

A Novel Relative Positioning Estimation System (RPES) Using MEMS-Based Inertial
Sensors

by

Hani Balkhair
B.Eng., Umm Al-Qura University, 2002

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Mechanical Engineering

© Hani Balkhair, 2011
University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy
or other means, without the permission of the author.

Supervisory Committee

A Novel Relative Positioning Estimation System (RPES) Using MEMS-Based Inertial
Sensors

by

Hani Balkhair
B.Eng., Umm Al-Qura University, 2002

Supervisory Committee

Dr. Nikolai Dechev, (Department of Mechanical Engineering)
Supervisor

Dr. Yang Shi, (Department of Mechanical Engineering)
Departmental Member

Abstract

Supervisory Committee

Dr. Nikolai Dechev, (Department of Mechanical Engineering)
Supervisor

Dr. Yang Shi, (Department of Mechanical Engineering)
Departmental Member

The use of MEMS-based inertial sensors for a relative positioning estimation system (RPES) was investigated. A number of data acquisition and processing techniques are developed and tested, to determine which one would provide the best performance of the proposed method. Because inertial-based sensors don't rely on other references to calibrate their position and orientation, there is a steady accumulation of errors over time. The errors are caused by various sources of noise such as temperature and vibration, and the errors are significant. This work investigates various methods to increase the signal-to-noise ratio, in order to develop the best possible RPES method. The main areas of this work are as follows: (i) The proposed RPES application imposes specific boundary conditions to the signal processing, to increase the accuracy. (ii) We propose that using redundant inertial rate sensors would give a better performance over a single rate sensor. (iii) We investigate three Kalman filter algorithms to accommodate different combinations of sensors: Parallel sensors arrangement, Series sensors arrangement, and compression arrangement. In implementing these three areas, the results show that there is much better improvement in the data in comparison to using regular averaging techniques.

Table of Contents

Supervisory Committee	ii
Abstract.....	iii
Table of Contents	iv
List of Tables	viii
List of Figures.....	xi
Acknowledgments	xvi
Dedication.....	xvii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Literature Review	4
1.2.1 Global Positioning Systems (GPS).....	4
1.2.2 Indoor Positioning System (IPS)	5
1.2.3 Inertial Navigation System (INS).....	5
1.2.4 Existing Methods for Position Estimation.....	6
1.2.5 Error Estimation with Inertial Navigation Sensors (INS)	7
1.3 Patents Review	7
1.3.1 High Accuracy Inertial Sensors from Inexpensive Components	7
1.3.2 Hand-Held Surveying Device and Surveying Method for Such A Surveying Device.....	9
1.4 Thesis Overview	10
2 Positioning Systems.....	11
2.1 Measurements Tools Overview	11

2.2 Global Positioning System (GPS)	12
2.2.1 GPS Overview	12
2.2.2 GPS Working Principles.....	13
2.2.3 Trilateration	14
2.2.4 GPS Distance Measurements	16
2.2.5 Sources of Error in GPS	17
2.3 MEMS and Microsystems	17
2.3.2 Sensors.....	19
2.3.2.2 Sensor Characteristics Definitions	19
2.3.3 Inertial Sensors	21
2.3.3.1 Accelerometers	21
2.3.3.1.1 Capacitive Accelerometers	23
2.3.3.1.2 Piezoresistive Accelerometers.....	24
2.3.3.2 Gyroscopes	24
3 Overview of Signal Processing	26
3.1 Inertial Navigation System (INS).....	26
3.1.1 Angular Accelerometers and Gyroscopes	27
3.2 Positioning Measurements in Hand-Held Devices.....	28
3.3 Errors	29
3.4 Noise.....	31
3.4.1 Types of Noise.....	32
3.4.2 Sources of Noise.....	33
3.4.3 Noise Affects on Digital Signals	34

3.5 Filters	36
3.5.1 Types of Filters	36
3.5.2 Kalman Filter	39
4 Relative Positioning Estimation System (RPES) Device	44
4.1 Device Overview	44
4.1.1 Prototype Device Component Specifications	46
4.1.2 Prototype Device Layout with Single Accelerometer	46
4.1.3 Prototype Device Layout with Four Redundant Accelerometers	48
4.2 Experiment Set-Up	49
4.2.1 Kalman Filter implementation in MATLAB	52
5 Experimental Set-Up	54
5.1 Single Accelerometer Experiment	54
5.1.1 Applying The Kalman Filter	57
5.1.2 Different Techniques to Reduce Offset and Drift Error	58
5.1.3 Applying the Moving Average Filter	70
5.1.4 Kalman Filter and Standard Averaging Techniques	73
5.2 Multiple Accelerometers Experiment	76
5.2.1 Proposed Methods	83
5.2.1.1 The Parallel Design [38]	83
5.2.1.2 The Series Design [38]	88
5.2.1.3 The Compression Design [38]	91
6 Results and Comparisons	95
6.1 Single Sensor Approach	95

6.2 Multi-Sensor Approach	99
7 Discussion.....	102
7.1 Single Sensor Approach	102
7.1.1 Use of Boundary Conditions	102
7.1.2 Use of Moving Averages & Boundary Conditions	104
7.1.3 Use of Kalman Filter & Boundary Conditions.....	107
7.1.4 Comparison of All Single Sensor Processing Methods.....	108
7.2 Single Sensor Short-Duration Analysis.....	110
7.3 Multiple Sensors Approach	111
7.3.1 Use of Standard Averaging Techniques	112
7.3.2 Use of Kalman Multi-Sensor Processing Techniques	113
7.4 Other Observations	115
8 Contributions & Future Work	116
8.1 Contributions	116
8.2 Future Work.....	116
Conclusion	119
Bibliography	121
Appendix-A (RPES Components Specifications)	124
Appendix-B (C-Code for the Micro-Controller AT90USB1287)	129
Appendix-C (MATLAB Code)	135

List of Tables

Table 1-1: Summary of the Boundary Conditions	3
Table 2-1: Forms of Energies Converted by a Sensor.....	19
Table 5-1: Table of Processing Techniques	59
Table 6-1: Results of the Raw Processing Techniques and the Best and Worst Technique (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).	96
Table 6-2: Results of the Best Raw Processing Techniques.	96
Table 6-3: Results of the Moving Average Processing Techniques with 11-Sample Size (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).	96
Table 6-4: Results of the Moving Average Processing Techniques with 21-Sample Size (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).	97
Table 6-5: Results of the Moving Average Processing Techniques with 41-Sample Size (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).	97
Table 6-6: Results of the Best Moving Average Processing Techniques.	98
Table 6-7: Results of the Kalman Single Processing Techniques (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).	98
Table 6-8: Results of the Best Kalman Single Processing Techniques.....	99
Table 6-9: Summary of the Single Sensor Approach (All Experiments are 1800 Seconds Duration).....	99

Table 6-10: Results of the Multi-Sensor Processing Techniques (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).	100
Table 6-11: Results of the Kalman Multi-Sensor Processing Techniques (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).	100
Table 6-12: Summary of Multi-Sensor Approach (All Experiments are 1800 Seconds Duration).....	101
Table 7-1: Results of the Best Raw Processing Techniques.	103
Table 7-2: Comparison between the Moving Average Processing Techniques with 11-Sample Size	105
Table 7-3: Comparison between the Moving Average Processing Techniques with 21-Sample Size	105
Table 7-4: Comparison between the Moving Average Processing Techniques with 41-Sample Size	106
Table 7-5: The Comparison between the Moving Average with Different Sample Sizes	107
Table 7-6: Comparison between the Kalman Single Processing Techniques	108
Table 7-7: Summary of the Single Sensor Approach (All Experiments are 1800 Seconds Duration).....	109
Table 7-8: Comparison Between the Short-Duration and the Long-Duration Experiment	111
Table 7-9: Comparison between the Multi-Sensor Processing Techniques (All Experiments are 1800 Seconds Duration).	112

Table 7-10: Comparison between the Kalman Multi-Sensor Processing Techniques (All Experiments are 1800 Seconds Duration).	113
Table 7-11: Summary of Multi-Sensor Approach (All Experiments are 1800 Seconds Duration).....	113
Table A-1: The AT90USBKey Components and Description [45]	125
Table A-2: The RPES Components Description	127

List of Figures

Figure 1-1: The Construction Model of Patent No.: US 6882964 B2 [20]	8
Figure 1-2: (a) The Device Overview of Patent No.: PCT/EP2005/056589. (b) The Working Principle of the Same Device [21]	9
Figure 2-1: The NAVSTAR Global Positioning System Constellation of 24 Earth- Orbiting Satellites [1][“Auto GPS Navigation.”]	13
Figure 2-2: The Trilateration Principle [M. Brain and T. Harris, “HowStuffWorks ‘3-D Trilateration’.”]	15
Figure 2-3: The GPS Time Propagation Measurement [23]	16
Figure 2-4: Initial MEMS Devices [“GREINER: What are Microsystems / MEMS?”][27]	18
Figure 2-5: MEMS Accelerometers [29]	22
Figure 2-6: Capacitive Accelerometers [“Introduction to Microelectromechanical Systems (MEMS) Compliant Mechanisms.”]	23
Figure 2-7: Piezoresistive Accelerometers [30]	24
Figure 2-8: Principal of the Gyroscope [8]	25
Figure 3-1: IMU (a) In Three Dimensions. (b) In Two Dimensions [31]	27
Figure 3-2: Tilt Sensor Output	27
Figure 3-3: Diagram of the Linear and Angular Data Combination [8]	28
Figure 3-4: Wii-mote Consoles [“Wii Console Official Factsheet.”]	28
Figure 3-5: iPhone Video Game [C. Liu, “The MEMS Motion Sensor Perspectives.”] ..	29
Figure 3-6: System Nonlinearity [32]	30
Figure 3-7: Systematic and Random Error [32]	31

Figure 3-8: Ideal and Typical Circuits.....	33
Figure 3-9: Relative Magnitudes of Signal and Noise [33].....	35
Figure 3-10: Filter Response [“Cutoff frequency - Wikipedia, the free encyclopedia.”] ..	37
Figure 3-11: Low Pass Filter	37
Figure 3-12: High Pass Filter	38
Figure 3-13: Different Types of Filters	39
Figure 3-14: Typical Kalman Filter Application [35]	40
Figure 3-15: Kalman Filter Equations [37]	43
Figure 4-1: The RPES Prototype Device Layout with Single Accelerometer Module	46
Figure 4-2: The RPES Prototype Device Layout with Four Accelerometer Modules	48
Figure 5-1: Plot of Measured and Actual Acceleration.....	55
Figure 5-2: Plot of Measured and Actual Velocity	55
Figure 5-3: Plot of Measured and Actual Displacement	56
Figure 5-4: Plot of Measured and Actual Acceleration and the Kalman Filter Output....	57
Figure 5-5: The Error Covariance	61
Figure 5-6: Close-up of the Error Covariance	62
Figure 5-7: Close-up of the White Noise Error in the First 45 Seconds	63
Figure 5-8: Plot of Measured and Actual Acceleration and the Kalman Filter Output after the First 20 Seconds of data were removed.....	63
Figure 5-9: Close-up of the White Noise Error after the First 20 Seconds were removed	64
Figure 5-10: Plot of Measured and Actual Acceleration, the Kalman Filter and Two Boundary Conditions Applied.....	65

Figure 5-11: Plot of Measured and Actual Acceleration and the Kalman Filter Output after applying all the Techniques.....	66
Figure 5-12: Plot of Measured, Actual and the Kalman Filter Velocities after applying all the Techniques	67
Figure 5-13: Plot of Measured, Actual and the Kalman Filter Displacements after applying all the Techniques	68
Figure 5-14: Plot of Measured, Actual and the Kalman Filter Velocities after applying all the Techniques and the Third Boundary Condition was applied on the measured and the Kalman Filter Velocities.....	69
Figure 5-15: Plot of Measured, Actual and the Kalman Filter Displacements after applying all the Techniques and the Third Boundary Condition was applied on the measured and the Kalman Filter Velocities.....	69
Figure 5-16: Plot of Measured and Actual Acceleration, and the 11-Sample Size Moving Average Filter output.....	71
Figure 5-17: Plot of Measured and Actual Acceleration, and the 21-Sample Size Moving Average Filter output.....	71
Figure 5-18: Plot of Measured and Actual Acceleration, and the 41-Sample Size Moving Average Filter output.....	72
Figure 5-19: Plot of Measured and Actual Acceleration, and the Kalman filter and 11- Sample Size Averaging output	74
Figure 5-20: Plot of Measured and Actual Acceleration, and the Kalman filter and 21- Sample Size Averaging output	74

Figure 5-21: Plot of Measured and Actual Acceleration, and the Kalman filter and 41-	
Sample Size Averaging output	75
Figure 5-22: Plot of Measured Acceleration of Four Accelerometers	77
Figure 5-23: Plot of Measured Acceleration of the Four Accelerometers after applying the	
Boundary Conditions	77
Figure 5-24: The Average of the Acceleration Readings	78
Figure 5-25: The Average of the Four Filtered Accelerations	79
Figure 5-26: The Average of the Four Velocities	80
Figure 5-27: The Average of the Four Displacements	82
Figure 5-28: Parallel Design Output Vector	83
Figure 5-29: The Error Covariance Matrix	84
Figure 5-30: The Parallel Design Diagram	86
Figure 5-31: The Parallel Design Acceleration	87
Figure 5-32: The Parallel Design Displacement	87
Figure 5-33: The Series Design Diagram	88
Figure 5-34: The Series Design Acceleration	90
Figure 5-35: The Series Design Displacement	91
Figure 5-36: The Compression Design Diagram	92
Figure 5-37: The Compression Design Acceleration	93
Figure 5-38: The Compression Design Displacement	94
Figure A-1: AT90USBKey [45]	125
Figure A-2: The Bluetooth Module (left) and Dongle (right) [46][47]	125

Figure A-3: The Dual Axis ADXL203 Accelerometer [“ADXL203EB Eval Bd $\pm 1.5g$ Dual Axis ADXL203 Accelerometer eBay.”]	126
Figure A-4: The Connection Components of AT90USBKey [45]	126
Figure A-5: The Voltage Vs. Temperature Graph [48]	128
Figure A-6: The Sensitivity Vs. Temperature Graph [48]	128

Acknowledgments

I would like to express my profound gratitude to Dr. Nikolai Dechev for his supervision, knowledge, guidance, advice, and inspiration for me, my work and other life aspects. His sage advice, insightful criticisms, and patient encouragement aided the writing of this thesis in innumerable ways. I am delighted to involve in this interesting project, where I enhanced my knowledge and understanding in a lot of areas.

I also would like to thank Ed Haslam, for his help on the hardware programming and whose steadfast support of this project was greatly needed and deeply appreciated.

Moreover, I would like to thank Patrick Chang for letting me access the mechatronics lab and further investigate the micro-controller and giving me practical advices and recommendations for the work.

I would like to thank and acknowledge my friends and families for giving me the support, encouragement, and motivation for the completion of this work.

Dedication

To my lovely, warm-hearted and supportive wife, who was with me all the time in my journey, who had stood with me in the difficult times and who had been a great source of motivation and inspiration.

To my Mother, brothers, and sisters who always were there for me and gave me an extra supports all the way since the beginning of my studies and who offered me unconditional love and support throughout the course of this thesis.

In the memory of my late father, who did not have the chance to share my accomplishment with me.

Finally, I would like to express my grateful gratitude and thankfulness to the Saudi Arabian Ministry of Higher Education for the financial support for my program and for their unlimited encouragement and assistant for me and the family throughout my Master program.

1 Introduction

1.1 Background and Motivation

A position estimation system is a system that determines the position and orientation of an object in space. There are various position estimation systems available. Some are able to cover worldwide ranges such as the global navigation satellite systems (GNSS) [1], while others go down to workspace systems such as Wii remote bar [2]. With any position estimation system, the main concern is the error generated from the system, which could vary from millimeters up to meters, and can accumulate over time.

The objective of this thesis is to develop a novel, relative position estimation system (RPES), based on micro-electro mechanical (MEMS) inertial sensors (accelerometers and gyroscopes). The RPES is envisioned to be a hand-held device about the size of a common television remote control, that can record it's own translation and rotation as it is moved through space, without the need of external references. The MEMS inertial sensors provide the translation and rotation information, and this information is further processed to create an estimate of the total distance and orientation traveled through space, from a designated starting point, to a designated end point. It is envisioned that such a device could be useful for surveying purposes, especially inside buildings, underground tunnels, mines or parking lots. In addition, such a device could be useful for building construction for determining wall flatness, perpendicularity of two surfaces, or other geometric configurations, by measuring several points on those surfaces, and post-processing the results.

The research methodology is as follows: The proposed RPES was developed with the inertial sensors in a static (motionless) arrangement. This approach was useful because it provides a 100% accurate reference (i.e. no displacement), against which the estimated displacement and orientation of the RPES can be compared. Also, it was done to evaluate the sensors' noise stated in the data sheet, and to estimate the impact of atmospheric changes such as temperature and pressure on the sensors' readings. For this case, the measured acceleration signal from the inertial sensors is digitized, and then mathematically integrated to get the angular rate estimate, and integrated again to get a displacement estimate. The real challenge of this approach is that the accelerometer measurement contains a considerable amount of noise, in addition to the true acceleration signal, and the measured signal is a combination of both. The noise is any undesired signal that is generated either from the electronic components themselves or from external sources, such as: Temperature, pressure, vibration, etc. It is impossible to separate or recognize the noise in the signal from the true acceleration signal. Throughout the measurement period, the noise in the signal varies in an unpredictable/non-repeatable way. Therefore, when this mixed noise and acceleration signal is double integrated to get the displacement estimate, there can be a considerable error in estimated displacement due to the integration of the noise. The longer the measurement period, the more error in displacement estimate, because more noise is integrated. For these reasons, developing a positional measurement device based on these principles is a challenging subject.

Therefore, one of the main goals of this research is to find ways to increase the signal-to-noise ratio as much as possible. To do this, a number of different methods were investigated:

- 1- Application of boundary conditions specific to the proposed RPES device application.
- 2- Use of advanced filtering techniques, such as: Kalman filters.
- 3- Investigate the use of redundant sensors per axis, and combining their output with the Kalman filter, to determine if noise can be reduced.

It is envisioned that the proposed RPES device will operate as follows: The device starts from one stationary point in space (point A), is then moved/rotated through space to the end point in space (point B), where it stops. Therefore, it is assumed that the initial and final acceleration of the handheld device, along with the initial and final velocity are zero. Additionally, since this is a “relative” measurement, it can be assumed the initial position is zero. These five assumptions form the basis of five boundary conditions that are used during the signal processing steps of this application and they are summarized in Table 1.1.

Name	Code	Description
Initial acceleration rotation	R_i	Rotating acceleration data around the final point to zero the first point
Final acceleration rotation	R_f	Rotation acceleration data around the first point to zero the final point
Initial velocity rotation	VR_i	Rotating velocity data around the final point to zero the first point
Final velocity rotation	VR_f	Rotation velocity data around the first point to zero the final point
Initial displacement rotation	DR_f	Rotating displacement data around the final point to zero the first point

Table 1-1: Summary of the Boundary Conditions

Additionally, other signal processing techniques were used. Kalman filters were investigated because they have been reported extensively for use in navigation and object tracking applications [3][4]. In addition, they have a good ability to follow rapid changes in the input parameter. Therefore, they have a good dynamic linearity. Interestingly, Kalman filters have also been investigated for combining measurements from redundant sensors to minimize the variance of the signal error [3], in other applications. Therefore, one research goal is to develop the best way to merge multiple, redundant information to provide the best estimate of displacement among the methods proposed in this work.

1.2 Literature Review

Positioning estimation systems are widely used in various navigation and surveying purposes. A number of positioning system technologies have been developed to accommodate the need for higher accuracies and lower costs. These various systems will be described in the following paragraphs.

1.2.1 Global Positioning Systems (GPS)

GPS is one of the most popular positioning and navigation systems used in the last 20 years, and is very accurate in positioning and timing estimation [5]. However, it is dependent on the use of a network of line-of-sight links to satellites, and becomes blind when the satellite signals are blocked by roofs, walls or other natural obstructions. It is also ineffective in some areas such as large buildings, under ground parking areas, mines, or tunnels.

1.2.2 Indoor Positioning System (IPS)

The IPS based systems make use of radio, ultrasonic or infrared signals for location estimation within indoor places [6]. The main idea is that an energy beam in the form of radio, ultrasonic, infrared, or other energy is emitted, reflected off of surrounding objects, and received by the system. Based on the time elapsed by the beam travel, the distance between the transceiver and the object is evaluated. The disadvantages are loss of the signals due to obstructions, unwanted signal reflections that cause false signals, and interference caused by high frequency [7].

1.2.3 Inertial Navigation System (INS)

An INS depends on inertial measurements for navigation. In the past, such systems were based on high-cost mechanical gyroscopes such as those used in ships, submarines, or rockets [8]. Recently, MEMS accelerometer and MEMS rate-gyro technology has replaced old-style mechanical gyroscopes. Accelerometers are now used to measure the acceleration, and rate-gyros to measure the angular rate, as the components of modern INS. The main advantage of these systems is that there is no need for any external reference. However, precise measurement of the acceleration signal, very low signal noise, and precise time measurement are all critical for reasonable accuracy. The proposed RPES of this work is a variant of an INS.

