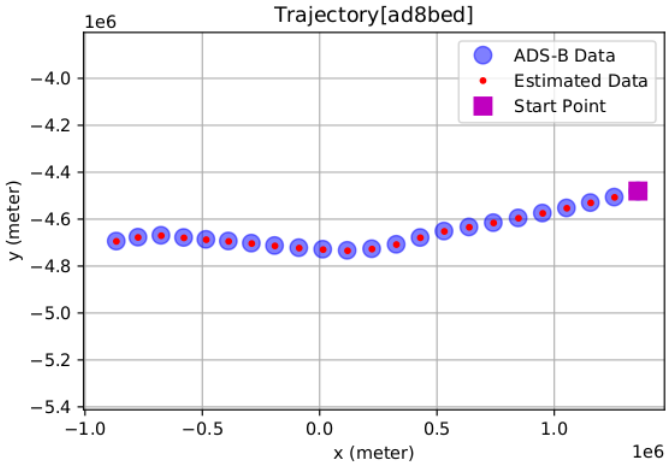

FIGURE 4.7: Latitude and Longitude Trajectory

As seen in figure 4.8 and 4.8, there is a small deviation between the estimated data and ADS-B data. To measure those position difference, MSE calculation is conducted. The result of the MSE calculation are displayed on table 4.3.

The result of MSE in the original coordinate of the latitude and longitude form also needs to be calculated. This was aimed to calculating actual position difference, which was not influenced by other factors. To accomplish that, this research reconverting the coordinate using the *pymap3d* library. After the reconversion, both coordinates can be plotted using *basemap* library so that the visualization of the aircraft trajectory is straightforward. The result of the *basemap* plot and MSE result can be seen in the figure 4.9 and table 4.4.

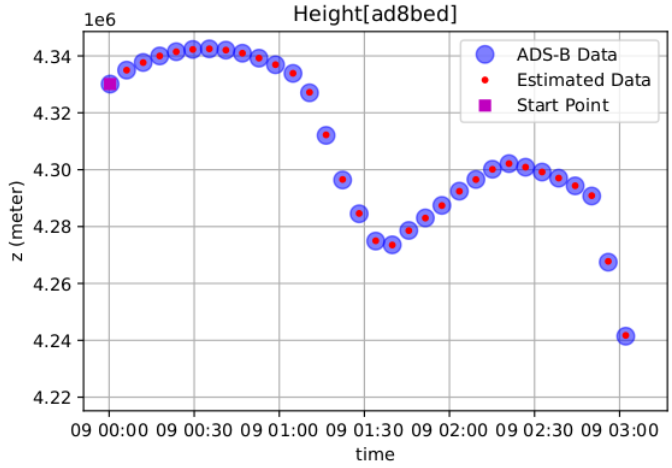

FIGURE 4.8: Altitude Trajectory

MSE of Distance	MSE (m^2)
Latitude of Estimated ECEF and ADS-B ECEF	2.55×10^6
Longitude of Estimated ECEF and ADS-B ECEF	2.35×10^6
Altitude of Estimated ECEF and ADS-B ECEF	2.80×10^3

TABLE 4.3: MSE of ECEF Coordinate

MSE of Distance	MSE (degree)
Latitude of Estimated and ADS-B	9.24×10^{-5}
Longitude of Estimated and ADS-B	3.42×10^{-4}

TABLE 4.4: MSE of Earth Coordinate

To make it easier to assess each data's actual difference, this research implemented another MSE calculation for using the distance using the meter unit. The

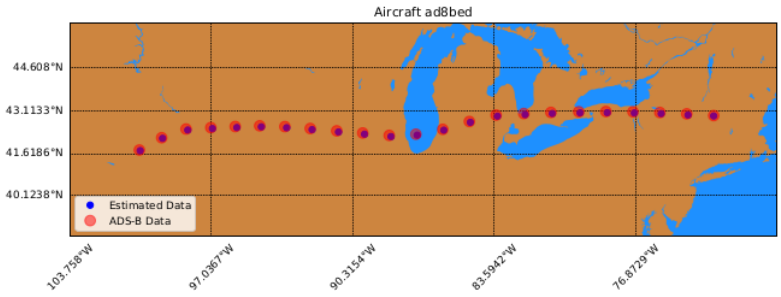

FIGURE 4.9: Aircraft Trajectory

distance between coordinates in each data point can be calculated using *geopy* library and using the *vincenty* command. The result of MSE distance using meter unit can be seen in the tabel 4.5.