1.2.4 Existing Methods for Position Estimation

There has been significant research conducted by others in different applications, to combine inertial sensors with other navigation techniques. This combination attempts to overcome the disadvantages that each technique has and reduce the error. For example, inertial sensors have been combined with the GPS systems for localizing vehicle position [9][10][11] when the GPS signal is lost, such as in a tunnel. It is important to explain the concept of “relative vs. absolute” localization. Relative localization schemes acquire information from onboard the vehicle, using INS sensors (gyros, accelerometers, etc....). They have a high acquisition frequency, however, they also have a high error growing with time. Absolute localization acquires information from an external source, and has accuracy subject to the transmitter signal strength, but is subjected to lose the signals. Kalman filters can be used and designed to combine these two types of information and compensate for most of these errors to yield a better estimate [9]. Another example is an underwater remotely operated vehicle (ROV) used for inspection and operation at depths. Their navigation system contains a number of sensors including: accelerometers, rate gyros, inclinometers, Doppler velocity sensors, and others, to provide information about the position [12]. Inertial sensors can also be combined with wireless technologies such as: ultrasonic, infrared or radio signals for position estimation [7]. The properties of visual and inertial sensors have also been exploited in robot navigation and tracking for egomotion estimation [12]. Andrey Soloviev has developed navigation in an urban environment by combining the features of an INS sensor, a laser scanner, and the Global positioning system (GPS) [13][14]. The performance of low-cost MEMS accelerometers to be used for distance measurement for a short duration with the use of a Kalman filter

has also been evaluated [15]. The use of inertial sensors for short time durations to estimate position, while the GPS system is not available/reliable was discussed in [16]. The position of a mobile platform was evaluated by low-cost accelerometers and signal processing using double integration of the acceleration with time [17].

1.2.5 Error Estimation with Inertial Navigation Sensors (INS)

A study to investigate the position estimation error caused by noise and drift has been carried out, and methods to calibrate the sensor before use have been proposed in [18]. Some studies have discussed the mechanical - thermal noise with the use of pull-in time to remove the noise of the sensor circuits [19].

One of the methods investigated in this thesis is to find an approach to appropriately select a mode from a bank of Kalman filter [12]. This can be done by designing a Kalman filter algorithm to accommodate multiple combinations of sensors.

1.3 Patents Review

1.3.1 High Accuracy Inertial Sensors from Inexpensive Components

This patent was invented by David S. Bayard and Scott R. Ploen from the California Institute of Technology with a patent No.: US 6882964 [20]. Their idea came from a study that combined the time from three watches relative to each other, and compared the result to a universal time to get an improved time estimate. The idea of their patent was to combine the output of a number of rate sensors into a single reading, which should improve the performance over any individual rate sensor and produce a minimum

variance rate estimate. The minimum-variance gain matrix filter plays an important role in describing how the combined measurements are smoothed and weighted in respect to each other. The invention makes use of the Kalman filter for combining the readings by solving the Riccati differential equations (RDE) matrix, which is able to specify the gain matrix of the optimal filter. An essential contribution of their research is to analytically solve the differential equations of the virtual gyro Riccati.

Basically, a matrix of rate random noise and angle random noise is to be constructed, the matrices are combined into what is called a Multiple Gyro Mess, and then the Kalman filter is applied. The Kalman filter is a recursive-processing filter that derives a higher accuracy output signal using the characteristics of the rate sensors' noise and the statistical correlation.

Fig. 1.1 shows the construction model of the patent.

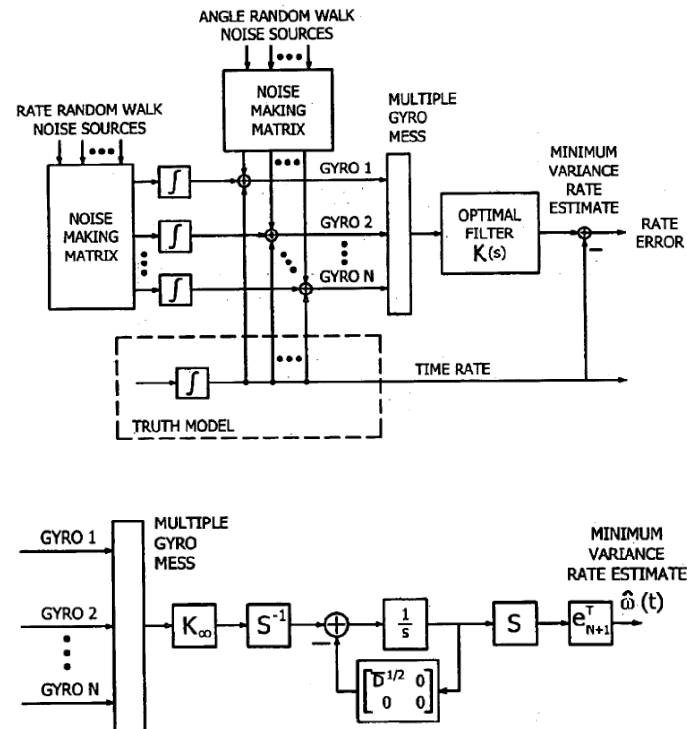


Figure 1-1: The Construction Model of Patent No.: US 6882964 B2 [20]

1.3.2 Hand-Held Surveying Device and Surveying Method for Such A Surveying Device

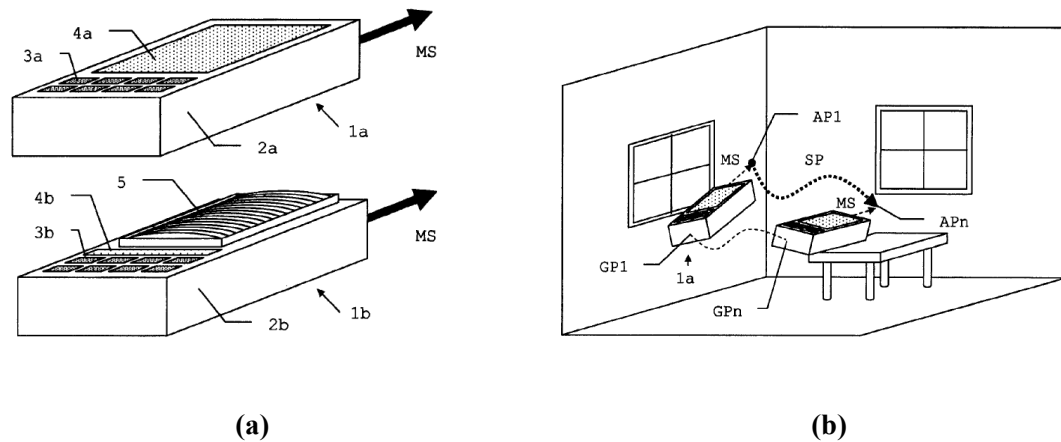


Figure 1-2: (a) The Device Overview of Patent No.: PCT/EP2005/056589. (b) The Working Principle of the Same Device [21]

Knut Siercks has invented patent No.: PCT/EP2005/056589 [21]. The idea proposed is a hand-held measurement device, where it is composed of a rangefinder to provide a measurement distance and a positional detection component to measure the position of the device and the orientation of the rangefinder. The device operates by generating successive signals containing information about the current position and orientation of the device, and it can also be used for surveying purposes.

Not much information could be disclosed from this patent, since all the available documents are just about the patent claims. Furthermore, no scientific publications about this patent were found that could be helpful for getting more information.

Fig. 1.2 (a) is an overview of the device, and (b) is the device working principle.

1.4 Thesis Overview

This thesis was outlined as follows. Chapter 2 gave an insight about the background of the measurement tools and how it evolved. Then it discussed in some details the most popular positioning systems available, which is the GPS. An introduction to MEMS and micro-systems was also talked about by introducing the inertial sensors and how they were used for navigation systems. The noise errors were explored as well and finally some deliberation about different types of filters and the Kalman filter in particular.

Chapter 3 discussed the RPES device overview, the specification of the components and the software utilized. Then it gone through the device layout and the experiment set up.

Chapter 4 was about the lab methodology. First it presented the single sensor experiment, different technique was studied to investigate the performance, and then the multiple sensors approach was explored, where different Kalman filter designs were examined.

In chapter 5 the results of the experiments were shown and in chapter 6 these results were observed and discussed. Other observations were discussed as well.

Finally, some works for the future was suggested and then the thesis was concluded.

2 Positioning Systems

2.1 Measurements Tools Overview

Since the old ages, people have used common techniques for measurements such as, pacing or hands, where the number of steps determines the distance. Also, simple devices such as a wheel with a known diameter were used to measure the distance, by counting the number of rotations. Other devices were also used such as: chains, guessing, or direct or indirect ranging rods. Other standard methods of measuring distances or linear measurements in ancient times were chains and tapes. At the same time people also wanted to measure angles, so they discovered the compass, which was a very simple instrument that worked on the principle of a magnetic needle that aligned itself to the Earth's magnetic field as a basis to measure angles. There was also theodolite, which used in angle measurements in the horizontal and vertical planes in a more sophisticated and accurate way.

In more modern times, people required more accurate measurements; therefore, new methods to standardize the measurements were established.

Scientists started converting instruments from conventional mechanical devices into electronic instruments. One such example is the EDM (Electronic Distance measuring instrument), which basically fires an electro-magnetic pulse with a velocity (v) of about $300 \times 10^6 m/sec$ and the distance was calculated by:

$$D = v \times t \quad (2.1)$$

After that, scientists have come up with Aerial photogrammetry, which was about taking photographs of the ground from the air. It was first used for spying purposes in

World War 1 and 2, where cameras were set up on pigeons. Later on, people got used of Balloons and Aircrafts, they flew on a certain altitude depending on the kind of application, then an image was captured, after a short time, another image was captured, in which the two images should be overlapping. A 3-dimensional model diagram could be generated of the overlapped area by putting the two captured images together in stereo pair to create an illusion of depth and give the perception of 3-D depth. These images were then used for measuring distances, angles, coordinates and so on. However this method was costly and not possible in bad weathers. Then Satellite remote sensing was discovered. It started commercially in 1972 with a Satellite called "Landsat" from the United States that was available for civil users. This method was Distinct by its availability all the time and any where on the earth, in addition to that it got a synoptic view in several wave bands. Nowadays, the most prominent modern tool for measuring was the GPS (Global Positioning System).

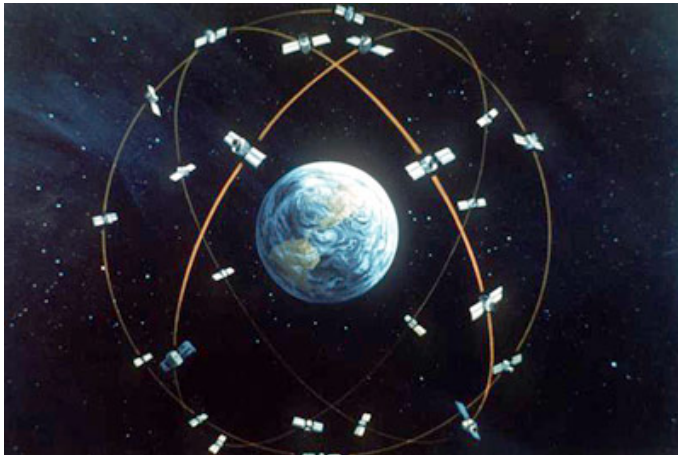
2.2 Global Positioning System (GPS)

2.2.1 GPS Overview

The Global Positioning System is a worldwide system that is used to locate one's position, anywhere on the earth's surface. It is available all the time, where the signals from the Satellite cannot be disrupted or blocked by the clouds or bad weather.

In 1978, the US department of defense launched the first satellites of the NAVSTAR Global Positioning System, for military purposes. Between 1989 and April 27th of 1995, the initial GPS system of 24 satellites was launched and became fully operational, as

shown in Fig. 2.1. Now, the GPS system is the most dominant technology in surveying, localization and positioning estimation.



(a)



(b)

Figure 2-1: The NAVSTAR Global Positioning System Constellation of 24 Earth-Orbiting Satellites [1][“Auto GPS Navigation.”]

2.2.2 GPS Working Principles

The GPS system is composed of three segments: the user segment, the space segment and the control segment. The user segment is composed of an antenna, which is adjusted to match the frequencies of the satellites, a processor, and a crystal oscillator clock. The space segment consists of a constellation of 24 satellites (there are extra satellites in case one fails). Each of these satellites travels at an altitude of 20,200 km, making two complete orbits around the globe everyday. The satellites orbits are arranged in such a way that anytime, everywhere, there should be 3 or more satellite visible in the sky. The third segment is the control segment, which consists of a master control station at Schriever Air Force Base in Colorado Springs, four ground antennas, and six monitor

ground stations placed in different places around the world. The monitoring stations track the path of the satellites, then the tracking information is sent to the master control, then each satellite is contacted regularly for updating, by synchronizing the satellite atomic clock to within few seconds of each other and to correct the satellite ephemeris constant, which is done by a Kalman Filter. The GPS system idea is that the user segment locates three or more satellites and figures out the distance to each of them, then based on a mathematical principle called trilateration, the GPS can locate one's position.

2.2.3 Trilateration

Trilateration is a process to determine the relative locations of points by measurement of distances, using the geometry of spheres or triangles [22]. When determining the distance to a satellite, it means that (A) position could be anywhere on the surface of a sphere with a radius equal to the distance between (A) and the satellite. Then when determining the distance between (A) and another satellite, another sphere with a radius equal to the distance between (A) and the second satellite will intersect with the first sphere in a circle, which means that (A) position could be anywhere on that circle. Finally, when determining the distance between (A) and a third satellite, another sphere will intersect with the circle in two points, one is in the space and one is on the surface of the earth. The earth is considered as the fourth sphere and since it intersects with one of the points that point will be considered as the location of (A) and the other point in the space will be neglected, as shown in Fig. 2.2.

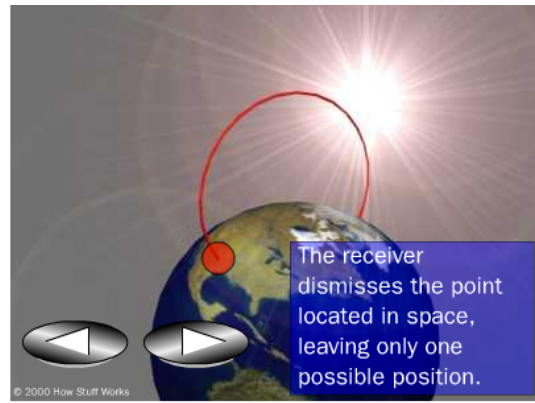
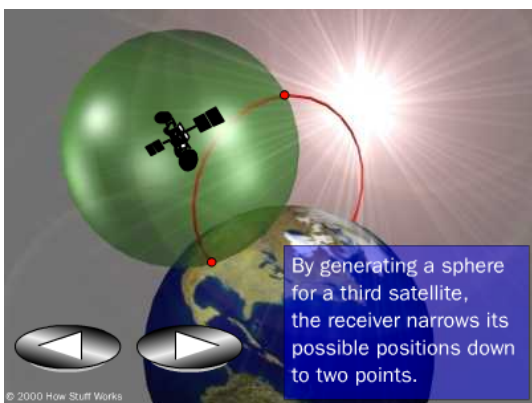
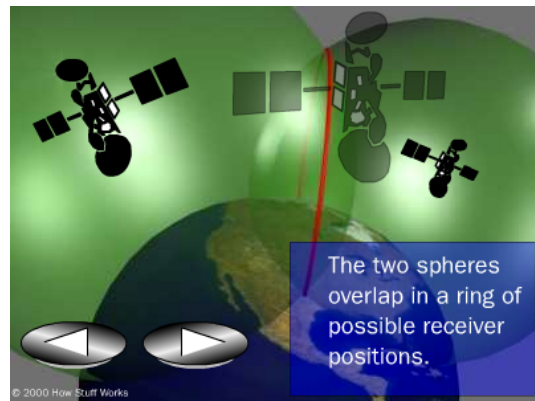
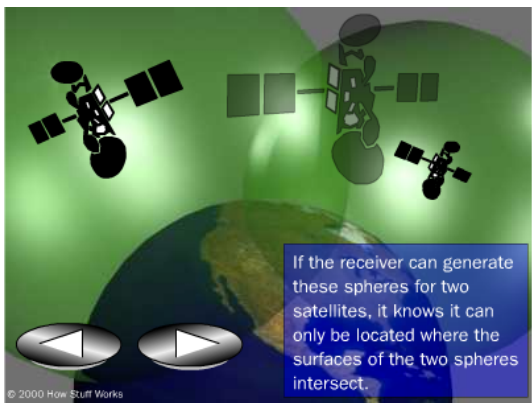
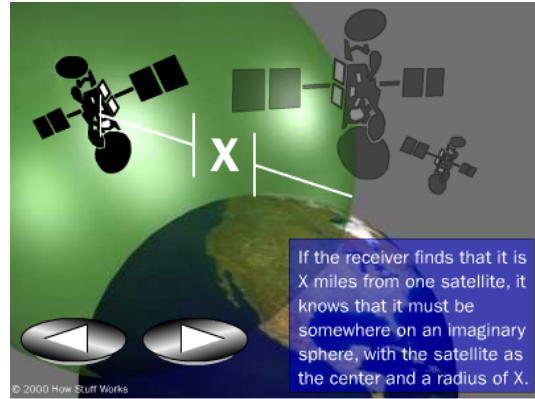
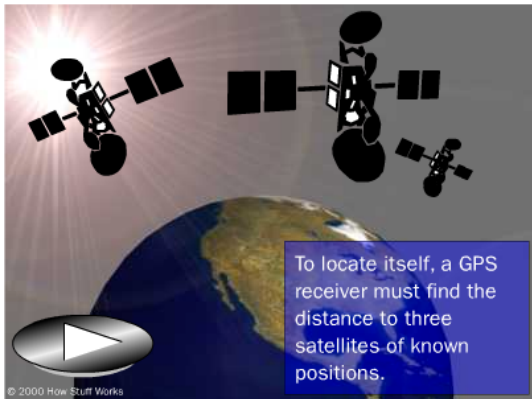


Figure 2-2: The Trilateration Principle [M. Brain and T. Harris, “HowStuffWorks ‘3-D Trilateration’.”]

2.2.4 GPS Distance Measurements

The distance between the space segment to the user segment is calculated based on the satellites' transmitted signals time of arrival TOA. The satellite's atomic clock sends signals in the form of a digital pattern code called the Pseudo-random number (PRN), which is a sequence of ones and zeros, at specific times. The GPS user segment's internal clock starts running the same digital code at exactly the same times. When the two digital codes are compared, the satellite's code will lag a bit behind the receiver's code. The length of the delay is considered the time the signal transmitted from the satellite takes to reaches the GPS receiver and when it is multiplied by the speed of light - 299,792,458 meters per second - the distance between the user's end and the satellite's end is determined.

Fig. 2.3 illustrates the GPS time propagation measurement.

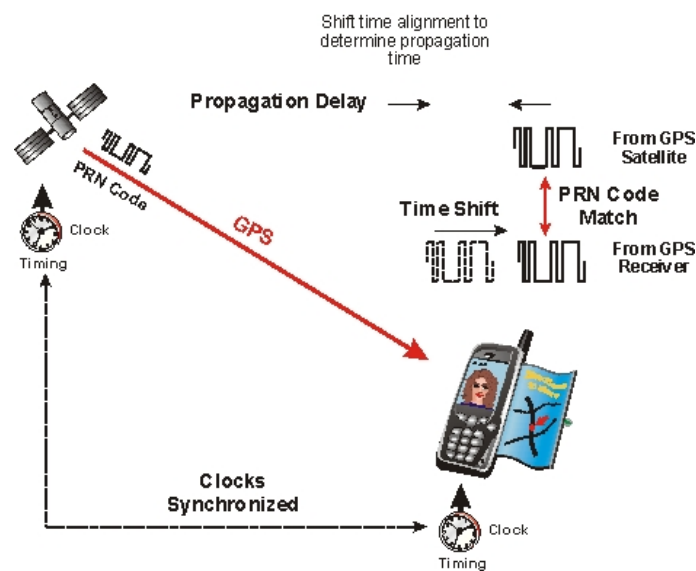


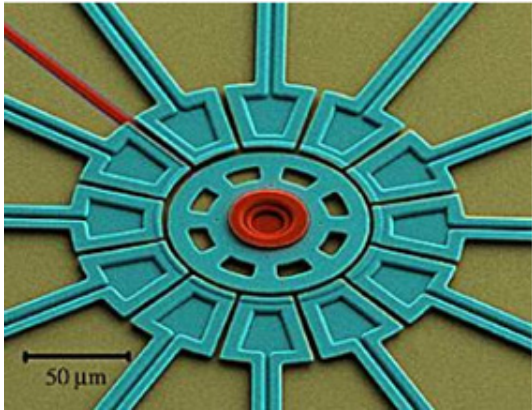
Figure 2-3: The GPS Time Propagation Measurement [23]

2.2.5 Sources of Error in GPS

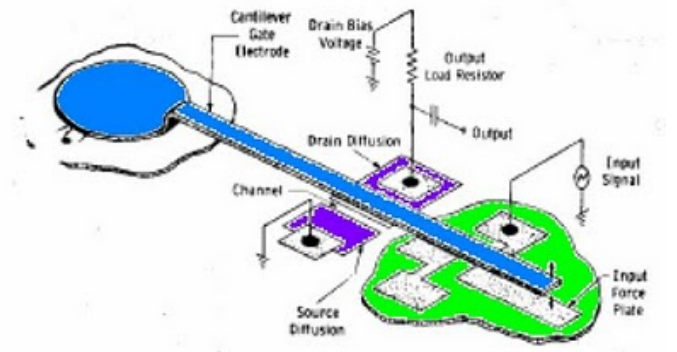
Every satellite in the GPS has a very precise atomic clock, - with a resolution down to nanoseconds. In order to make a match perfect, every GPS receiver should also have an atomic clock, which is impossible due to the high price of the atomic clock, which would make it too expensive for the consumer. Another error arises from the inaccuracy of the transmission time, due to the slowing of signal speed through the atmosphere. This will cause inconsistency, or sometimes the satellite sends bad data, which causes an incorrect info of the satellite's current position.

2.3 MEMS and Microsystems

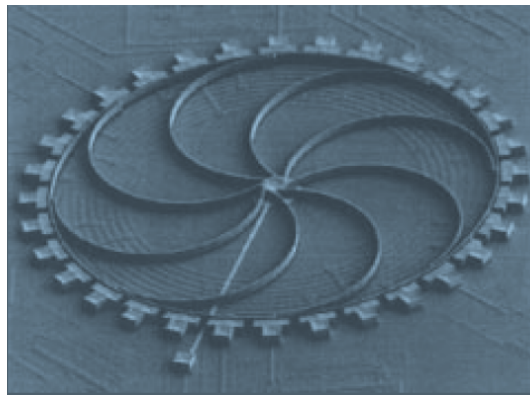
MEMS or Micro-electromechanical systems are devices that range in size from a few millimeters to a few micrometers. MEMS devices are composed of various sub-systems, which can be micro-sensors, micro-actuators, other micro-mechanical structures. These MEMS sub-systems are all integrated on a single silicon chip, which is connected to external systems such as electronics for power, signal processing chips to process data, batteries, and displays. Richard Feynman (Nobel Prize in physics) was one of the first who observed the possibilities of utilizing micro and nano-scaled devices [24]. One of the first attempts for MEMS was the development of the Resonant Gate Transistor, which was introduced in 1967 [25], and is shown in Fig. 2.4(b). The first MEMS micromotor was demonstrated in 1989, and shown in Fig. 2.4(a) [26]. The development of the commercial MEMS based accelerometer analog device was in 1993 as in Fig. 2.4(c) [27].



(a)



(b)



(c)

Figure 2-4: Initial MEMS Devices [“GREINER: What are Microsystems / MEMS?”][27]

MEMS devices have risen in importance, because they have distinctive features:

- **Miniaturization:** the micro-scale size provides some advantages, which include handling micro-objects, higher and better sensitivity, linearity and responsivity, and lower power consumption.
- **Lower cost:** Following on the principles of microelectronic mass production, a reduction in MEMS cost may be realized for some applications.
- **Integration with Microelectronics:** A whole system can be created by integrating microelectronic control with MEMS sensor and actuators

2.3.2 Sensors

Sensors are derived from a Latin word “Sentire” which means, “to perceive.” They are devices that can measure an input signal or energy that cannot be observed by our senses, and convert that information into an appropriate output signal or energy. There are different forms of energies converted by a Sensor, as shown in Table 2.1:

Energy Form	Types of Conversion
Atomic Energy	The force between nuclei and electrons
Electrical Energy	Related to electric field, current, voltage, etc....
Gravitational Energy	Pertains to the gravitational attraction between a mass and the earth
Magnetic Energy	Deals with magnetic field, etc....
Mass Energy	As described by the equation $E = mc^2$, where c is the speed of light, which is equal to 299 792 458 m/s
Mechanical Energy	Related to the motion, displacement, force, etc....
Molecular Energy	The binding energy in molecules
Nuclear Energy	The binding energy between nuclei
Radiant Energy	Related to electromagnetic radio-waves, microwaves, infrared, visible light, ultraviolet, X-rays, and gamma rays
Thermal Energy	Related to the kinetic energy of atoms and molecules

Table 2-1: Forms of Energies Converted by a Sensor

Each of these signals has a corresponding signal associated with it, where there are chemical, electrical, magnetic, mechanical, radiant and thermal signals [28].

2.3.2.2 Sensor Characteristics Definitions

There are a lot of parameters that can alter/affect the characteristics of a sensor. These parameters can impact the performance/effectiveness and the cost of a sensor, and include:

- Ambient conditions: Operating environment conditions such as temperature, vibration, pressure, shock, and moisture.
- Linearity: Closeness between the calibration curve and a specified straight line.
- Warm-up time: The time between applying sensor power and the moment when the sensor can operate within its sensing accuracy.
- Saturation: At some levels of input stimuli, the output signal of a sensor will no longer be responsive and the sensor is said to be saturated.
- Offset: The output of a sensor with zero measurand applied at nominal ambient conditions.
- Operation life: Minimum length of time over which the sensor will operate without changing performance characteristics beyond specified tolerance.
- Overload characteristics: Maximum magnitude of a measurand that can be applied to a sensor without causing a change in performance beyond specific tolerances.
- Selectivity: Ability of a sensor to measure one measurand in presence of others.
- Sensitivity: Ratio of change in sensor output to the change in the value of the measurand.
- Speed of response: Time constant, τ , at which the output reaches 63% of its final value in response to a step change in the measurand.
- Repeatability: Ability of a sensor to reproduce output readings under identical condition of the measurand.
- Stability: Ability of a sensor to maintain its performance characteristics for a certain period of time [28].

2.3.3 Inertial Sensors

One of the most important types of MEMS sensors is the inertial sensor for measuring linear acceleration and angular velocity. MEMS accelerometers alone have the second largest sales volume after MEMS pressure sensors. The accelerometer market has been estimated to be \$609 million in 2005, and has grown rapidly since that time. Gyroscopes measure rate of angle of rotation and they will attract more attention in the next few years.

Nowadays, MEMS inertial sensors are used in many different applications, and they can be found everywhere around us including:

- Automobile air bag deployment systems
- Active stabilization of the picture in camcorders and cameras
- Video game and entertainment systems
- Three-dimensional mouse
- Various sports equipment
- Biomedical applications for activity monitoring
- Industrial applications; robotics, machine and vibration monitoring
- Tracking and monitoring mechanical shock and vibration during transport and handling of a variety of equipment and goods

2.3.3.1 Accelerometers

Accelerometers can be used for static or dynamic measurements. This includes static measurements such as gravitational force for tilt and inclination, and dynamic measurements such as: vibration and shock, or velocity and acceleration.

The operational principle of an accelerometer can be described as a simple spring-mass system, as shown in Fig. 2.5. Their design normally requires a proof mass (seismic mass), an elastic spring, the device frame (case), and a method to measure the displacement of the proof mass with respect to the device frame. Fig. 2.6 shows a simple accelerometer moving with a side ways motion.

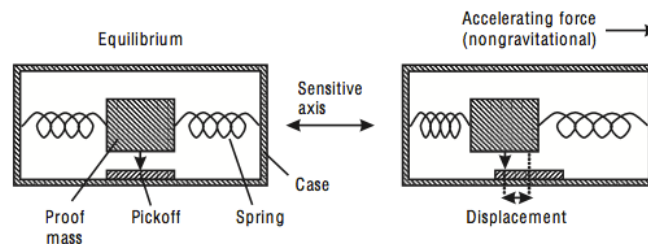


Figure 2-5: MEMS Accelerometers [29]

The proof mass will experience an inertial force as a result of an acceleration or deceleration of the device frame. The elastic spring that mechanically supports the proof mass will attempt to restore the mass to its neutral position after the acceleration is removed. The movement of the proof mass is read by the pickoff. Sometimes, a damper is added to the design to control the motion of the proof mass and to obtain favourable frequency response characteristics. If an accelerometer is allowed to free fall, it reads zero because the gravity pulls both the proof mass and the device frame at the same time, so there is no relative movement between the mass and the frame. Gravity can be measured by an accelerometer since the proof mass will be pulled down in relation to the frame.

There are two main sensory principles that are used to detect the proof mass's movement in conventional accelerometers. These are, piezoresistive accelerometers,

which depend on the change in resistance due to strain in the springs supporting the proof mass, and also capacitive accelerometers, which depend on the change in capacitance between the proof mass and frame.

2.3.3.1.1 Capacitive Accelerometers

Fig. 2.6 shows a capacitive accelerometer, which consists of an inertial mass, flexible springs, and stationary interdigitated fingers. Typically, when the device is subjected to acceleration, the inertial mass, suspended by 4 anchors, moves accordingly, governed by the equation ($F = ma$). This relative motion will cause the interdigitated fingers attached to the proof mass to move towards or away from the stationary fingers on the frame, and would cause the spacing between the fingers to change. This change in the spacing between the interdigitated fingers is detected as a change in capacitance therefore, the total amount of capacitance changes according to the movement.

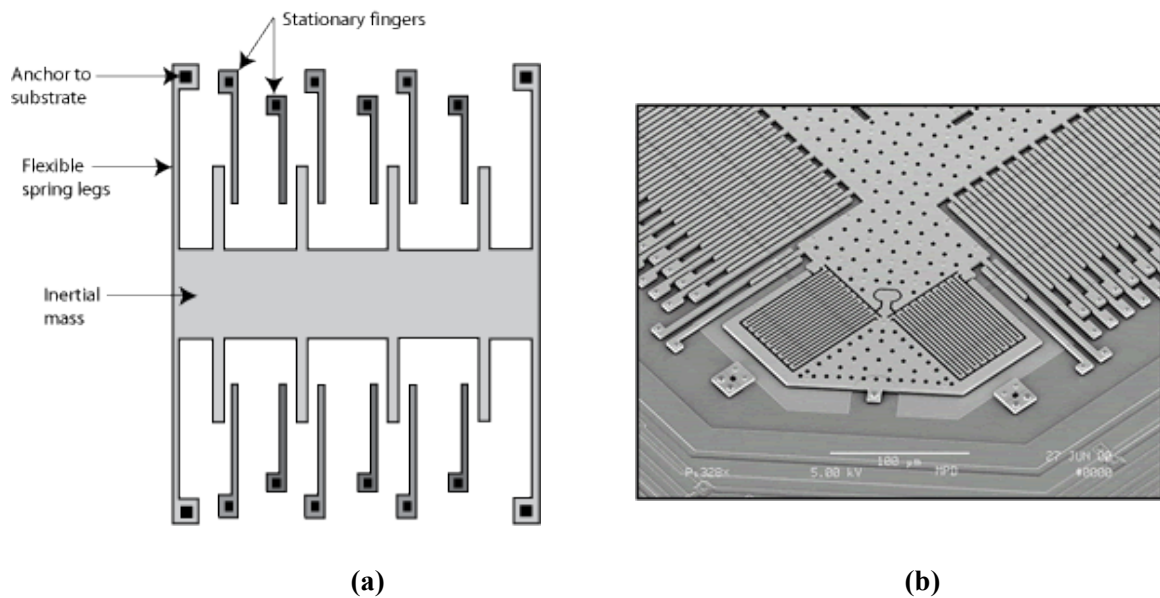


Figure 2-6: Capacitive Accelerometers [“Introduction to Microelectromechanical Systems (MEMS) | Compliant Mechanisms.”]

2.3.3.1.2 Piezoresistive Accelerometers

Fig. 2.7 shows a piezoresistive accelerometer, which consists of a proof mass suspended on a flexible beam. Other devices sometimes make use of a proof mass suspended on a flexible membrane. When acceleration is applied to this device, the deformation of the beam will result, which causes a change in piezoresistor resistance. The piezoresistor is protected, by limiting the movement of the proof mass by a housing to surround it [30].

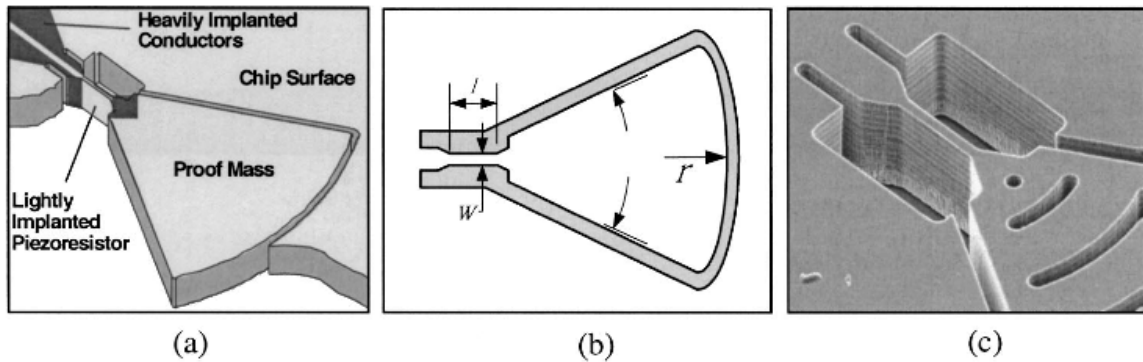


Figure 2-7: Piezoresistive Accelerometers [30]

2.3.3.2 Gyroscopes

Gyroscopes are used to sense and measure the orientation or the angular velocity of a device. For INS applications, this happens without using an external reference like magnetic north or gravity, and hence only measures their own rotation. They make use of the Coriolis effect, which happens when a mass (m) is moving with a velocity (v) and the frame of reference is rotating with an angular velocity (ω). When that happens the mass experiences a Coriolis force (F_c), which is determined by:

$$F_c = -2m(\omega \times v) \quad (2.2)$$

In MEMS gyroscopes, a mass is moving back and forth really quickly, and the Coriolis effect can be measured from that mass oscillation. Fig. 2.8 shows a simple geometry with a mass oscillating along a drive axis.

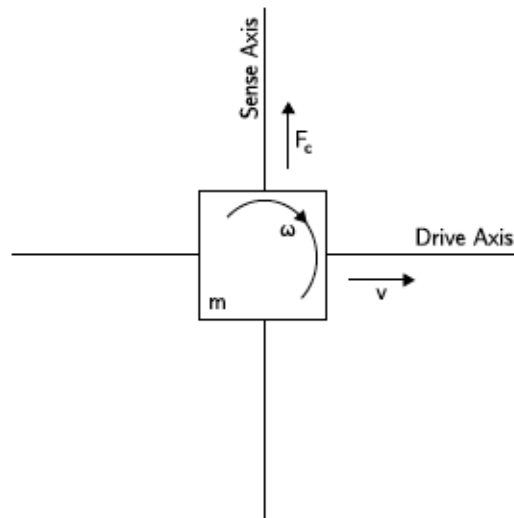


Figure 2-8: Principal of the Gyroscope [8]

3 Overview of Signal Processing

3.1 Inertial Navigation System (INS)

In the 19th century, the use of inertial navigation system was first established, as a simple gyro for determining the north. In World War 2, Germany developed the gimbaled INS, for use to navigate V2 rockets. The gimbaled INS was under research and development until late 1970's when the Strapdown Inertial Navigation System (SINS) was developed, where the sensors were strapped and mounted to the body of the vehicle. The advantages of the SINS over the gimbaled INS are many, such as: lower cost, higher accuracy, lower moving parts, less calibration, smaller size and more reliability.

Inertial Navigation System makes use of the inertial sensor for navigation and positioning estimation. It uses a combination of accelerometers, gyroscopes, and computer processing to be able to calculate acceleration, velocity (speed and direction), position and orientation. This is known as Dead Reckoning (DR), which is a process to determine the present position based on a previous known position, then progress that position based on a known speed over the time, and course.

To estimate the position between two or more points in space, a minimum of 6 degrees in freedom ($x, y, z, \theta_x, \theta_y, \theta_z$) in measurement is required. This usually consists of three accelerometers and three gyros - or three angular accelerometers. Fig. 3.1(a) shows the 6 degrees of freedom (DOF) in a space with respect to a reference point. For estimating the position between two or more points on a plane surface, at least two linear accelerometers and one gyro - angular accelerometer - are needed to form a device that works in 3 DOF. Fig. 3.2(b) shows the necessary 3 DOF in the y-z plane, in which two linear accelerations

in y and z, in addition to an angular velocity in respect to the x-axis is needed. In Flight dynamics, there are three parameters that are critical, which are called angles of rotation. In a three dimensional model, these three parameters are known as pitch, roll and yaw.

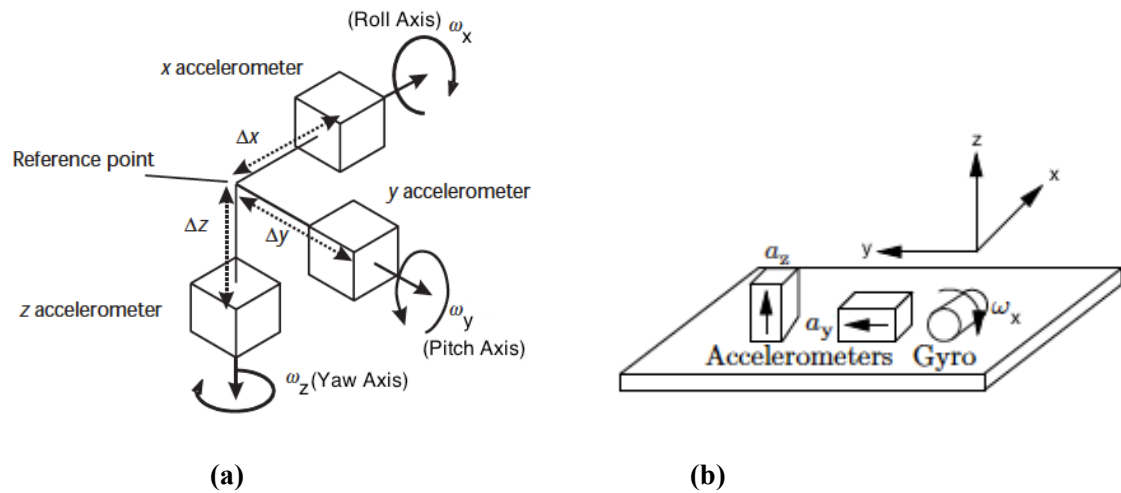


Figure 3-1: IMU (a) In Three Dimensions. (b) In Two Dimensions [31]

3.1.1 Angular Accelerometers and Gyroscopes

Fig. 3.2 shows an accelerometers being used as a tilt sensor. The black line in the graph shows the acquired signal, but it can be seen that there is a lot of noise, indicated by the zig-zag vertical variations in the black line. To create a smoother signal, signal processing with a low pass filter is needed to attenuate (to make more smooth) the signal, as shown by the red graph, which gives a smoother signal but it creates a delay.

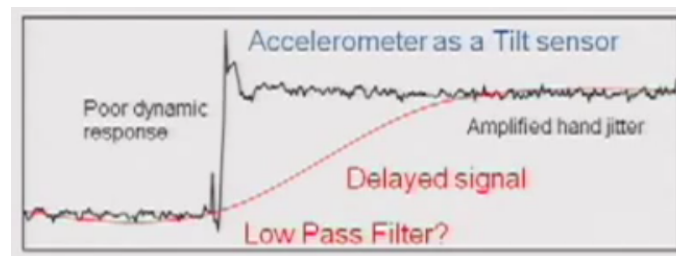


Figure 3-2: Tilt Sensor Output

Fig. 3.3 shows a block diagram of the gyro signals combined with the acceleration signals to ultimately create a device that can measure the orientation and the position of objects.

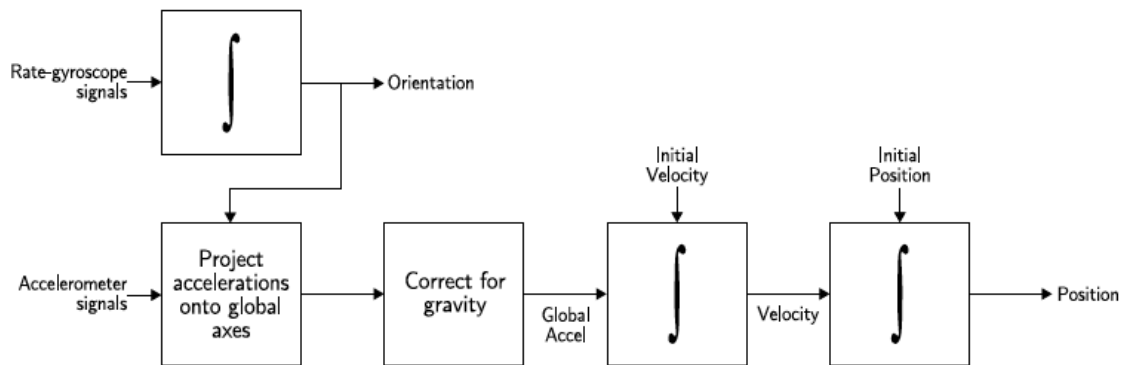


Figure 3-3: Diagram of the Linear and Angular Data Combination [8]

3.2 Positioning Measurements in Hand-Held Devices

Motion sensors (accelerometers and gyroscopes) have been extensively used in video games. It started with Nintendo when they implemented motion-sensing gaming in the Nintendo Wii controller. They have combined tri-axial accelerometers and multi-axis gyroscopes in the controller to increase the playability. Fig. 3.4 shows a Wii-mote console controller.



Figure 3-4: Wii-mote Consoles [“Wii Console Official Factsheet.”]

Mobile smart phones such as: the iPhone by Apple, and the Android by Google have incorporated motion sensors to detect the phone's orientation with respect to gravity, vertically or horizontally, and thereby changing the screen display accordingly (landscape and portrait). In addition, these sensors are used in interactive gaming, such as car driving simulation, where the user turns the device to simulate the steering wheel as in Fig. 3.5.



Figure 3-5: iPhone Video Game [C. Liu, “The MEMS Motion Sensor Perspectives.”]

3.3 Errors

In this document, “error” is defined as the difference between (deviation of) the measured value from the actual value. The “actual value” refers to the true (actual) value of the physical phenomena that is being measured. The “measured value” is the acquired raw data that is recorded by the sensor system, and is subject to various types of noise

leading to error. There are two main types of errors in measurement tools: Systematic errors and random errors.

Systematic error is the error that is consistent and repeatable for all readings and it is also called the “bias error” or “offset error”. In other words, if the same measurement tool was used to measure the same value a number of times, the same readings are obtained. Systematic error is caused by the calibration process and it is called calibration error, which is mainly caused by nonlinearity as shown in Fig. 3.6, and it represents the effect of the offset and nonlinearity on the measurements.