MSE of Distance	MSE (m^2)
Difference Between Estimated and ADS-B Data	1.10×10^5

TABLE 4.5: MSE of Earth Coordinate in m^2

It needs to remember that, theoretically, Kalman filter is always assumed to be more accurate. This is because, in the Kalman filter calculation, the noise or inaccuracy indicated by the quality indicator of NIC is minimized. In this case study, most of the generated NIC values were in NIC 8 and NIC 9. So, if most of the NIC value is 8 and 9, the resulting MSE is 1.10×10^5. This research used other levels of NIC to simulate the aircraft's trajectory if another quality indicator value is applied. There are three different values that were used, which are 10 m, 45 m, and 30000 m. The result of the comparison between normal fluctuated noise and the 3 other constant noise can be seen in the figure 4.10, 4.11, and 4.12, and

then table 4.6 is displaying the result as normal case, case 1, case 2, and case 3 for the normal MSE, 10 m MSE, 45 m MSE and 30000 m MSE respectively.

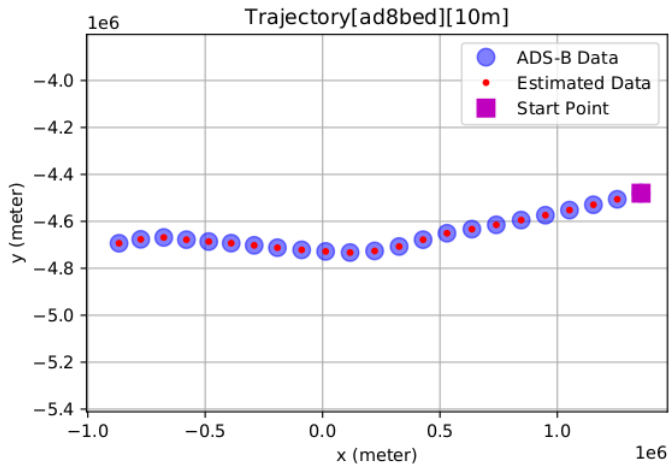

FIGURE 4.10: 10 meter Inaccuracy

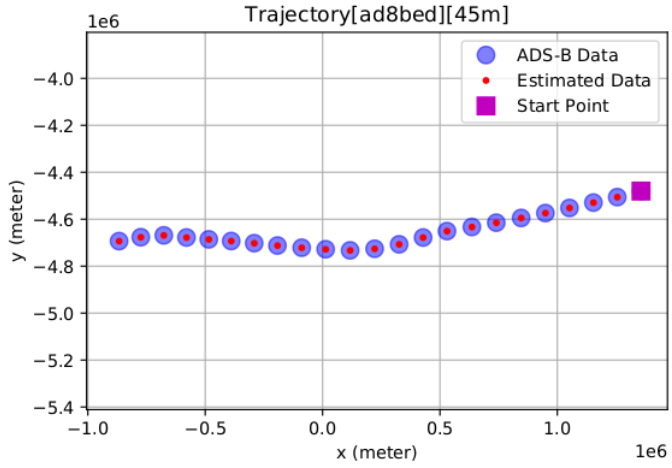

FIGURE 4.11: 45 meter Inaccuracy

In the Kalman filter calculation, determining the aircraft's position between the interruption period is not straight forward as the Interpolation method. To obtain