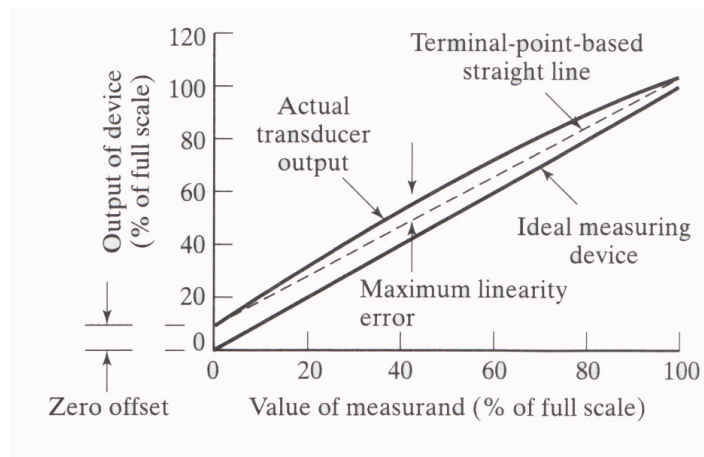


Figure 3-6: System Nonlinearity [32]

Another cause of systematic error is the loading error, which is the error due to the change of the measurand caused by the measurement tool. Generally, there are two types of measurement tools: intrusive devices, which produce a high loading error, and nonintrusive devices, in which their loading error can be ignored.

Random error is the error that differs with each measurement taken. To obtain a better estimate, the measurement is repeated a number of times with the same device, and the

average is calculated. The random error attributed to a single measurement can then be determined by subtracting this average from a single reading:

$$\text{Random error} = \text{reading} - \text{average of readings} \quad (3.1)$$

In addition, if the true value of the phenomena is known, the systematic error can be found by subtracting the true value from the average of the readings:

$$\text{Systematic error} = \text{average of readings} - \text{true value} \quad (3.2)$$

Fig. 3.7 illustrates the difference between the systematic error and the random error.

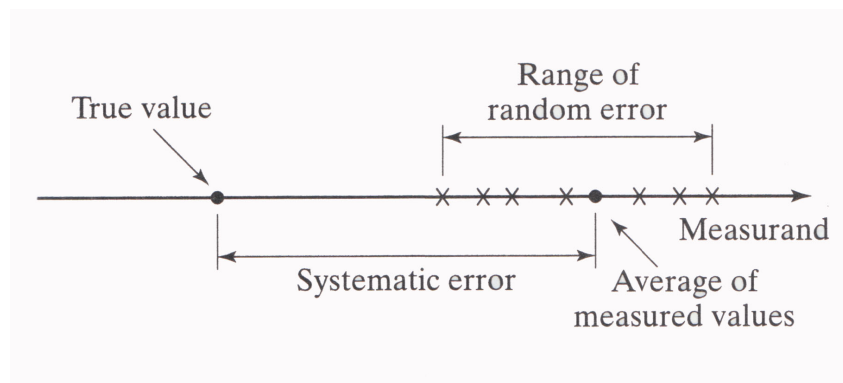


Figure 3-7: Systematic and Random Error [32]

The causes of random errors come from unpredictable disturbances, such as the change in the environment – pressure, temperature, and vibration. Other reasons could be caused by external magnetic and electrical fields, which affect the electrical components of the sensors or its electronics.

3.4 Noise

Signal noise is caused due to the random errors explained in the previous section. It is defined as the disturbance or interference of the electrical signal. Signal noise (or simply

“noise”) is undesirable because it distorts the original signals and changes their behaviour and nature by causing uncertainties. A good quotation is:

“It is easy to get an accelerometer to measure acceleration. The problem is to keep it from measuring everything else” [Walter Kistler].

3.4.1 Types of Noise

Noise can be divided into three main types:

- 1- White noise: This is random signal variation and it is inherent to all MEMS inertial sensors. It has a flat power spectral density, i.e. it has the same random amplitude of variation on all frequencies. Two conditions must be met for a random signal (v) to be considered as a white noise: Firstly, it the random signal should have a zero mean:

$$E(v) = 0 \quad (3.3)$$

Secondly, It must have a multiple identity matrix for the autocorrelation matrix:

$$E(vv^T) = \sigma^2 I \quad (3.4)$$

- 2- Drift, which can be described as a change in mean value of a random process. It is temporary (transitional) and unpredictable. For example, it could be caused by external environmental factors such as a room's lights or temperature.
- 3- Bias or offset error, which is constant and predictable. If the amount of bias can be determined, it can potentially be removed from the signal. For example, bias could be caused by poor calibration of the sensor, or constant interference from other sources.

3.4.2 Sources of Noise

Noise can be caused by internal sources or external sources. Internal sources are coming from the system itself, for instance, the thermal noise [19]. Noise produced from electrical circuits is generated from the electronic components. Fig. 3.8 shows two circuits, where circuit (a) has an ideal noise-free resistor, and circuit (b) has a typical resistor that has internal thermal noise with a noise voltage.

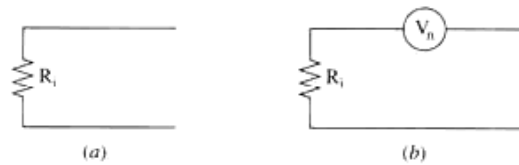


Figure 3-8: Ideal and Typical Circuits

For any temperature above 0°K , the electrons within electronic components exhibit random motion that generates pulses in the material, and these lead to internal electrical noise, denoted as:

$$V_n^2 = 4KTRB \text{ V}^2/\text{Hz} \quad (3.5)$$

Where:

V_n = The noise voltage

K = The Boltzmann's constant ($1.38 \times 10^{-23} \text{ J}/^\circ\text{K}$)

T = The temperature in Kelvin

R = The resistance in ohms

B = The bandwidth in hertz

External noise can be generated from many factors: Electromagnetic and radio signals interference and natural phenomenon such as lightening.

3.4.3 Noise Affects on Digital Signals

Digital signals are transmitted as high and low voltage constant states, typically 0 volts for low, and +5 volts for high. Another way to represent digital signals is with binary numbers, where 0 represents low, and 1 represents high. Therefore, in order for the noise to affect digital signals, the magnitude of the noise must be sufficient to “shift” the high or low voltage states significantly enough, to cause the processor to confuse the difference between high and low. Years ago, digital signal logic had large values and the speed of the systems was much slower than what is available now. Over years, the digital signal magnitude has been reduced and the system speed has been increased. However, the noise magnitude hasn’t changed at all. Fig. 3.9 shows two systems; the upper diagram has a logic signal to represent the high state as that between 20-to-30 volts, while the lower diagram has a logic signal that represents the high state as that between 3-to-5 volts. However, there is almost the same value of noise added to both systems. The noise will have less effect on the system that has a higher signal, than the one that has a lower signal.

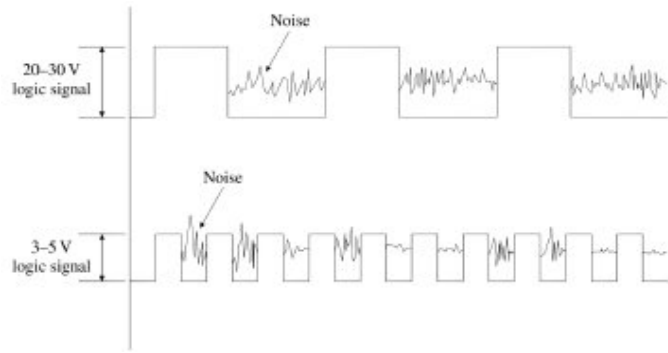


Figure 3-9: Relative Magnitudes of Signal and Noise [33]

The signal to noise ratio (SNR), is used to identify how much noise is in a signal. The SNR is defined as the signal power over the noise power. In other words, it determines the relationship between the signal to the noise, where the higher the SNR value is, the better the quality of the measured signal.

$$\text{SNR} = \frac{P_{\text{Signal}}}{P_{\text{Noise}}} = \left(\frac{A_{\text{Signal}}}{A_{\text{Noise}}} \right)^2 \quad (3.6)$$

Where:

P = Average Power

A = Root Mean Square

Because signals usually have a very wide dynamic range, they are often expressed in decibels (dB).

$$\begin{aligned} \text{SNR}_{\text{dB}} &= 10 \log_{10} \left(\frac{P_{\text{Signal}}}{P_{\text{Noise}}} \right) = \text{SNR}_{\text{dB}} = 10 \log_{10} \left(\frac{A_{\text{Signal}}}{A_{\text{Noise}}} \right)^2 \\ &= 20 \log_{10} \left(\frac{A_{\text{Signal}}}{A_{\text{Noise}}} \right) \end{aligned} \quad (3.7)$$

3.5 Filters

One of the main ways to remove unwanted signal noise, is to use a filter. A filter can be used to separate desired signals from undesired signals, block interfering signals, enhance speech and video, and alter signals in other ways. Almost all communication systems use filters. Often, a filter passes one band of frequencies while rejecting another band. It can be either passive or active:

- Passive filters are built with resistors, capacitors and inductors. They are generally used above 1 MHz, have no power gain, and are relatively difficult to tune.
- Active filters are built with resistors capacitors, and op amps. They are useful below 1 MHz, have power gain, and are relatively easy to tune [34].

3.5.1 Types of Filters

- Low pass filter: The low pass filter passes all frequencies from zero to the cutoff frequency and blocks all frequencies above the cutoff frequency – Fig. 3.13(a). The cutoff frequency is the boundary of the frequency responses where the signals attenuate or the frequency where is gain is reduced by 3 dB [32]. Attenuation refers to a damping/reduction of amplitude of signal with a constant input voltage. Fig. 3.10 shows the frequency response of a first-order system filter. The frequency response of a filter is generally represented using bode plots.

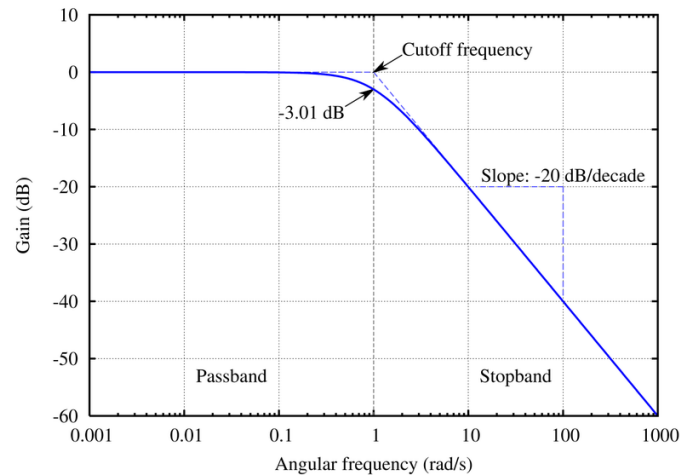


Figure 3-10: Filter Response [“Cutoff frequency - Wikipedia, the free encyclopedia.”]

The simplest form of an electronic Low-pass filter is shown in Fig. 3.11. This consists of a resistor in series with the signal path in conjunction with a capacitor in parallel with the signal path. The resistance times the capacitance ($R \times C$) is the time constant (T):

$$f = 1/2\pi\tau = 1/2\pi RC \quad (3.8)$$

Where:

f = The cutoff frequency in hertz

τ = The time constant in seconds

R = The resistance in ohms

C = The capacitance in farads

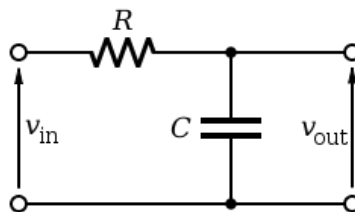


Figure 3-11: Low Pass Filter

- High Pass Filter: A high pass filter blocks all frequencies from zero up to the cutoff frequency and passes all frequencies above the cutoff frequency – Fig. 3.13 (b). The simplest form of an electronic high-pass filter is shown in Fig. 3.12. Here the capacitor is in series with the signal path and the resistor is in parallel with the signal path. The cutoff frequency can be calculated using the same equation for the low pass filter.

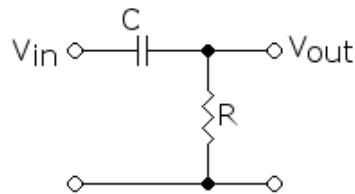


Figure 3-12: High Pass Filter

- Band pass filter: A band pass filter passes all the frequencies specified for the band, i.e. between the lower and upper cutoff frequencies. It blocks all frequencies below the lower cutoff, and above the upper cutoff – Fig. 3.13 (c).
- Band stop filter: A band stop filter passes all the frequencies from zero up to the lower cutoff frequency, and it passes all frequencies above the upper cutoff frequency. It blocks all the frequencies between the lower and upper cutoff frequencies, –Fig. 3.13 (d).

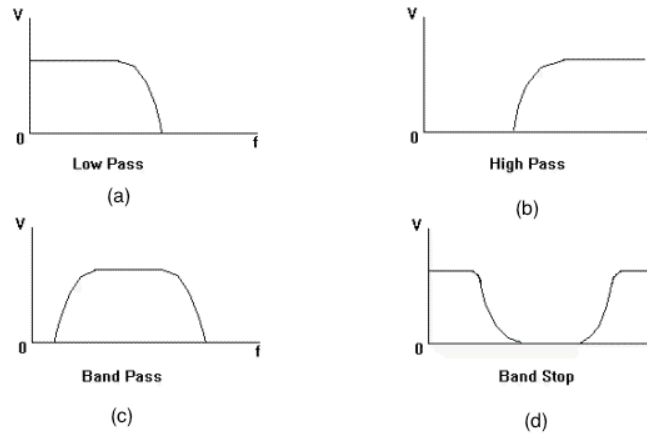


Figure 3-13: Different Types of Filters

3.5.2 Kalman Filter

The Kalman filter is a set of mathematical algorithms that uses prediction-correction type estimators to estimate the process variables to optimally minimize the error covariance. The Kalman filter makes use of many types of information, such as: (1) the system/plant dynamics, (2) the statistical information of the system noise and measurement noise, (3) system uncertainties and (4) initial conditions of system/plant, to provide an approximation of the system state.

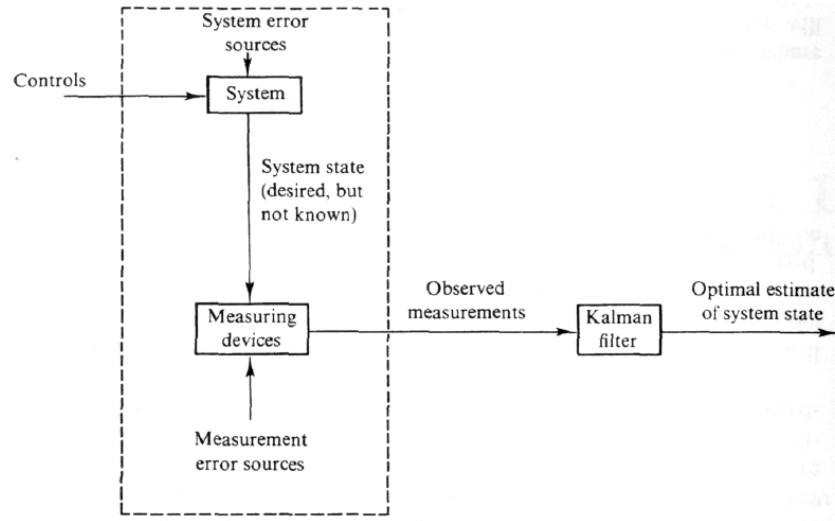


Figure 3-14: Typical Kalman Filter Application [35]

As shown in Fig. 3.14, some parameters cannot be read directly from the system and the observed measurement is corrupted by other disturbances (noise, bias, device inaccuracy etc.). The Kalman filter can be applied to linear systems, which are easier to be manipulated and many physical processes can be approximated as a linear system [36]. If a linear system cannot be used, and Extended Kalman filter can be used for non-linear systems. Consider the following system, which is described by the following state space equations:

The state equations:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (3.9)$$

The output equation:

$$Z_k = Hx_k + v_k \quad (3.10)$$

Where:

k : The time index

x_k : The system state equation

A: The system transition matrix

x_{k-1} : The system state in the previous time step

B: The input transition matrix

u_k : The known input of the system

w_{k-1} : The process noise

Z_k : The output measurement

H: The output transition matrix

v_k : The measurement noise

In order to apply the Kalman filter equation, two assumptions are required. Firstly, the average value of the process noise and the average value of the measurement noise must be zero. Secondly, at any time (k), the value of the process noise w, and the measurement noise v must be unrelated. Then the value of the process noise covariance is specified as:

$$Q = E(w_k w_k^T) \quad (3.11)$$

And the measurement noise covariance is:

$$R = E(v_k v_k^T) \quad (3.12)$$

Where:

Q: The process noise covariance

R: The measurement noise covariance

T: The transpose

E(): The expected value

Now, the Kalman filter equations can be applied, in which it is classified into two categories: time update equations and measurement update equations. The time update

equations are responsible for providing a prediction for the next time step and they are represented by the following equations:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \quad (3.13)$$

$$P_k^- = AP_{k-1}A^T + Q \quad (3.14)$$

Where:

P: The error covariance

The subscript $-$ (super minus) is called a priori, which indicates knowledge prior to step k.

Then there are the measurement update equations, which are responsible for the update information of the system state and the error covariance estimate, where they project a posteriori estimate and they are represented by the following equations:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad (3.15)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(Z_k - H\hat{x}_k^-) \quad (3.16)$$

$$P_k = (I - K_k H)P_k^- \quad (3.17)$$

Where:

K_k : The Kalman gain

I: The identity matrix

Finally, this operation is repeated for each measurement as shown in Fig. 3.15:

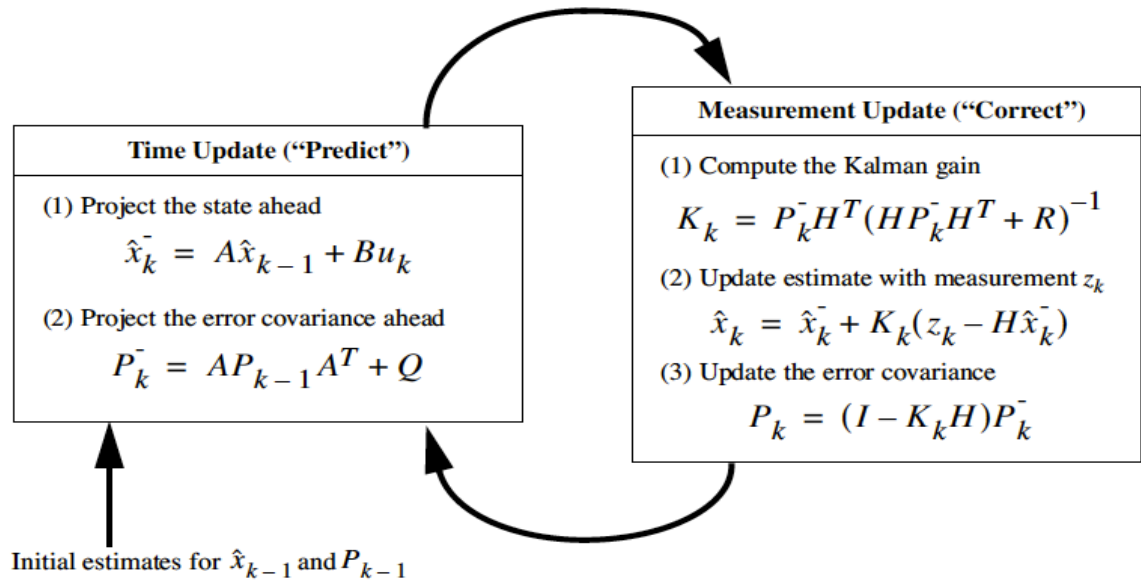


Figure 3-15: Kalman Filter Equations [37]

4 Relative Positioning Estimation System (RPES) Device

4.1 Device Overview

In this section, an overview of the relative positioning estimation system (RPES) is provided. Also, the RPES technical details including prototype device specifications, part names, and construction details are described. In addition, the software and applications utilized for the RPES are presented.

In this work, a prototype hand-held device has been constructed in order to implement the proposed research. The prototype device was designed to satisfy certain requirements such as: Portability, Low weight, and Low Cost components. The device uses MEMS accelerometers and gyroscopes, to provide information about linear acceleration and angular velocity. This information was acquired by the analog-to-digital ports of an on-board microcontroller, and then transmitted wirelessly through a Bluetooth module to a desktop PC equipped with a compatible Bluetooth dongle. The prototype device was powered by a 9-volt DC battery, which has a relatively low noise power supply. The micro-controller used to convert the MEMS sensors analogue signals to digital signals, was an AT90USBKey. It also provided the ability to simultaneously capture a number of sensors signals for later experiments, where it synchronously conducted the conversions.

Four main software packages were used for programming the components and processing the data. The first software package was the AVR studio 4, which was used to program the micro-controller. The program specified the micro-controller's timer to sample the signals accurately, in which the sampling frequency should be very high to avoid the aliasing phenomenon. The sampled signals was digitized into 2^{10} bits, which is

the A/D converter resolution. The A/D converter was capable of converting a number of input signals simultaneously. The AVR studio 4 also specified the micro-controller to run the Bluetooth module to work synchronously with the signals sampled, in which each bit of sampled data was sent by the Bluetooth module to a desktop PC to be stored. Based upon this, the A/D sampling frequency should be equal to or greater than the Bluetooth sampling frequency; otherwise, some data would be lost. Also, an interrupt feature was programmed to stop the device at the end of the experiment.

The second software package was Flip 3.3.4, which used for translating (compiling) the high-level computer language program written with the first software package, into a low-level computer language. Then this low-level processor-specific program is downloaded into the micro-controller.

The third software package was an open source serial terminal, which is HTerm 0.8.1 beta. It was used for communicating with the prototype device Bluetooth module to receive the data live and store it for later analysis.

MATLAB was the fourth software package used for the data analysis. It was used for mathematical computation and manipulation of the collected data. This software package was used for different operations such as: (i) converting the data from digital information, into voltages, and then into acceleration data, (ii) applying the Kalman filter, (iii) using conventional averaging techniques, and (iv) fusing the input signals through the Kalman filter using different design algorithms. It was also used for (v) manipulating the data by applying specific boundary conditions, (vi) removing unwanted sections of the signals, (vii) integrating the data twice to get the displacement and finally, (viii) plotting the acceleration, velocity and displacement graphs.

4.1.1 Prototype Device Component Specifications

The prototype device component specifications and descriptions are listed in Appendix-A. Fig. A.1 in Appendix-A shows the AT90USBKey layout, followed by the component descriptions. A 9-volt battery was used to provide power to the kit. The wireless Bluetooth system was implemented with a module on the prototype device, which communicated with a USB adaptor connected to a desktop computer. Fig. A.2 shows the Bluetooth module and the USB dongle.

The inertial sensors used were dual axis accelerometer evaluation boards from Analog Devices, using the model ADXL203 accelerometers. These accelerometers can detect accelerations in a range of $\pm 1.7 g$. The accelerometers output voltages range from 0 volts to 3.3 volts. The output voltages correspond to the detected acceleration, i.e. 3.3 volts correspond to $+1.7 g$ and 0 volts correspond to $-1.7g$.

4.1.2 Prototype Device Layout with Single Accelerometer

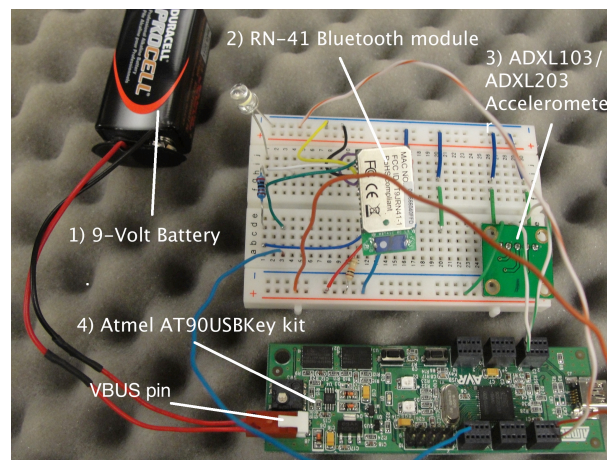


Figure 4-1: The RPES Prototype Device Layout with Single Accelerometer Module

Fig. 4.1 shows the RPES Prototype Layout, including the (1) The power supply using the 9-Volt battery, 2) the RN-41 Bluetooth module, 3) the ADXL203 accelerometer and 4) The Atmel AT90USBKey.

The AT90USBKey operation is as follows: The external 9-Volt battery supplies power to the VBUS pin, then the power supply voltage passes through a diode rectifier, which rectifies the voltage to 5-Volts. Then, it goes through a voltage regulator, which reduces the voltage to 3.3 volts to power the whole circuit. The AT90USBKey is used to power the breadboard shown in Fig. 4.1 with same 5-Volt power supply. The ADXL203EB dual axis accelerometer evaluation board was placed on a breadboard, which contains the ADXL103/ADXL203 accelerometer. There are five leads coming out of the evaluation board; one for the power, one for the ground, one for the self-test, which is also connected to the ground, one for the x-axis acceleration and one for the y-axis acceleration. The accelerometer's x-axis, y-axis and the self-test leads are connected with 0.1 μ F capacitors for each, which implements a low-pass filter for anti-aliasing and noise reduction according to the following equation:

$$F_{-3\text{ dB}} = 1/(2\pi(32\text{ k}\Omega) \times 0.10 \times 10^{-6}) \approx 50\text{ Hz} \quad (4.1)$$

The x-axis and y-axis leads are connected to the first and second channels of the analog to digital converter of the Atmel AT90USBKey kit. The Bluetooth module has eight wires where each wire is coloured differently to ease the connection. The orange color is used to set the Baud rate; high-level signals set the Baud rate to 9600 bps and low-level signals set the Baud rate to 115K bps. This wire is connected to the 3.3-volt power, which means it is always high and the Baud rate is set to 9600 bps. The red wire provides the module with the 3.3-volt power input, which comes from the AT90USBKey power

supply. The black wire is connected to the ground. The purple wire is used to indicate the status of the module, whether it is transmitting or receiving. The green wire is the UART (Universal asynchronous receiver/transmitter), which is used to receive input and is connected to the third bit of Port D on the AT90USBKey, which is the transmission data bit. The white wire is the UART transmit output. The yellow wire is connected to the ground to enable the transmission. The blue wire is for the module's status and it is connected to an LED to indicate when if the module is ON or OFF. The purple and the white wires are not used for our work since we are not going to be receiving any data.

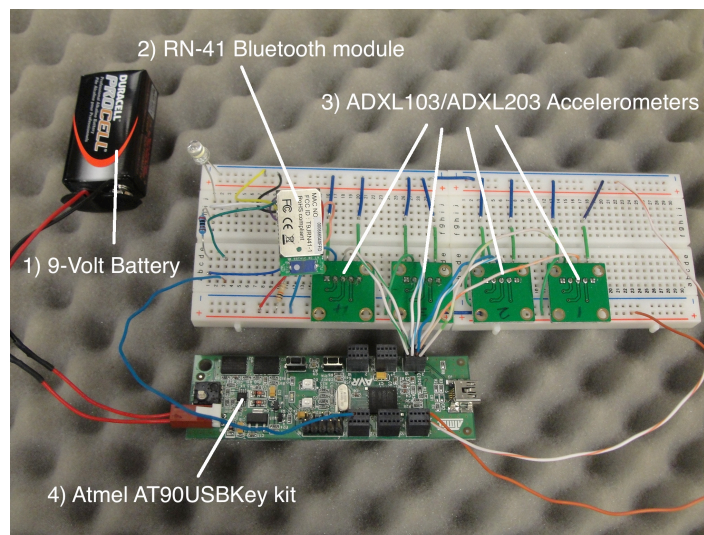


Figure 4-2: The RPES Prototype Device Layout with Four Accelerometer Modules

4.1.3 Prototype Device Layout with Four Redundant Accelerometers

The second device layout is exactly the same as the first one except that four identical accelerometers are placed on the breadboard, as shown in Fig. 4.2. Four accelerometers are used because of the limited channels of the A/D converter, less electrical complexity, and lower cost. Also, four were chosen since two units may not be sufficient to show any

possible change in data output, in comparison to the single sensor system. In future, more than four units could be tested by using a micro-controller with more than four A/D converter channels per axis. Each accelerometer's connections are the same as the one explained previously and they all are connected to the analog to digital converter through the 1st channel up to the 8th channel for the x-axis and the y-axis.

4.2 Experiment Set-Up

In order to conduct a single experiment, the following procedure was used:

- (1) The prototype device was switched on.
- (2) The reset button was pressed.
- (3) The terminal program was started.
- (4) The prototype device was moved to a certain distance.
- (5) The terminal program collected the data in real time.
- (6) The prototype device was switched off.
- (7) The data was saved in an editor file.
- (8) MATLAB was used to read the data from the editor, manipulated it then the estimated distance was obtained.

During the experiment, the accelerometer signal is passed through a 50 Hz low-pass filter. The accelerometer was placed perfectly parallel to the ground to avoid the effects of gravity. The micro-controller was programed to receive the filtered analog voltage signals through the first and second channels of the A/D converter. Then the signals were quantized according to the A/D converter resolution, which is 10-bit. According to the

accelerometer's data sheet, the voltage output range is from 0-volt to 3.3-volts. The 3.3-volt range is quantized into 2^{10} bit, which equals 1024:

$$Q = V_{FSR}/N = 3.3 \text{ volts}/1024 \approx 3.22 \text{ mV} \quad (4.2)$$

The accelerometer signal was sampled every 250 ms, hence the sample frequency was 4 Hz. Note that the sample frequency must allow for the maximum ADC conversion time, which is equal to 260 μ s, hence there was no problem sampling at 4 Hz. Each digitized sample contained up to $2^{10} = 1024$ bits of data and they were stored into two registers in the micro-controller, the ADCL and ADCH registers. The first two bytes ($2^2 = 4$) were stored into the register ADCL and the next eight bytes were stored into the register ADCH. The micro-controller was also programmed to transmit each register, right after it was sampled, through the USART transmit bit (Bit 3 of Port D) of the platform kit. The transmitted samples were received by the Bluetooth module, which received the signals live and transmitted them continuously to a compatible USB dongle with a transmission speed of 2 Mbits/s, and at a range of up to 10 meters. The desktop computer was running the HTerm terminal software, and was used to record all the data received by the USB dongle, on the computer screen in real time. When the experiment was done, the data captured with the terminal program was saved into a text file for further processing with MATLAB.

MATLAB program was used to manipulate and process the data. First, the collected data was in the form of digital values, where each sample value was a combination of two readings, the readings of the ADCL register and the ADCH register. So, to combine these data together, the following formula was applied:

$$\text{The ADC value} = (\text{ADCL} \cdot 4) + (\text{ADCH}/64) \quad (4.3)$$

Next, the ADC value was converted into voltages with respect to the reference voltage applied to the accelerometer (i.e. the microcontroller's power supply – 3.3 Volts) and then divided by 1023, which was the A/D converter 10-bit resolution, using the following equation:

$$V_{in} = ADC \cdot V_{ref}/1023 \quad (4.4)$$

Where:

V_{in} : The resulting voltage values

ADC: The combined digital value

V_{ref} : The selected voltage reference (3.3-volt) applied to the accelerometers.

Since the accelerometer's axis was approximately parallel to the ground, there little bias due to gravity. Therefore, the output voltage produced from the accelerometer during a 'static calibration test' was approximately 1.56 volts, which corresponded to zero acceleration. Hence, to convert the voltage data into acceleration data, 1.56-volt was subtracted from each corresponding voltage value, and then multiplied by the g (gravity) constant (9.80665 m/s²). Finally, this value was then divided by the accelerometer's sensitivity (0.626 volt/g) using the following equation:

$$a = g \cdot (V_{in} - V_{ideal})/S \quad (4.5)$$

Where:

a: The corresponding acceleration value

g: Earth gravity (9.80665 m/s²)

V_{ideal} : The output voltage of the accelerometer when it is stationary and parallel to the ground (1.56 v)

S: The accelerometer sensitivity (0.626 volts/gravity)

4.2.1 Kalman Filter implementation in MATLAB

For some experiments (as described in Chapter 5), a Kalman filter was applied. In those particular experiments, the measurement noise covariance was equal to the reading's variance and the process noise covariance was assumed to be a very small number (1×10^{-5} volt²). Initial conditions were set, where the initial system state (x_i) was equal to zero, the initial system state estimate ($\hat{x}_i$) was equal to zero, and the initial error covariance (P_i) was equal to 1. Finally, the transition metrics (A, B, and H) were set to be 1 for each. After the required parameters were set, the filter was ready to be applied and it was done by implementing a “for loop” from 1 to 3601, because the filter was applied every half a second for a 30 minute duration:

Number of times the filter was applied

$$\begin{aligned}
 &= \frac{1}{2} \text{ hour} \times 60 \text{ minute} \times 60 \text{ seconds} \times 2 \text{ times per second} \\
 &= 3600
 \end{aligned} \tag{4.6}$$

The Kalman filter equations were then applied inside the “for loop” as follow:

$$u = 0 \tag{4.7}$$

$$P = P + Q_w \tag{4.8}$$

$$K = P (P + R_v)^{-1} \tag{4.9}$$

$$\hat{x} = \hat{x} + K(a - \hat{x}) \tag{4.10}$$

$$P = (1 - K)P \tag{4.11}$$

Where:

u: The input value

Q_w : The process noise covariance

K: Kalman Gain

R_v : The measurement noise covariance

$\hat{x}$: The updated state of the system

a : The corresponding acceleration (the output of the accelerometer)

Then the output acceleration and the updated state of the system (the filter output) values are stored into arrays.

MATLAB was also used to graph various forms of the data, which included: (i) raw acceleration data derived from Equations (4.3), (4.4) & (4.5), (ii) The Kalman filter applied to raw acceleration data, (iii) the integrated Kalman filter data to get the velocity plot vs. time, and (iv) the integrated velocity data to get the displacement plot vs. time.

For the integration process, different types of numerical integration were considered for the data manipulation, such as:

- ✓ Left and Right Riemann Sums
- ✓ Trapezoid Sums
- ✓ Simpson's Rule

For this work, the cumulative trapezoidal integration method was used, which computes an approximation of the cumulative integer via the trapezoidal method with time.

5 Experimental Set-Up

There were two major types of experiments done, each using three main approaches. The first major type was the use of a “single sensor”, and the second type using “multiple redundant sensors”. For each of these, three approaches were investigated including: (i) application of boundary conditions, (ii) use of standard averaging techniques, and (iii) use of Kalman Filtering techniques. These experiments were done with the hardware setup illustrated and described in Section 4.1.2. All experiments were done as “static” tests, meaning that the true (actual) motion of the RPES device was equal to zero. The duration of each experiment was 30 minutes (1800 seconds), and there were a total of 48 experiments performed.

5.1 Single Accelerometer Experiment

The data collected from the accelerometer was converted in acceleration data, using the procedure described in Section 4.2. After conversion into acceleration, the data was plotted with time, as shown Fig. 5.1. In this figure, the true acceleration signal is shown as a green line, which was zero since the test was static. The measured acceleration data is shown as a red line. The measured acceleration contained three types of error: an offset, a drift and the Gaussian white noise. Then the true and measured accelerations were integrated and a plot of velocity with respect to time was obtained, as shown in Fig. 5.2:

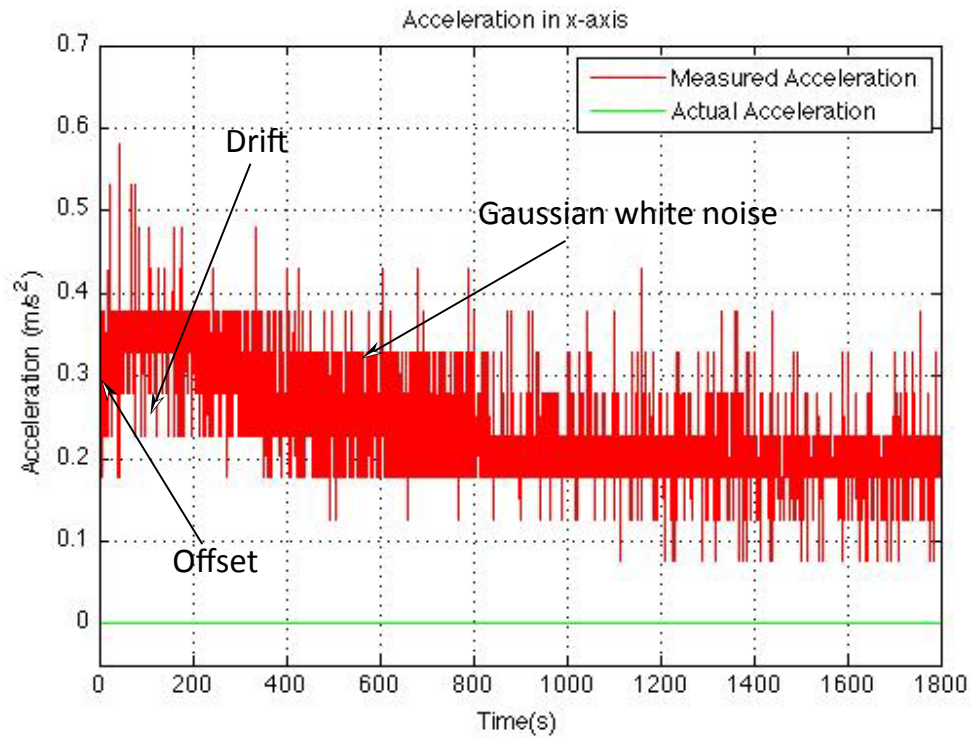


Figure 5-1: Plot of Measured and Actual Acceleration

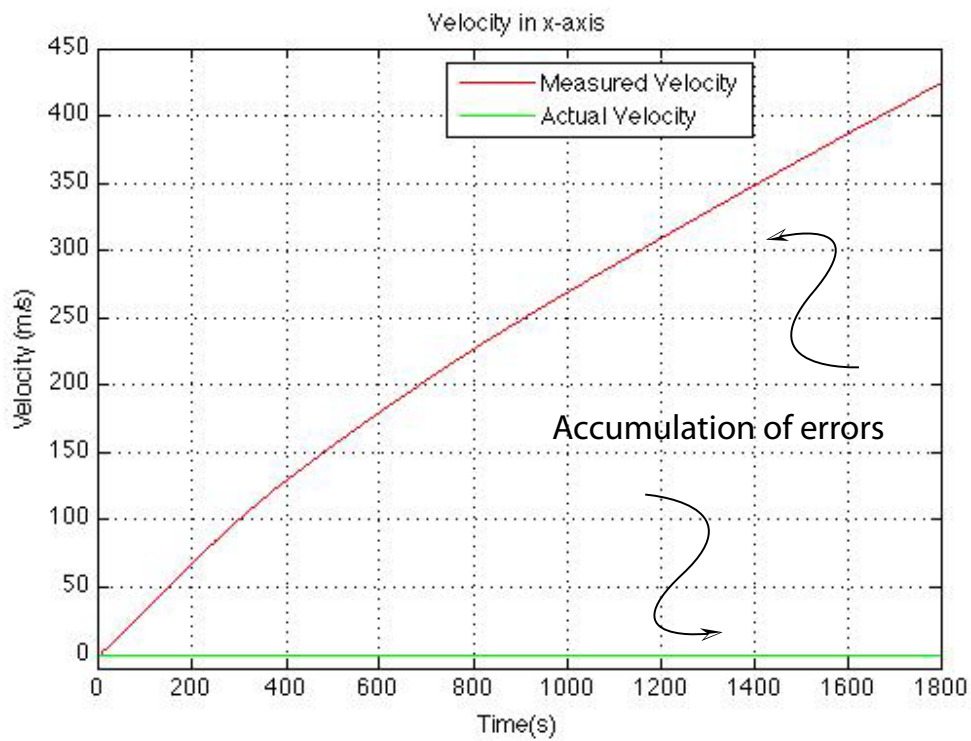


Figure 5-2: Plot of Measured and Actual Velocity

In this figure, it was observed that the three types of errors caused the measured velocity to diverge from the true velocity. Also, the divergence increases with respect to time, since the errors accumulated with time. Next, the measured and the true velocity curves were integrated one more time to obtain a plot of the displacements with respect to time, as shown in Fig. 5.3.

The same accumulation of errors can be observed in the displacement plot. As observed from this basic approach of using a double integral on the raw measured acceleration signal, the error is very high after 30 minutes. Therefore, more advanced approaches were investigated, as described in the following sections.

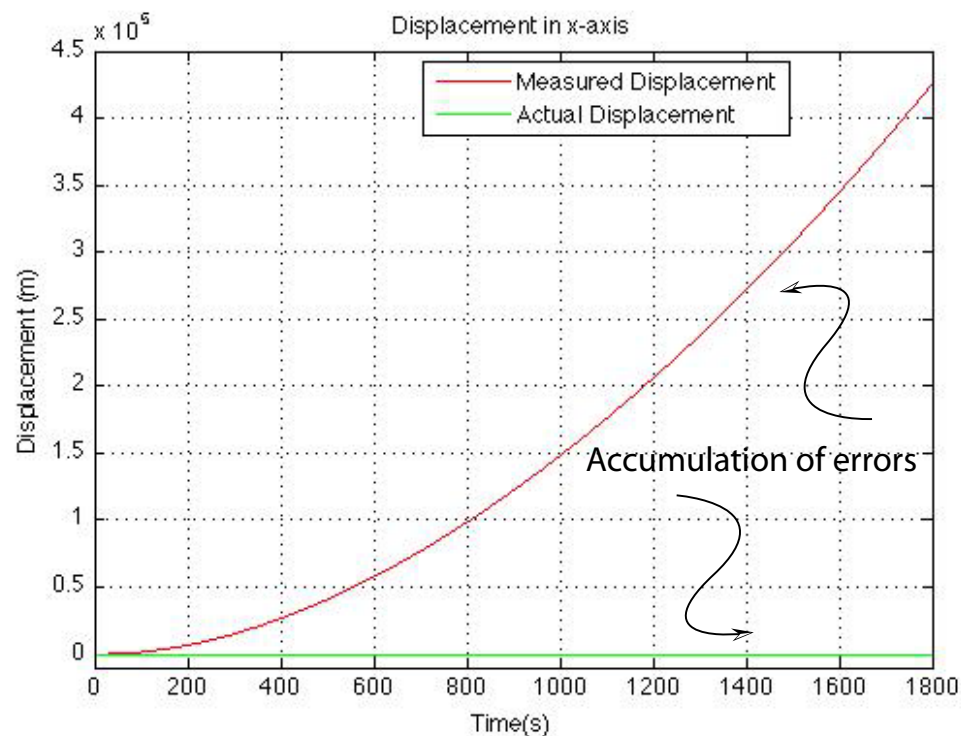


Figure 5-3: Plot of Measured and Actual Displacement

5.1.1 Applying The Kalman Filter

In order to improve the estimates of velocity and displacement, the Kalman filter was investigated for application to the measured acceleration data.

Fig. 5.4 shows the same measured acceleration data (in red) as Fig. 5.1. However, here, a Kalman filter was applied to that data, resulting in the blue line shown, where a lot of white noise is removed by the filter. There is an initial time period needed for the Kalman filter to fully predict the behaviour of the measured acceleration. In other words, it is the time needed for the filter error covariance to reduce to the minimum until it settles down. Therefore, the first few seconds of the Kalman filter output, and the measured acceleration were removed. This was only possible since the true/actual acceleration was known to be zero during this initial time. Note however, that the data offset, and the data drift remain. To deal with the offset and the drift, a couple of techniques were used.