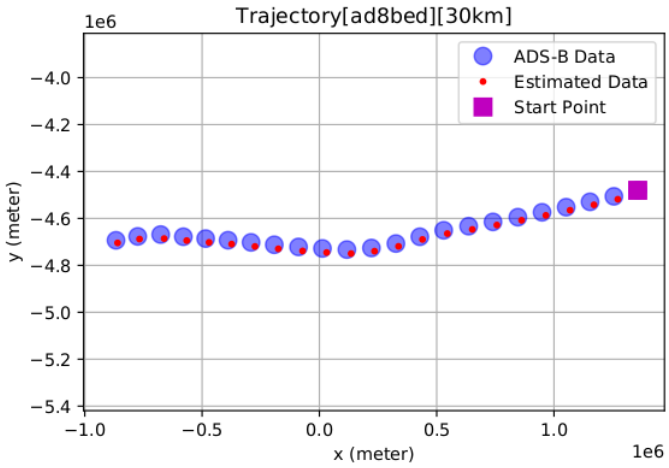

FIGURE 4.12: 30000 meter Inaccuracy

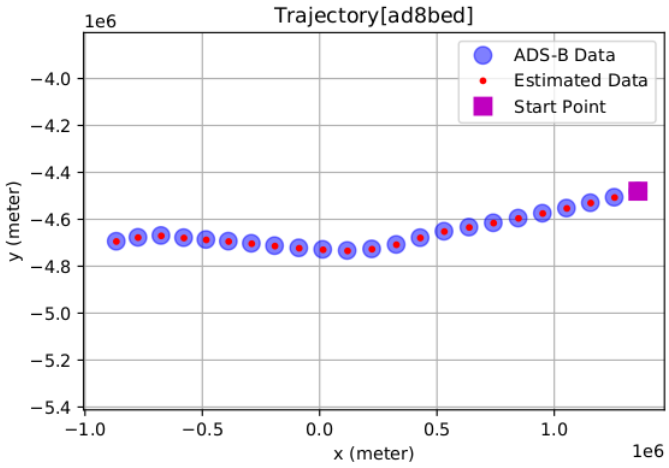

FIGURE 4.13: Normal Inaccuracy

this information, the delta time in calculation needs to be changed manually. For example, in the previous case study, the average interruption period is 10 seconds, and now the FPR needs aircraft position data if the interruption time is 5 seconds. To accomplish that, the dt in Kalman filter calculation needs to change manually

Based on ECEF Axis	Normal case MSE(m^2)	Case 1 MSE(m^2)	Case 2 MSE(m^2)	Case 3 MSE(m^2)
X	2.55×10^6	5.42×10^3	6.18×10^4	2.29×10^8
Y	2.35×10^6	4.52×10^3	5.49×10^4	2.12×10^8
Z	8.11×10^4	4.14×10^2	2.80×10^3	4.79×10^6

TABLE 4.6: MSE Comparison Between Difference Noise

to 5. The result of this method can be seen in figure 4.14.

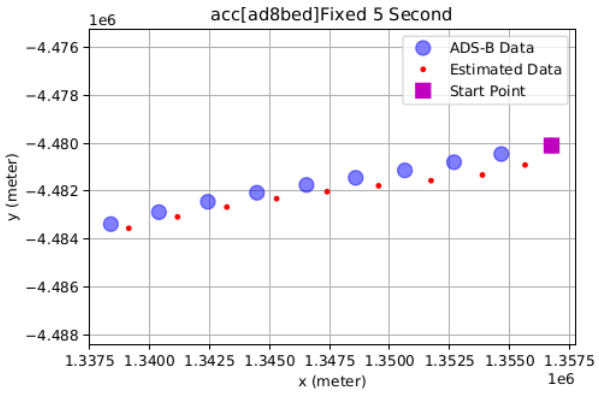

FIGURE 4.14: Fixed 5 Second Delta Time

This research also exploring ten differents aircraft trajectory so that the average of MSE can be obtained. Each MSE of ECEF component was calculated. Then the distance between each coordinate was also calculated the result of the calculation can be seen in table 4.7. However, it is important to mention that the aircraft a4e108, a009a4, ac9ff2, a732d9, a05a01, and a65c29 have a high intermission period, which ranges from 410 to 1040 second. This large intermission period may occur because the aircraft fly in the area that did not have any ADS-B receiver coverage.

4.3 Result Comparison

In this section, the MSE between each method and original ADS-B data will be compared. The comparison is aim to showing the magnitude of the MSE for each method. The comparison can be seen from table 4.8 to table 4.10

A/C Code	x (m^2)	y (m^2)	z (m^2)	Distance (m^2)
ad8bed	2.55×10^6	2.35×10^6	8.11×10^4	1.10×10^5
a4e108	4.14×10^6	2.64×10^6	6.71×10^5	4.26×10^6
a009a4	9.63×10^6	2.51×10^6	3.89×10^4	1.08×10^7
a01c37	1.66×10^6	2.42×10^6	1.06×10^4	1.04×10^5
ac9ff2	8.09×10^7	8.86×10^6	2.34×10^4	3.81×10^8
a732d9	1.29×10^7	1.62×10^6	6.32×10^5	1.50×10^7
a05a01	1.34×10^7	2.64×10^6	1.90×10^5	1.46×10^7
a65c29	1.12×10^7	3.26×10^6	1.08×10^4	1.30×10^7
a149f1	6.53×10^6	5.49×10^6	4.63×10^3	8.51×10^6
a3a8ef	3.41×10^6	3.69×10^6	1.45×10^4	2.75×10^5
Average	1.46×10^7	3.54×10^6	1.67×10^5	4.47×10^7

TABLE 4.7: Other Aircrafts MSE

MSE of Latitude	Kalman (degree)	Linear (degree)	Spline (degree)	PCHIP (degree)
ADS-B	9.24×10^{-5}	0	6.12×10^{-26}	0

TABLE 4.8: MSE of Latitude Result Comparison

MSE of Longitude	Kalman (degree)	Linear (degree)	Spline (degree)	PCHIP (degree)
ADS-B	3.42×10^{-4}	0	2.73×10^{-25}	0

TABLE 4.9: MSE of Longitude Result Comparison

MSE of Distance	Kalman (m^2)	Linear (m^2)	Spline (m^2)	PCHIP (m^2)
ADS-B	1.10×10^5	0	2.74×10^{-18}	0

TABLE 4.10: Distance Difference Comparison in m^2

CHAPTER 5
SUMMARY, CONCLUSION, AND RECOMMENDATION

5.1 Summary

In summary, FPR or flight path reconstruction by using the interpolation method and Kalman filter method have been done. In this research, both approaches were made using Python programming language. It is mainly using the *scipy* library for the interpolation and *filterpy* library for the Kalman filter calculation. The analysis that was conducted in this research is primarily based on the MSE comparison between original ADS-B data and result from both calculation methods. To visualize the result of interpolation and Kalman filter trajectory, this research mainly use the *matplotlib* library's in Python programming language.

5.2 Conclusion

In conclusion, the result of interpolation result virtually did not have any error value, or in other words, the original ADS-B data is similar to the interpolation result. This occurred due to the small delta time between timestamp or small interruption period. As a result of that, the interpolation MSE of distance between coordinates were very small if compared with the original ADS-B data. The exact value of the distance MSE between aircraft trajectory is 0 m^2 for linear interpolation, 2.74×10^{-18} m^2 for spline interpolation, and 0 m^2 for PCHIP interpolation which means there is virtually no MSE because of the small maximum MSE value. However, those calculation results neglected the inaccuracy/noise of the ADS-B system. As a result, the interpolation method became inaccurate.

For the result of Kalman filter, theoretically this method result is the most accurate trajectory. With the assumption of most of the NIC Value is 8 and 9, the MSE between the original data and Kalman filter data is $1.10 \times 10^5\ m^2$. For various inaccuracy values of 10, 45 ,and 30000 meters the MSE became $5.42 \times 10^3\ m^2$, $4.52 \times 10^3\ m^2$, and $4.14 \times 10^2\ m^2$ for x, y, and z of 10 meters inaccuracy, $6.18 \times 10^4\ m^2$, $5.49 \times 10^4\ m^2$, and 2.80e+03 m^2 for x, y, and z of 45 meters inaccuracy and $2.29 \times 10^8\ m^2$, $2.12 \times 10^8\ m^2$, and $4.79 \times 10^6\ m^2$ for x, y, and z of 30000 meters inaccuracy. Then for producing aircraft trajectory between the interruption period, interpolation method was show to be much easier because when using Kalman filter the tools user needs to input the desired delta time manually.

However, it is important to remember that the result of the interpolation calculation methods has 0 or very small MSE value, it does not mean that it was accurate because, in this calculation, the inaccuracy/noise of the ADS-B system did not taken into consideration. While the MSE in Kalman filter did not consider as an error but as a trajectory deviation after noise/inaccuracy of ADS-B data is minimized. In other words, the noise/inaccuracy of the ADS-B data resulting an MSE of $1.10 \times 10^5\ m^2$ (for aircraft ad8bed).