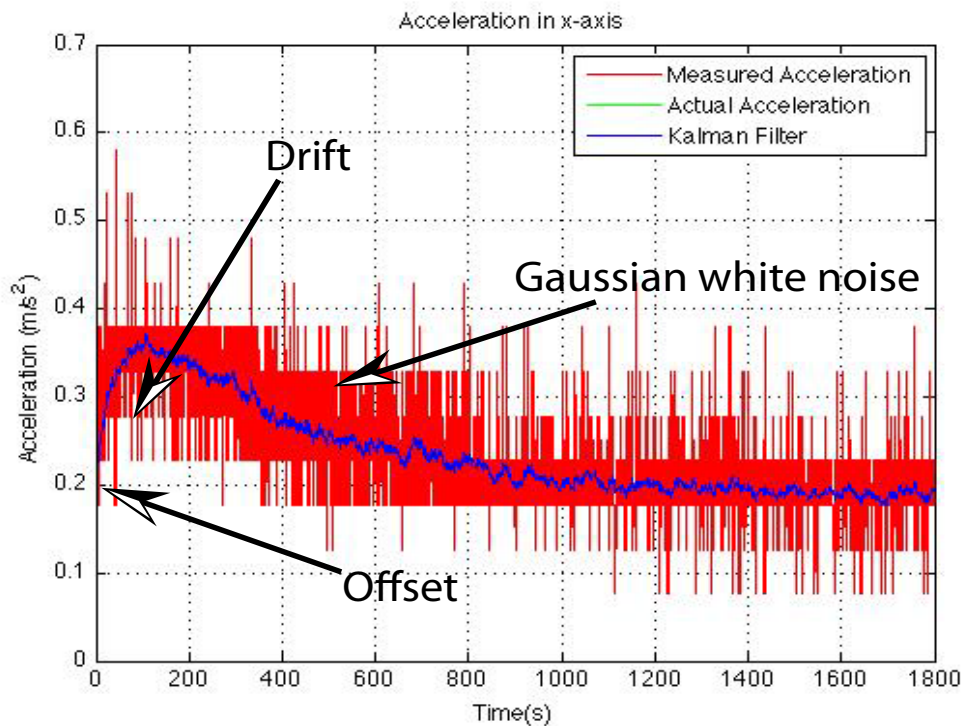


Figure 5-4: Plot of Measured and Actual Acceleration and the Kalman Filter Output

5.1.2 Different Techniques to Reduce Offset and Drift Error

In order to reduce the offset and drift error, the boundary conditions specific to the application of the RPES device could be applied. These boundary conditions and their rationale are described in Section 1.1, They are as follows: The initial and final acceleration are assumed as zero, the initial and final velocity are assumed as zero, and the initial position is assumed as zero. These boundary conditions were implemented by “mathematically shifting” the acceleration and velocity plots using a formula, referred to as a “rotation”. The formula for the acceleration data rotation is presented as:

$$R_i: A(t) - \left(A(1)(t - T) / (T - 1) \right) \quad (5.1)$$

$$R_f: A(t) - \left(A(T)(t - 1) / (T - 1) \right) \quad (5.2)$$

The formula for the velocity data rotation is presented as:

$$VR_i: V(t) - \left(V(1)(t - T) / (T - 1) \right) \quad (5.3)$$

$$VR_f: V(t) - \left(V(T)(t - 1) / (T - 1) \right) \quad (5.4)$$

Where:

t : The time step.

T : The entire time the experiment was conducted.

$A()$: The acceleration data.

$V()$: The velocity data.

To ease the explanations, specific notations have been assigned to each operation as presented in Table 5.1:

Letter Code	Description
F	Kalman filter applied
R_i	Rotating acceleration data around the final point to zero the first point
R_f	Rotation acceleration data around the first point to zero the final point
VR_i^*	<i>*Note: VR_i is not used because it is automatically applied as a consequence of Applying R_i</i>
VR_f	Rotation velocity data around the first point to zero the final point
DR_i^{**}	<i>**Note: DR_i is not used because it is automatically applied as a consequence of Applying R_i</i>
C	Removing the first few seconds (the time needed for the filter to fully understand the behaviour of the measured acceleration)
I	Integration
$11M$	Moving Average Filter with 11-Sample Size
$21M$	Moving Average Filter with 21-Sample Size
$41M$	Moving Average Filter with 41-Sample Size
AF	Take the Average of the multiple Raw Data then apply the Kalman Filter on the Single Reading
FA	Apply the Kalman Filter on the multiple Raw data the take the Average
VA	Take the Average of the Velocity data
AD	Take the Average of the Displacement Data

Table 5-1: Table of Processing Techniques

Error is defined as the difference between the true signal and the measured signal. Therefore, the estimate error in the Kalman filter refers to the difference between the true signal and the estimated signals, and it is represented by the following equation:

$$e_t = x_t - \hat{x}_t \quad (5.5)$$

Where:

e : The estimated error.

t : The time step.

x : The true signal.

$\hat{x}$: The estimated signal.

Covariance is a measure to determine how much two random variables move together. Consequently, the estimated error covariance specifies the strength of the relationship between the estimated error random variables, which governed by the following equation:

$$P_t = E(e_t e_t^T) \quad (5.6)$$

Where:

P : The estimated error covariance.

e : The estimated error.

$E()$: The expected value.

T : The transpose

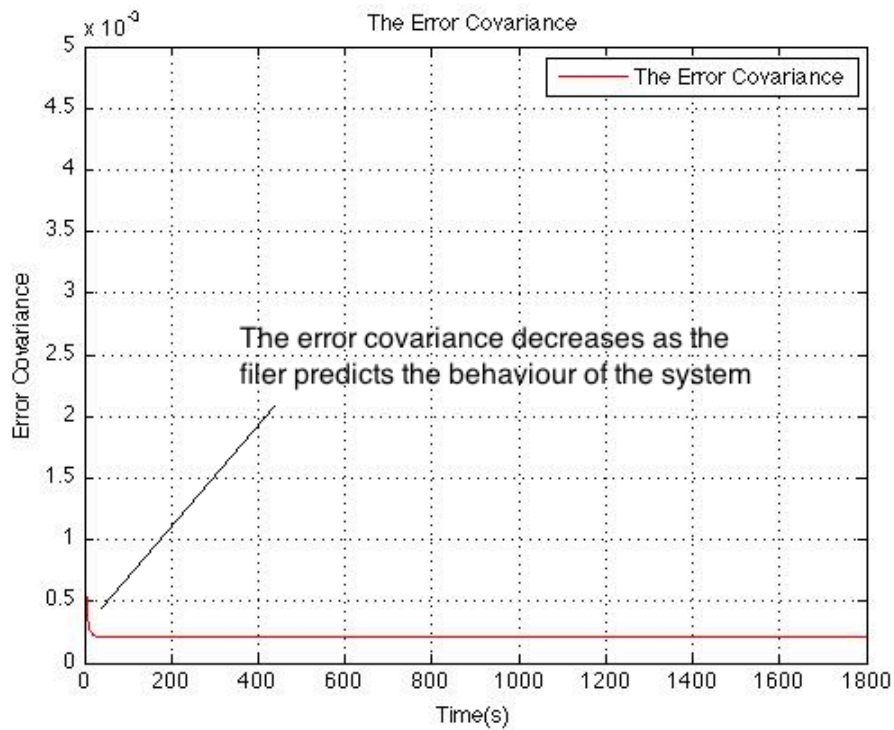


Figure 5-5: The Error Covariance

This graph shows the Kalman filter's error covariance, which decreases from an initial value, until it settles down to a constant value.

In Fig. 5.6, a close-up view of the first 100 seconds of data is shown, with a close-up of the error covariance. The figure shows that the time needed for the error covariance to become minimum, is about 25 seconds.

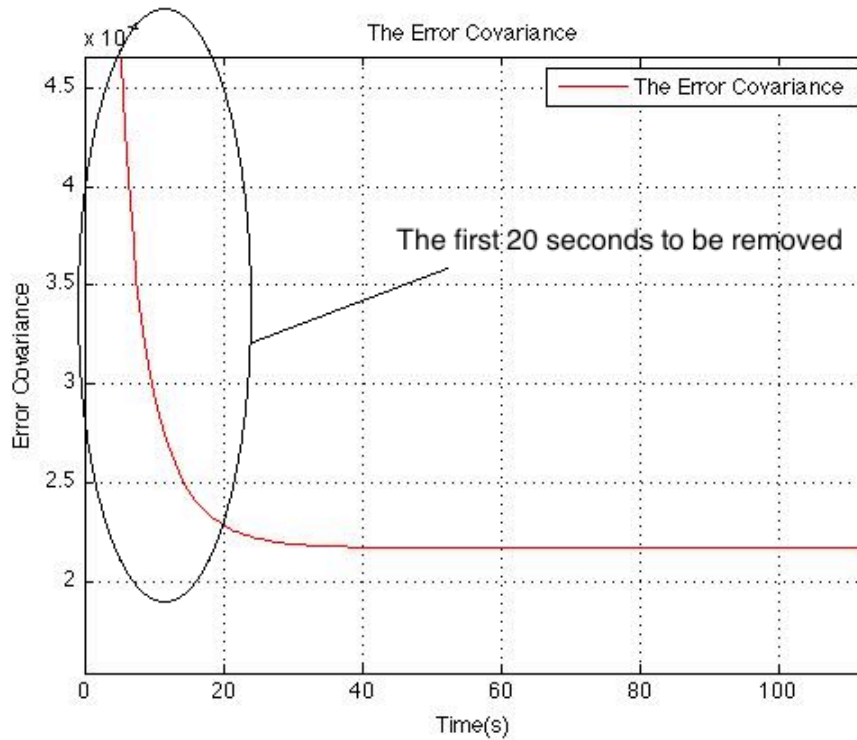


Figure 5-6: Close-up of the Error Covariance

Fig. 5.7 shows a close-up of the white noise error in the first 45 seconds of the measured acceleration and the Kalman filter, where it was observed that there were some large spikes in the filter output. Therefore, the first 20 seconds of data were removed.

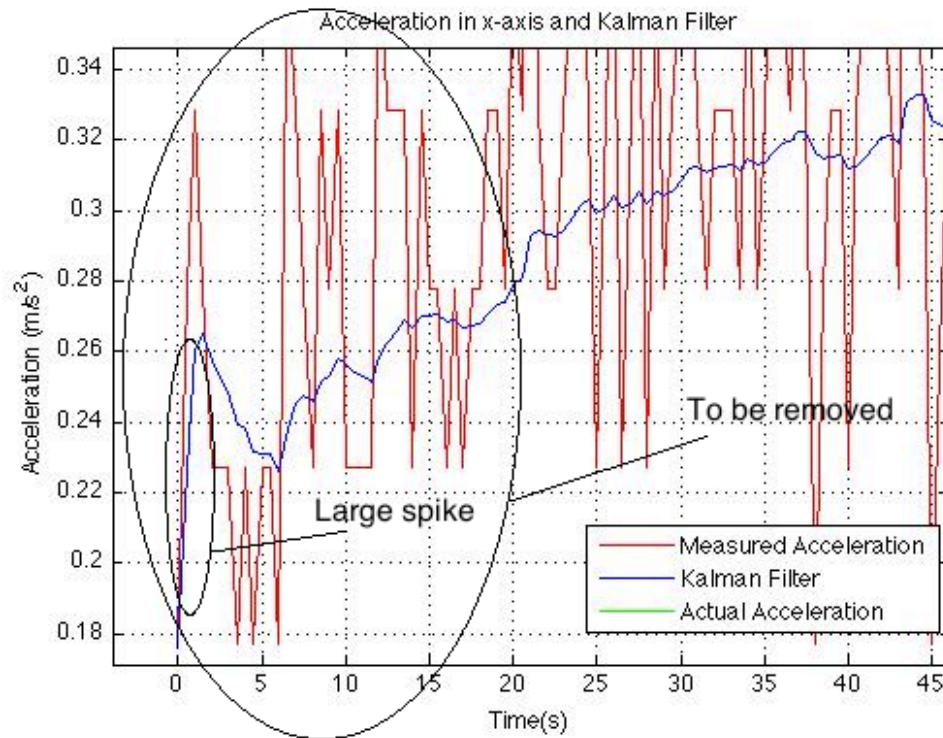


Figure 5-7: Close-up of the White Noise Error in the First 45 Seconds

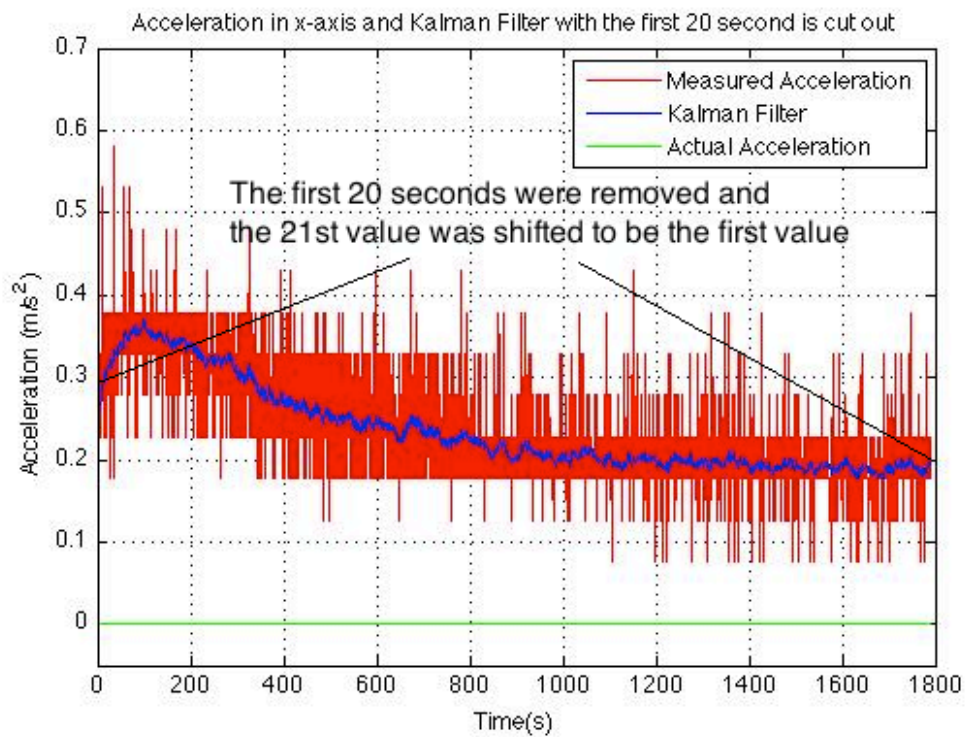


Figure 5-8: Plot of Measured and Actual Acceleration and the Kalman Filter Output after the First 20 Seconds of data were removed

Next, the data set was shifted to the left, as shown in Fig. 5.8. Fig. 5.9 shows the resulting improvement of the filter output, where it was considerably smoother with no large spikes observed, as a result of removing the first 20 seconds of data.

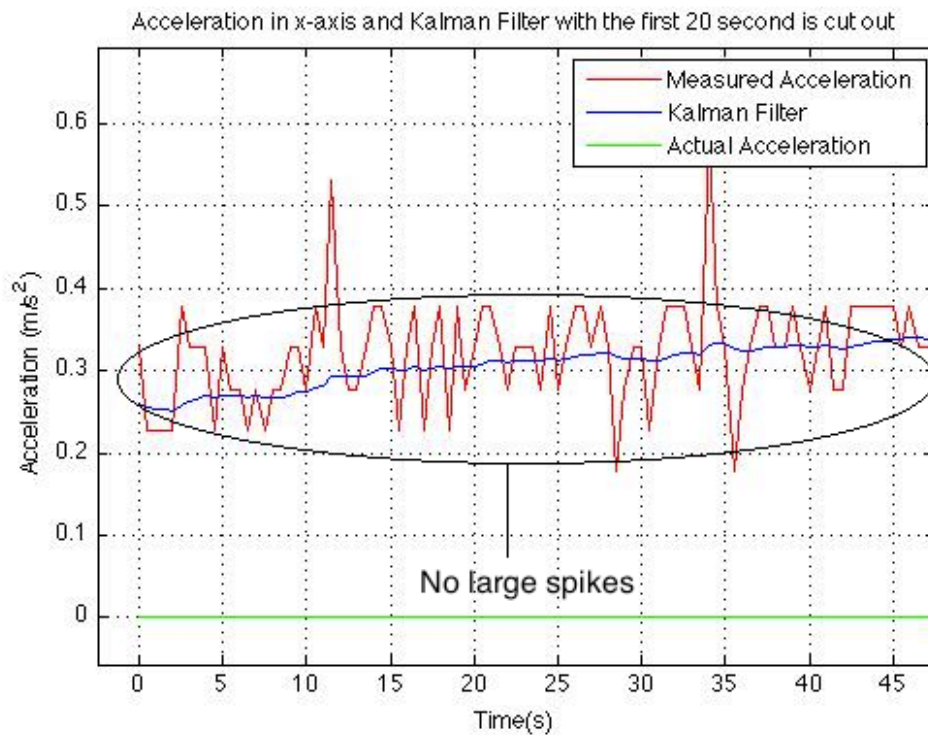


Figure 5-9: Close-up of the White Noise Error after the First 20 Seconds were removed

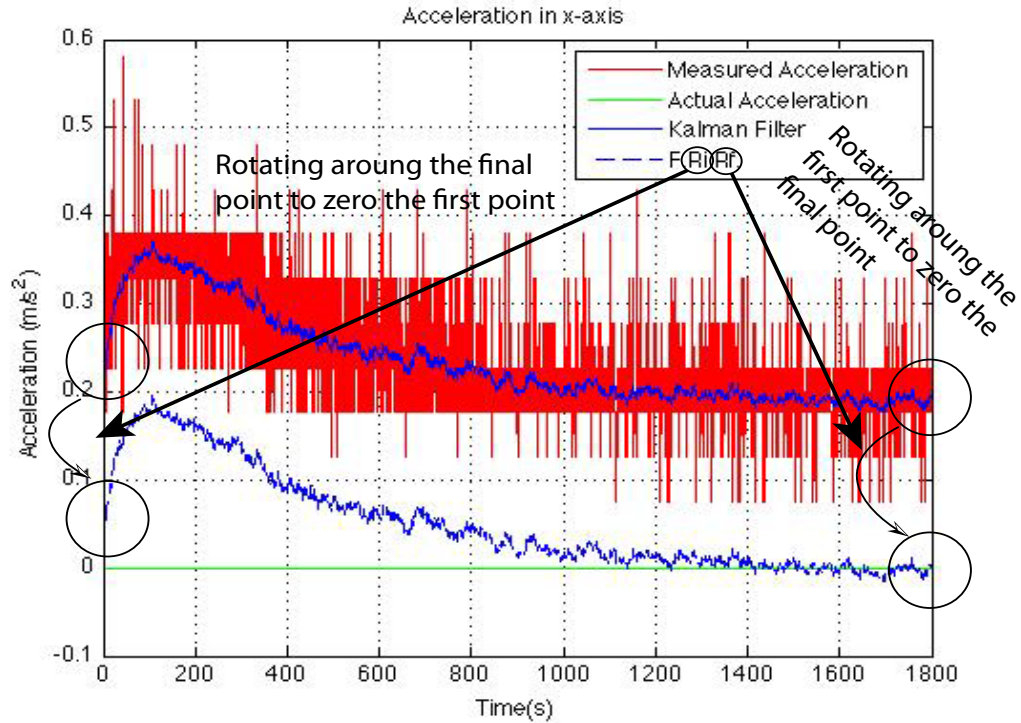


Figure 5-10: Plot of Measured and Actual Acceleration, the Kalman Filter and Two Boundary Conditions Applied

After the Kalman Filter procedure, F, and the data cutting procedure, C, the first boundary conditions are applied to reduce the offset error as shown in Fig. 5.10. The implementation is done by applying Equation (5.1), which mathematically rotates the Kalman filter plot around the final point to force the initial point to be zero. Then using Equation (5.2) to rotate the resulting graph around the initial point to force the final point to be zero. Applying the same equations on the actual acceleration results in the graph shown in Fig. 5.11. Note the caption shown in the top-right of the plot in the box. The red line is denoted as “C.Ri.Rf.” This means the raw acceleration data had a 20 second cut, C, and next the two acceleration rotation formulas were applied, Ri and Rf. Similarly, the blue line is denoted as “F.C.Ri.Rf” which is almost the same, except that

the Kalman filter (F) is applied first before the other processing. As can be seen both the red line and the blue line clearly start at value zero, and end at value zero.

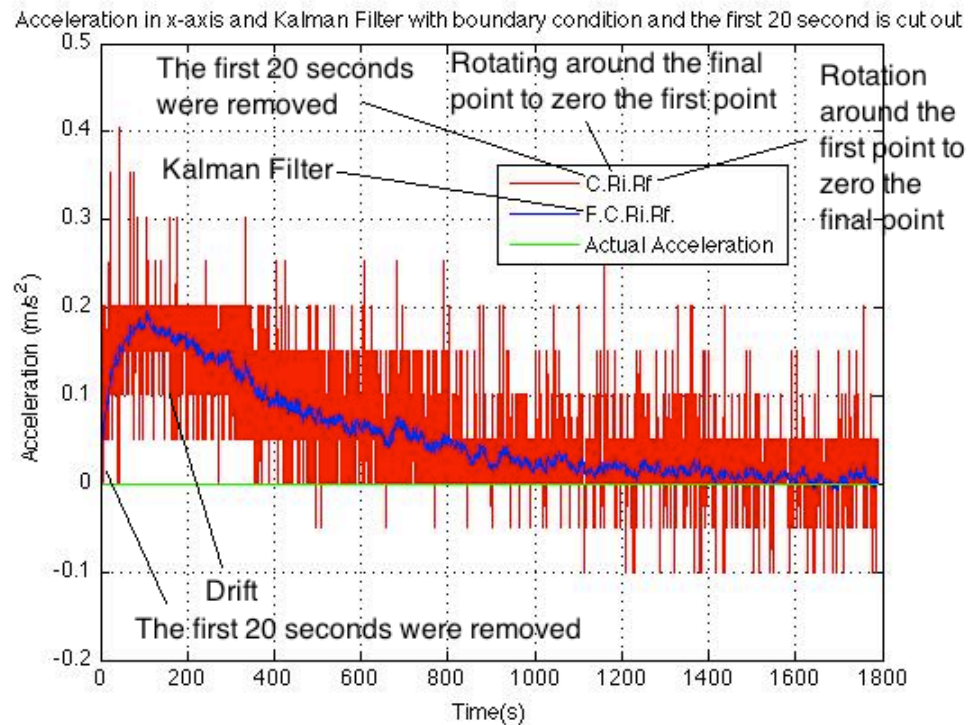


Figure 5-11: Plot of Measured and Actual Acceleration and the Kalman Filter Output after applying all the Techniques

Next, integration step was applied, and the results shown in Fig. 5.12. Note the caption shown in the top-left of the plot, where the letter “I.” has been added. This “I.” means the data was integrated after the earlier processing steps. From this graph, we can see that the initial velocity is zero as expected, but the final velocity has drifted significantly as time progresses. At around 800 seconds, it is possible to see the difference in output between the processed raw data, and the processed Kalman filtered data. Also shown is the true velocity, with the green line.