5.3 Recommendation

As for the recommendation for future research in the same field, this research recommends using an ADS-B data set that has the original quality indicator value and not using a randomly generated version. This is aimed for producing a result that is not influenced by human-generated data. Then, the next recommendation is to develop a new python library for Kalman filter calculation. So, it is not required to perform a data adjustment before calculation the Kalman filter.

Aircraft Flight Path Reconstruction Based on ADS-B Data Using Kalman Filter

ORIGINALITY REPORT

8%	%	%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	A.M.P. de Leege, M. M. Van Paassen, M. Mulder. "Using Automatic Dependent Surveillance-Broadcast for Meteorological Monitoring", AIAA Guidance, Navigation, and Control Conference, 2012 Publication	2%
2	82.94.179.196 Internet Source	<1%
3	Submitted to AUT University Student Paper	<1%
4	www.kalmanfilter.net Internet Source	<1%
5	Submitted to Lake Michigan College Student Paper	<1%
6	docplayer.net Internet Source	<1%
7	Submitted to University of Melbourne Student Paper	<1%

8	Submitted to Ramapo Indian Hills Regional High School District Student Paper	<1%
9	garuda.ristekbrin.go.id Internet Source	<1%
10	Submitted to Embry Riddle Aeronautical University Student Paper	<1%
11	Submitted to University of Salford Student Paper	<1%
12	opensky-network.org Internet Source	<1%
13	Ali, Busyairah Syd, Washington Yotto Ochieng, Wolfgang Schuster, Arnab Majumdar, and Thiam Kian Chiew. "A safety assessment framework for the Automatic Dependent Surveillance Broadcast (ADS-B) system", Safety Science, 2015. Publication	<1%
14	Submitted to Cranfield University Student Paper	<1%
15	archive.org Internet Source	<1%
16	www.avidyne.com Internet Source	<1%

17	Submitted to Bogazici University Student Paper	<1%
18	Zhijun Wu, Tong Shang, Anxin Guo. "Security Issues in Automatic Dependent Surveillance - Broadcast (ADS-B): A Survey", IEEE Access, 2020 Publication	<1%
19	en.wikipedia.org Internet Source	<1%
20	Tengyao Li, Buhong Wang, Fute Shang, Jiwei Tian, Kunrui Cao. "Online sequential attack detection for ADS-B data based on hierarchical temporal memory", Computers & Security, 2019 Publication	<1%
21	Submitted to Ayrshire Regional College Student Paper	<1%
22	www.lexology.com Internet Source	<1%
23	Submitted to bne-catholic Student Paper	<1%
24	Submitted to University of New South Wales Student Paper	<1%
25	Reza Noval Pahlevy, Agus Dwi Prasetyo, Edwar. "Nanosatellite ADS-B Receiver Prototype for Commercial Aircraft Detection",	<1%

	2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2018 Publication	
26	Ravindra V. Jategaonkar. "Flight Vehicle System Identification: A Time-Domain Methodology, Second Edition", American Institute of Aeronautics and Astronautics (AIAA), 2015 Publication	<1%
27	www.ncbi.nlm.nih.gov Internet Source	<1%
28	Tengyao Li, buhong wang, Fute Shang, Jiwei Tian, Kunrui Cao. "Threat Model and Construction Strategy on ADS-B Attack Data", IET Information Security, 2020 Publication	<1%
29	Zhao, Y.. "Analysis on the diffusion hazards of dynamic leakage of gas pipeline", Reliability Engineering and System Safety, 200701 Publication	<1%
30	prr.hec.gov.pk Internet Source	<1%
31	Submitted to Indian Institute of Technology, Kanpur Student Paper	<1%

32	www.ronnieduisters.com Internet Source	<1%
33	halshs.archives-ouvertes.fr Internet Source	<1%
34	Ming-Shih Huang, Ram M. Narayanan, Yan Zhang, Arthur Feinberg. "Tracking of Noncooperative Airborne Targets Using ADS-B Signal and Radar Sensing", International Journal of Aerospace Engineering, 2013 Publication	<1%
35	aeroelectric.com Internet Source	<1%
36	freeproject.com.ng Internet Source	<1%
37	feast.ucsd.edu Internet Source	<1%
38	Shuqing Zhang, Junyan Zhang. "Theoretical analytics of stereographic projection on 3D objects' intersection predicate", International Journal of Geographical Information Science, 2010 Publication	<1%
39	Sanat Kaul. "Chapter 96 Smallsats, Hosted Payload, Aircraft Safety, and ADS-B Navigation Services", Springer Science and Business	<1%