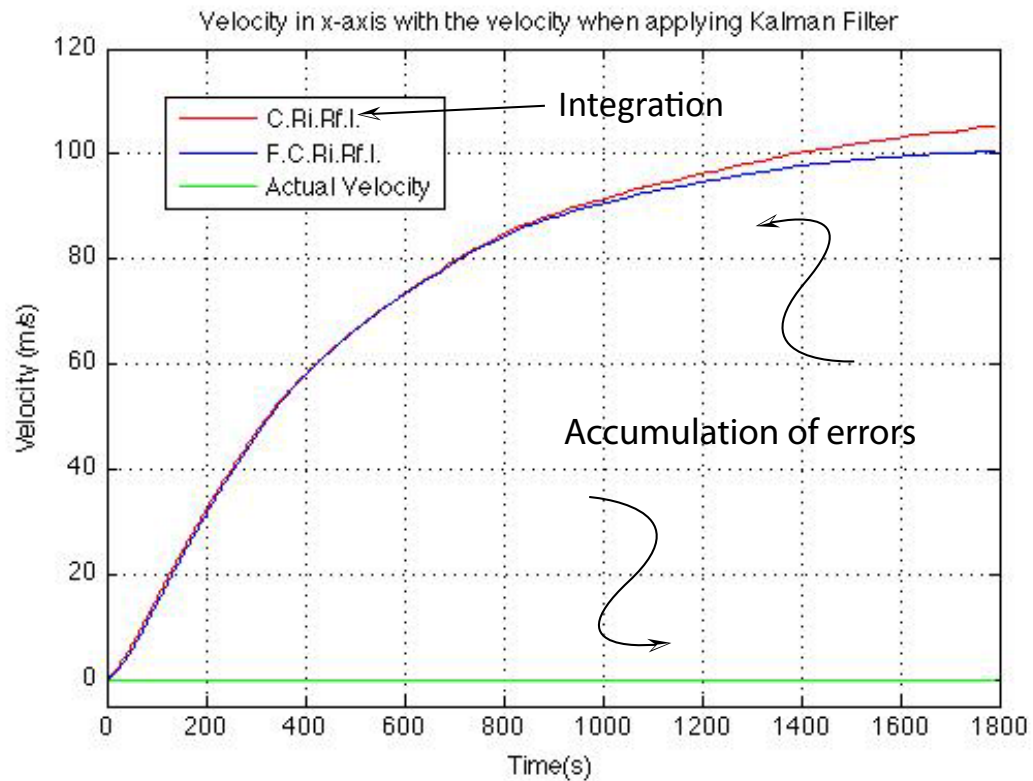


Figure 5-12: Plot of Measured, Actual and the Kalman Filter Velocities after applying all the Techniques

From Fig. 5.12, it can be seen that there was no significant improvement between the raw measured velocity and the Kalman filter velocity and, therefore, there was still a very large error after half an hour. The next step was to integrate the data again, and the resulting displacement vs. time plot is shown in Fig. 5.13. Note the caption shown in the top-left of the plot, where there are now two integrations, I.I. that were done after initial processing steps. However, there was not much improvement after double integrating the acceleration, and both the raw measurement and Kalman filter data results had significant error.

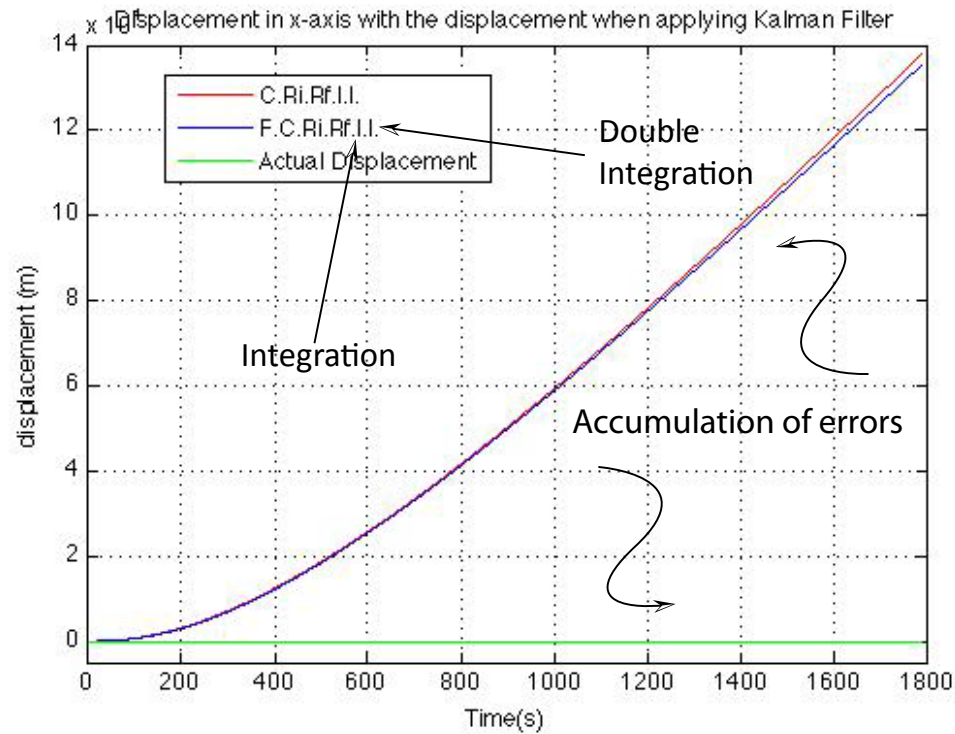


Figure 5-13: Plot of Measured, Actual and the Kalman Filter Displacements after applying all the Techniques

Since the result of Fig. 5.13 is still poor, an additional processing step to implement another boundary condition was done. This step sets the final velocity to zero. In other words, the final velocity of Fig. 5.12 was forced to zero, by rotating the whole plot, as described by Equation (5.4) (VRf). Fig. 5.14 shows the resulting velocity vs. time plot. Note the caption showing the processing step sequence used for the various plots. The plot shows the original velocity plot from Fig. 5.12 (solid lines) and the new rotated velocity plot (dashed lines).

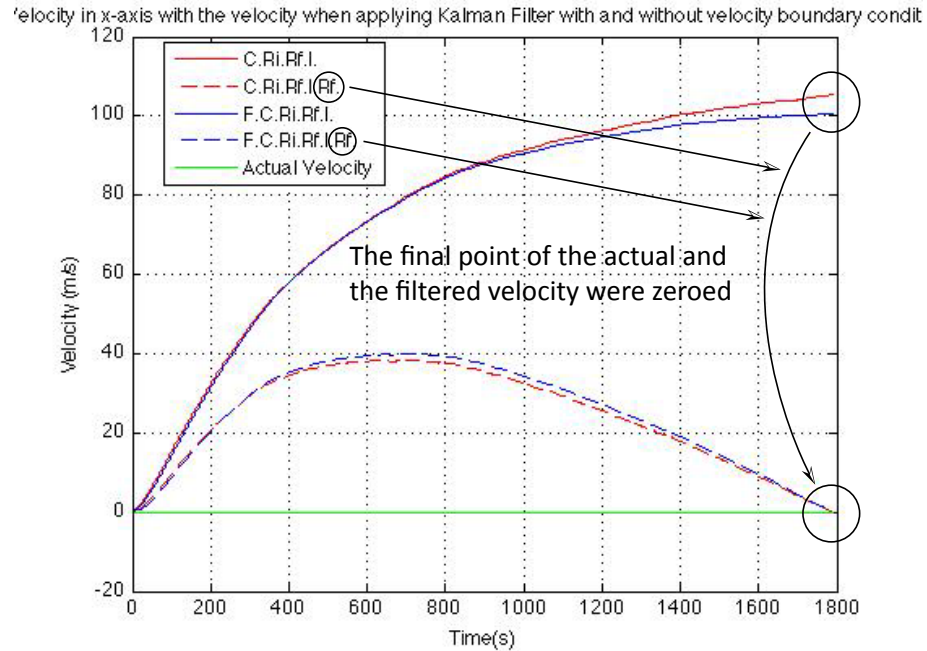


Figure 5-14: Plot of Measured, Actual and the Kalman Filter Velocities after applying all the Techniques and the Third Boundary Condition was applied on the measured and the Kalman Filter Velocities

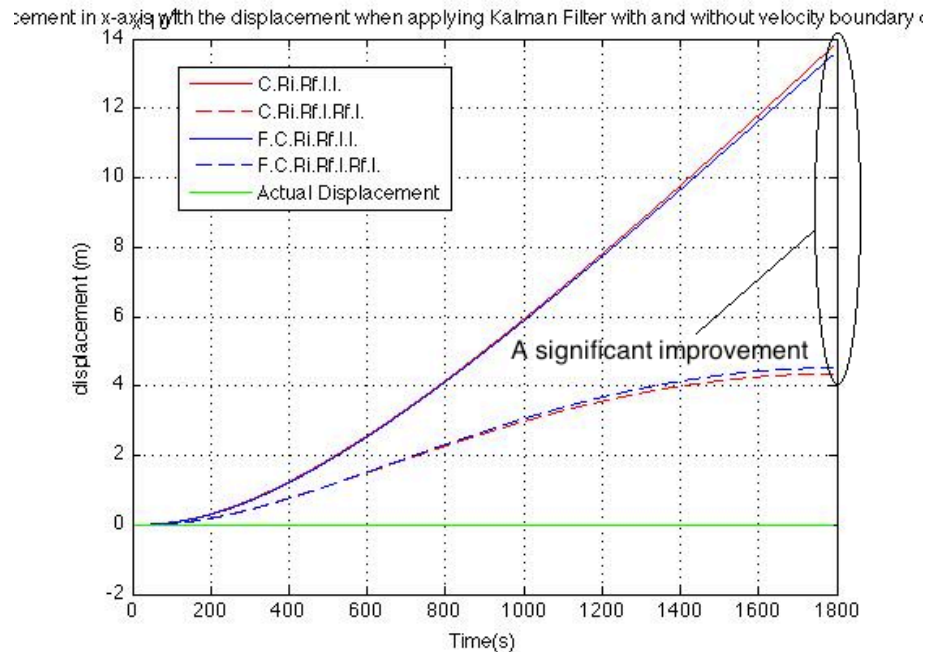


Figure 5-15: Plot of Measured, Actual and the Kalman Filter Displacements after applying all the Techniques and the Third Boundary Condition was applied on the measured and the Kalman Filter Velocities

The new velocity or the velocity after the third boundary condition was applied was integrated and it was indicated with similar dotted lines. Fig. 5.15 shows the result of integration of Fig. 5.14. The displacement estimate after the velocity plot rotation (dashed line) shows a significant improvement comparison to the displacement estimate (solid line) where VRf was not applied. This was in this particular case and after reviewing the results and comparisons section for all the experiments, the best values will be considered.

5.1.3 Applying the Moving Average Filter

A second approach was investigated, which is the use of the moving average filter to the measured acceleration data. Three types of moving average filters were studied: 11-sample size, 21-sample size and 41-sample size. Then the same techniques described in section 5.1.2 were applied.

Fig. 5.16, 5.17, 5.18 show the measured acceleration data (in red) after the boundary conditions, described in section 5.1.2, were applied. These boundary conditions are that the initial and final acceleration are assumed as zero. However, 11-sample size moving average filter, 21-sample size moving average filter and 41-sample size moving average filter were applied to that data, resulting in the black lines shown, where a lot of noise is removed.

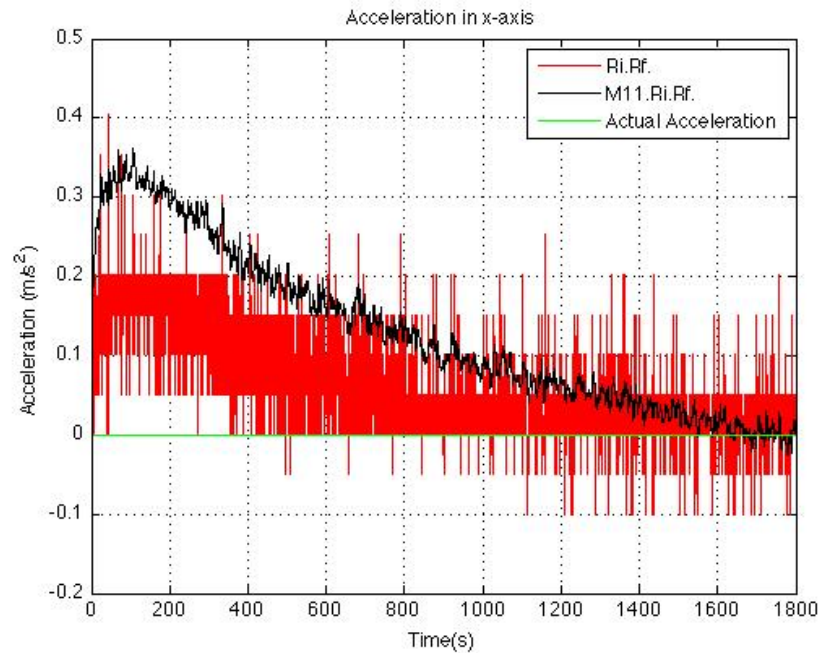


Figure 5-16: Plot of Measured and Actual Acceleration, and the 11-Sample Size Moving Average Filter output

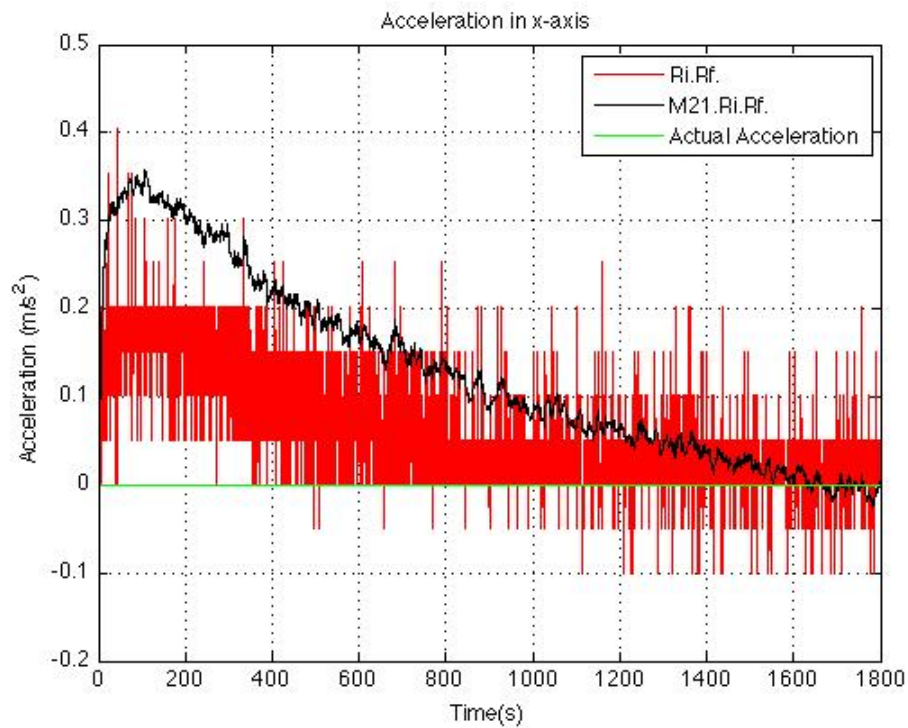


Figure 5-17: Plot of Measured and Actual Acceleration, and the 21-Sample Size Moving Average Filter output

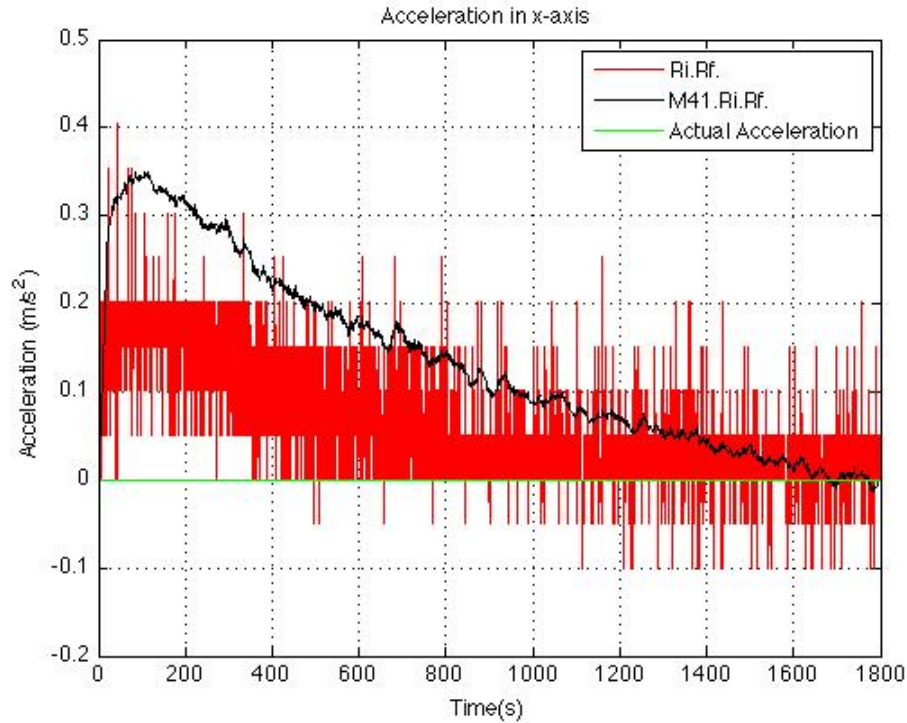


Figure 5-18: Plot of Measured and Actual Acceleration, and the 41-Sample Size Moving Average Filter output

After applying the moving average filter with 11-sample size, M11, 21-sample size, M21, and 41-sample size, M41, the results are shown in Fig. 5.16, 5.17, 5.18 (in black). Note the caption shown in the top-right of the plot in the box. The red line is denoted as “Ri.Rf.” This means that the two acceleration rotation formulas were applied, Ri and Rf to the measured data. Similarly, the black lines were denoted as “M11.Ri.Rf”, “M21.Ri.Rf” and “M41.Ri.Rf” which is almost the same, except that the moving average filter with the different sample sizes (M11, M21, M41), were applied first before the other processing. As can be seen both the red lines and the black lines clearly start at value zero, and end at value zero.

From Fig. 5.16, 5.17, 5.18, it was observed that the plots of the moving average filter (in black) with 11-sample size (M11), 21-sample size (M21), and 41-sample size (M41) have different white noise errors. The moving averaging filter with 11-sample size (M11) had the highest white noise error, followed by the 21-sample size (M12), and then the 41-sample size, which had the lowest white noise error among the three moving averaging filters. That means, the larger the sample size, the less the white noise error.

5.1.4 Kalman Filter and Standard Averaging Techniques

Comparisons between the Kalman filter and the moving average filter with different sample sizes: 11-sample size, 21-sample size, and 41-sample size were investigated.

In the comparisons, different techniques proposed to reduce the offset and drift error were applied to the measured acceleration data, the Kalman filter output, and the moving average filter output. These techniques are described in section 5.1.2: The initial and final acceleration are assumed as zero, and the first 20 seconds is cut.

Cutting the first 20 seconds of the moving average filter outputs was investigated and the results were that in 96% of the 48 experiments had a very high white noise error.