Media LLC, 2020
Publication

40	Tengyao Li, Buhong Wang, Fute Shang, Jiwei Tian, Kunrui Cao. "Dynamic temporal ADS-B data attack detection based on sHDP-HMM", Computers & Security, 2020 Publication	<1%
41	Yue Zhang, Hairong Chu. "The angular velocity estimation principal of α-β-γ filter based on high sampling rate", 2011 Second International Conference on Mechanic Automation and Control Engineering, 2011 Publication	<1%
42	Eman Hableel, Joonsang Baek, Young-Ji Byon, Duncan S. Wong. "How to protect ADS-B: Confidentiality framework for future air traffic communication", 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2015 Publication	<1%
43	atmsymposium.eurocontrol.fr Internet Source	<1%
44	Jing Wang, Yunkai Zou, Jianli Ding. "ADS-B spoofing attack detection method based on LSTM", EURASIP Journal on Wireless Communications and Networking, 2020 Publication	<1%

45	www.elprocus.com Internet Source	<1%
46	J.A. Mulder, Q.P. Chu, J.K. Sridhar, J.H. Breeman, M. Laban. "Non-linear aircraft flight path reconstruction review and new advances", Progress in Aerospace Sciences, 1999 Publication	<1%
47	mafiadoc.com Internet Source	<1%
48	Taehwan Cho. "Automatic Dependent Surveillance - Broadcast for surveillance of Unmanned Aircraft System", 2017 Integrated Communications, Navigation and Surveillance Conference (ICNS), 2017 Publication	<1%
49	Zhang Tao, , and Tang Xiaoming. "High-accuracy radar calibration based on ADS-B", IET International Radar Conference 2015, 2015. Publication	<1%
50	Sandra Sendra Compte. "Deployment of Efficient Wireless Sensor Nodes for Monitoring in Rural, Indoor and Underwater Environments", Universitat Politecnica de Valencia, 2013 Publication	<1%
51	Vijay Ragothaman, Fariha Baloch, Ravi Pendse. "Unassisted Aircraft Landing Via Co-Operative	<1%

Data Exchange", 2006 ieee/aiaa 25TH Digital Avionics Systems Conference, 2006
Publication

---

52 Caterina Grillo, Francesco Vitrano. "State Estimation of a Nonlinear Unmanned Aerial Vehicle Model using an Extended Kalman Filter", 15th AIAA International Space Planes and Hypersonic Systems and Technologies Conference, 2008 <1%
Publication

---

53 Christine Falk, Juan Gonzalez, Jose Perez. "Using Automatic Dependent Surveillance-Broadcast Data for Monitoring Aircraft Altimetry System Error", AIAA Guidance, Navigation, and Control Conference, 2010 <1%
Publication

---

54 Akshaya U Kulkarni, Amit M Potdar, Suresh Hegde, Vishwanath P Baligar. "RADAR based Object Detector using Ultrasonic Sensor", 2019 1st International Conference on Advances in Information Technology (ICAIT), 2019 <1%
Publication

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Curriculum Vitae

	Basic Information
Name	Yazfan Tabah Tahta Bagaskara
Place of Birth	Jakarta
Date of Birth	13, 01, 1998
Address	Malidar, Jatihasih, Bekasi, Jawa Barat
Year	**Education**
2015 - present	Internation University Liaison Indonesia (IULI)
2012 - 2015	PB. Soedirman Islamic High School
2009 - 2012	Al-Fajar Islamic Junior High School
2003 - 2009	Al-Fajar Islamic Elementary School
Year	**Courses**
2016	Festo: Pneumatic and Hydraulic Training
Year	**Seminars & Workshops**
2015	6th GAST Cnference
2017	MRO(maintenance, repair, and operations) is Promising Career
2017	AOA(Angle of Attack) Seminar
Year	**Work Experiences**
2017	Citilink
2018	GMF(Garuda maintenance facility)