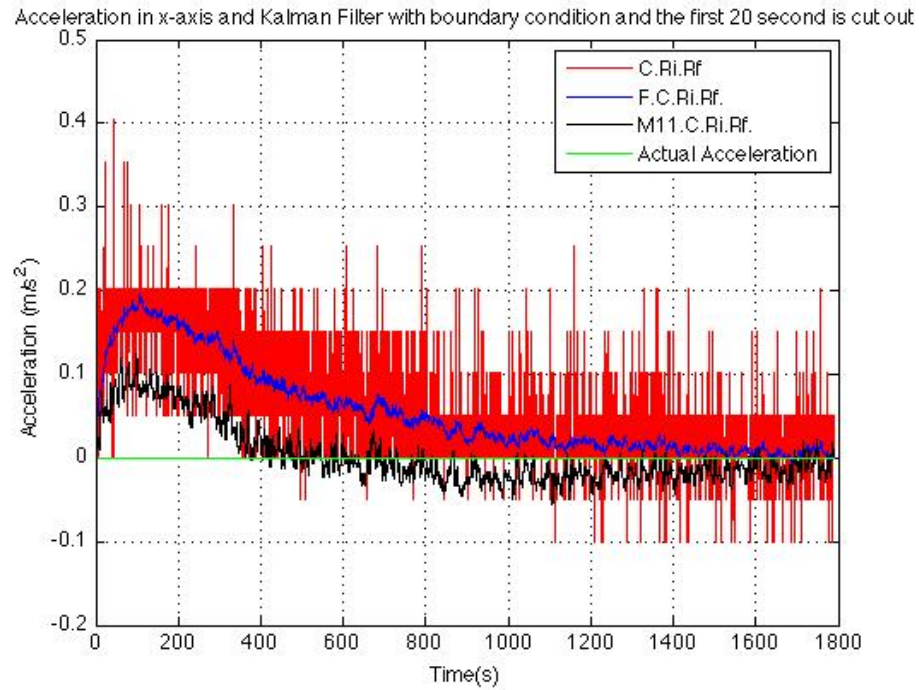


Figure 5-19: Plot of Measured and Actual Acceleration, and the Kalman filter and 11-Sample Size Averaging output

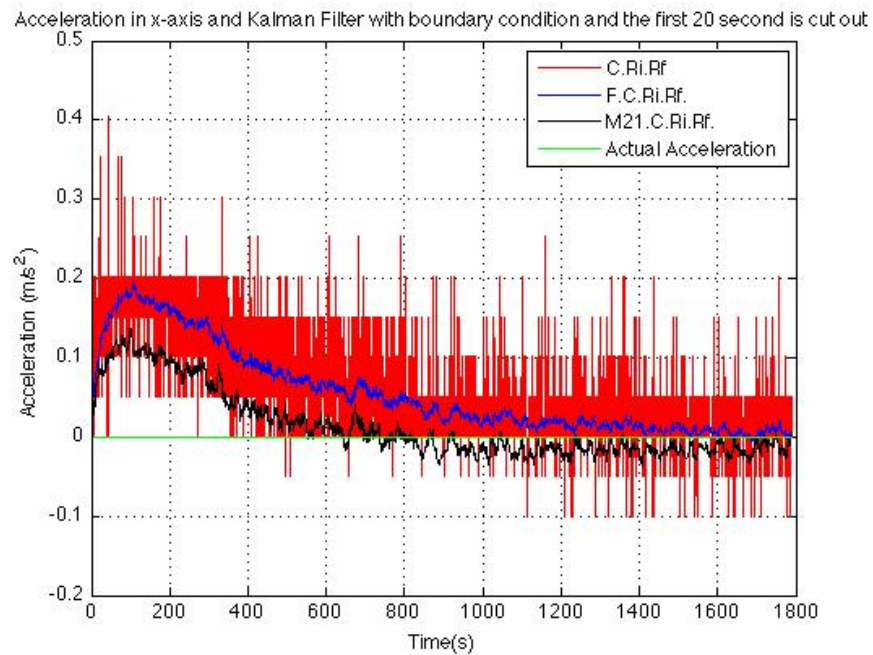


Figure 5-20: Plot of Measured and Actual Acceleration, and the Kalman filter and 21-Sample Size Averaging output

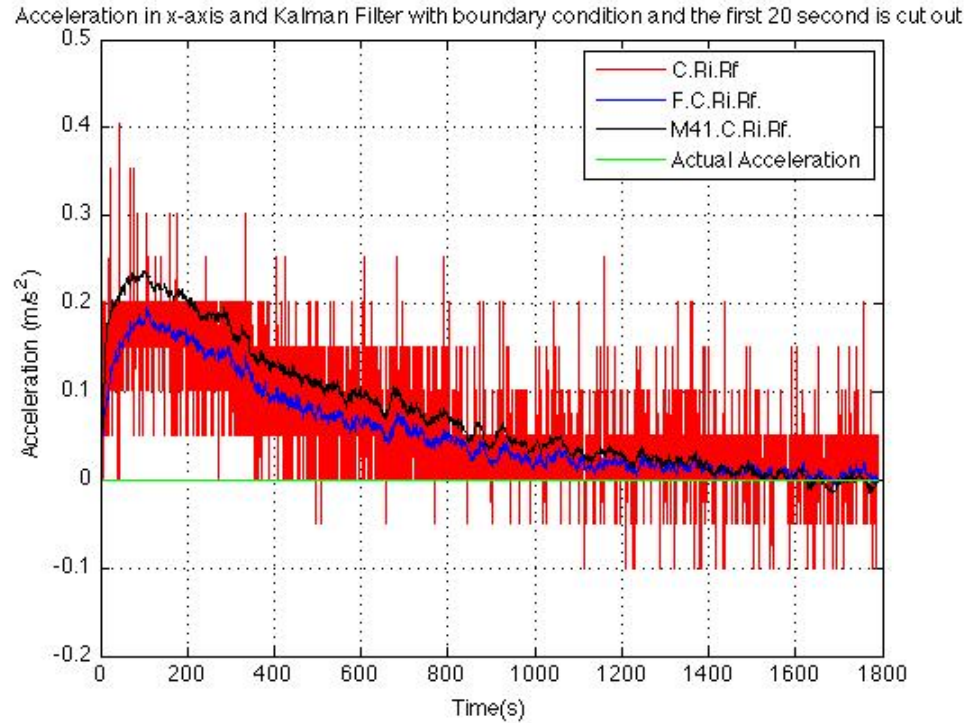


Figure 5-21: Plot of Measured and Actual Acceleration, and the Kalman filter and 41-Sample Size Averaging output

Fig. 5.19, 5.20, 5.21 show the measured accelerations signal (in red), the Kalman filter (in blue) and moving average filter with 11-sample size, 21-sample size and 41-sample size (in black). Note the caption shown in the top-right of the plot in the box. The red line is denoted as “C.Ri.Rf.” This means the raw acceleration data had a 20 second cut, C, and next the two acceleration rotation formulas were applied, Ri and Rf. Similarly, the blue line is denoted as “F.C.Ri.Rf” which is almost the same, except that the Kalman filter (F) is applied first before the other processing. Likewise, the black line is denoted as “M11.C.Ri.Rf,” “M21.C.Ri.Rf,” and “M41.C.Ri.Rf,” which means that the moving average filter (M11, M21, M41) are applied first before the other processing.

It can be observed from Fig. 5.19, 5.20 that the 11-sample size and 21-sample size moving average plots (black) had less drifting error than the Kalman filter plot (blue). However, they had more white noise error than the Kalman filter plot. From Fig. 5.21 the 41-sample size averaging plot had more drifting error than the Kalman filter plot and both almost had the same noise error. It can also be observed from the three figures that the Kalman filter plot had more constancy with the measured acceleration signals than the moving average filter plots.

The numerical comparisons between the different types of filters and the results were discussed in chapter (7).

5.2 Multiple Accelerometers Experiment

The second part of the experiment was to set a number of redundant accelerometers then combine the measured signals into a single reading. As explained earlier in the device set-up section (4.1.3), four accelerometers were placed together and they all were perfectly parallel to the ground. The measurements of the accelerometers were taken Simultaneously and stored into a computer station for analyzing.

Fig. 5.22 shows the acceleration measurement signals of the four accelerometers on the same graph and it was observed that they had different offset, drift, and white noise error values. The offset error was the same for all the 48 experiments and, on the other hand, the drift and white noise errors varied.

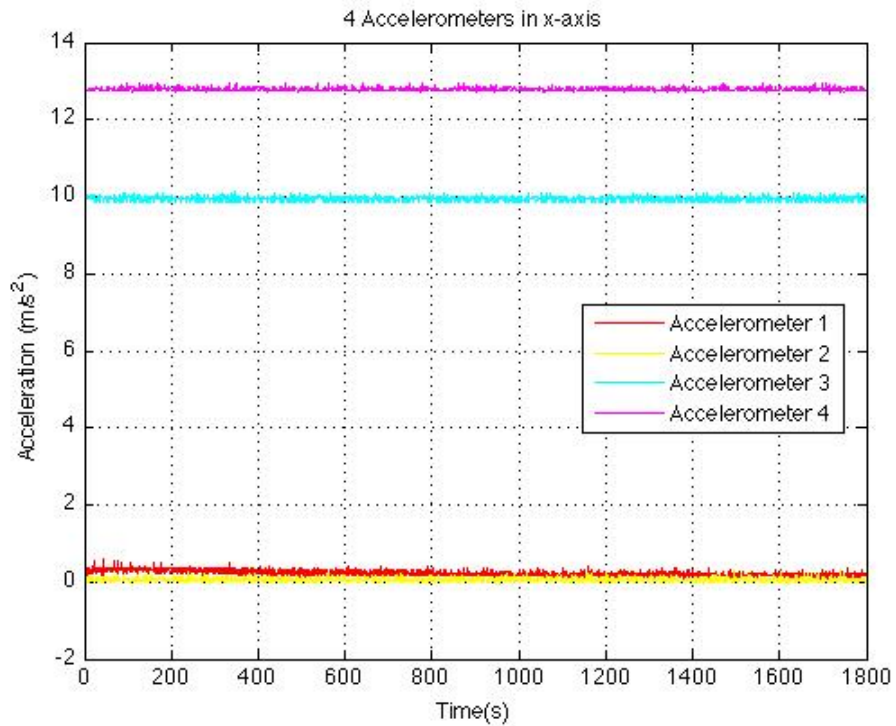


Figure 5-22: Plot of Measured Acceleration of Four Accelerometers

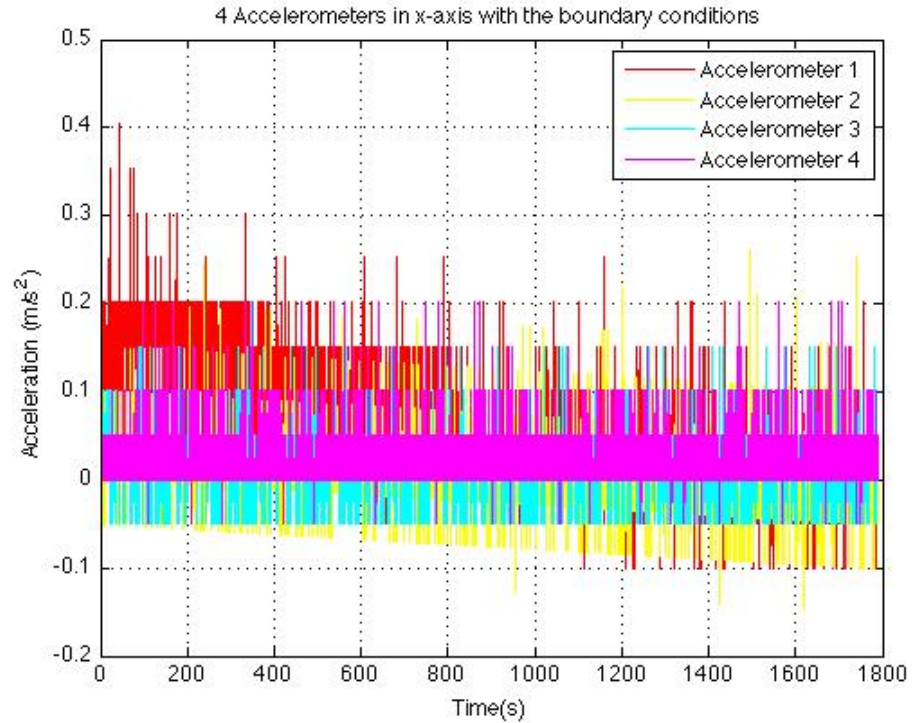


Figure 5-23: Plot of Measured Acceleration of the Four Accelerometers after applying the Boundary Conditions

After applying two boundary conditions, in which the first and the last acceleration values were zero, the four accelerometers' plots would be on top of each other because the offset error had been removed as shown in Fig. 5.23.

Different approaches had been investigated. The first approach was to take the average of the measurement signals to get a single reading then apply the Kalman filter as shown in Fig. 5.24. This technique is defined as follows:

$$\text{AFCRiRfIVRfI is defined as: } \left. \begin{array}{l} \text{Accel. 1} \\ \text{Accel. 2} \\ \text{Accel. 3} \\ \text{Accel. 4} \end{array} \right\} \text{Average Accel.} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \rightarrow VR_f \rightarrow I$$

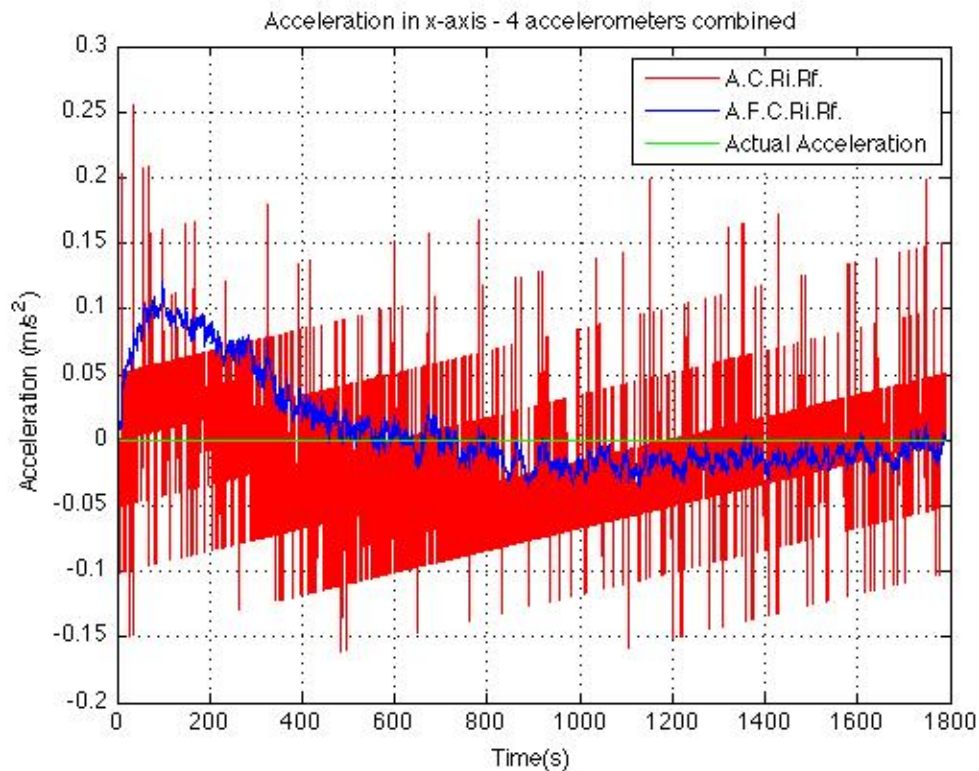


Figure 5-24: The Average of the Acceleration Readings

Fig. 5.24 shows the average of the measured acceleration signals (red) and the Kalman filter (blue). Note the caption shown in the top-right of the plot in the box. The red line is denoted as “A.C.Ri.Rf.” This means the four accelerometers signals was averaged, A, then the data had a 20 second cut, C, and next the two acceleration rotation formulas were applied, Ri, and, Rf. Similarly, the blue line is denoted as “F.A.C.Ri.Rf” which is almost the same, except that the Kalman filter (F) is applied first before the other processing.

The second approach was to apply the Kalman filter to each measurement acceleration signal individually then the average of the four filters are taken, such as in Fig. 5.25. The process technique can be define as follows:

$$\text{FACRiRfIVRfI is defined as: } \left. \begin{array}{l} \text{Accel. 1} \rightarrow F \\ \text{Accel. 2} \rightarrow F \\ \text{Accel. 3} \rightarrow F \\ \text{Accel. 4} \rightarrow F \end{array} \right\} \text{Average Accel.} \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \rightarrow VR_f \rightarrow I$$

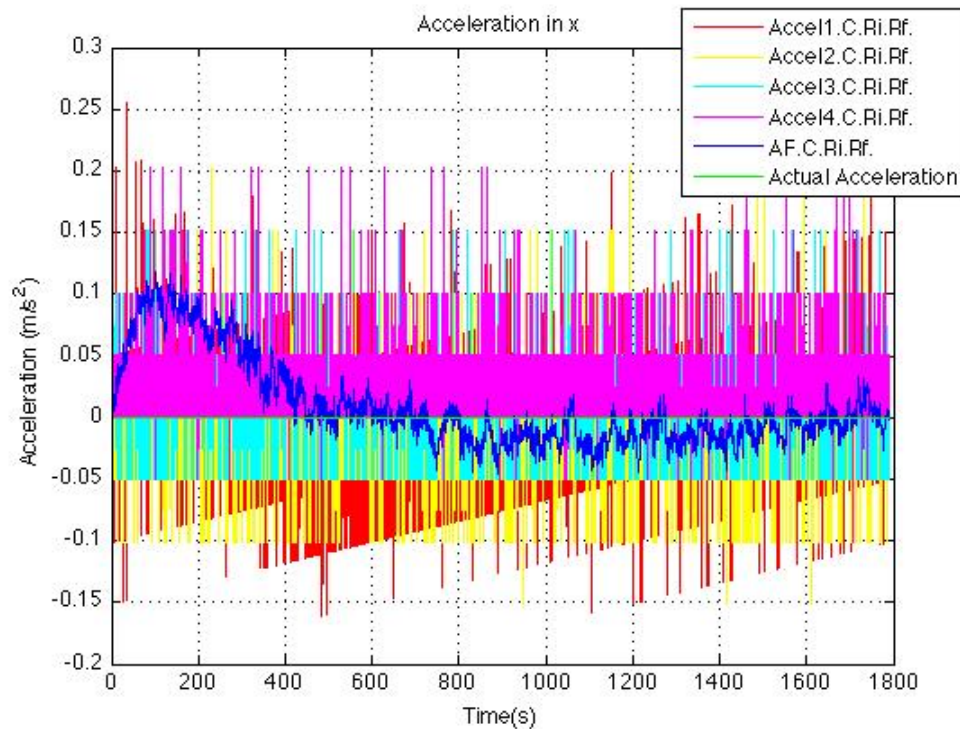


Figure 5-25: The Average of the Four Filtered Accelerations

Fig. 5.25 shows the measured acceleration signals of the four accelerometers (red, yellow, cyan, and magenta) and the Kalman filter (blue). Note the caption shown in the top-right of the plot in the box. The red line is denoted as “Accel1.C.Ri.Rf.” This means the first accelerometer signals, Accel1 had a 20 second cut, C, and next the two acceleration rotation formulas were applied, Ri, and, Rf. Similarly, the other three accelerometers. The blue line is denoted as “AF.C.Ri.Rf” which is almost the same, except that the Kalman filter (F) is applied first to every accelerometer signal individually then the average of the filter was taken, AF.

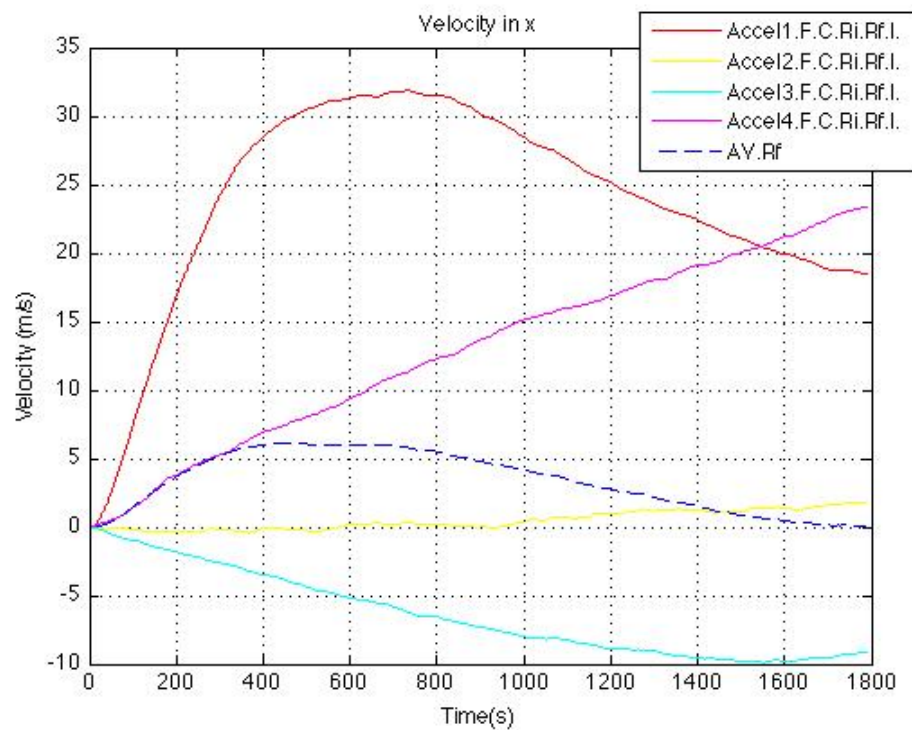


Figure 5-26: The Average of the Four Velocities

The third approach was to apply the Kalman filter to each measurement acceleration signal, integrate once to get the velocity then take the average of the four velocity data into one velocity reading. This is shown in Fig. 5.26, and it is defined by the following:

$$\text{FCRiRfIAVVRfI is defined as: } \left. \begin{array}{l} \text{Accel. 1} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \\ \text{Accel. 2} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \\ \text{Accel. 3} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \\ \text{Accel. 4} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \end{array} \right\} \text{Average Vel.} \rightarrow VR_f \rightarrow I$$

Fig. 5.26 shows the measured velocity signals of the four accelerometers (red, yellow, cyan, and magenta) and the Kalman filter (dotted blue). Note the caption shown in the top-right of the plot in the box. The red line is denoted as “Accel1.F.C.Ri.Rf.” This means that the Kalman filter was applied on the first accelerometers signal, the it had a 20 second cut, C, and next the two acceleration rotation formulas were applied, Ri, and, Rf. Then it was integrated once, I. Similarly, the other three accelerometers. The dotted blue line is denoted as “AV.Rf” which means that the four velocity signals were averaged, AV, then the third boundary condition was applied, VRf.

The last approach was to integrate each velocity data individually. Then the boundary condition, VRf, was applied. Next, take the average of the four displacements as shown in Fig. 5.27 and it is defined as:

$$\text{FCRiRfIVRfIAD is defined as: } \left. \begin{array}{l} \text{Accel. 1} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \rightarrow VR_f \rightarrow I \\ \text{Accel. 2} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \rightarrow VR_f \rightarrow I \\ \text{Accel. 3} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \rightarrow VR_f \rightarrow I \\ \text{Accel. 4} \rightarrow F \rightarrow C \rightarrow R_i \rightarrow R_f \rightarrow I \rightarrow VR_f \rightarrow I \end{array} \right\} \text{Average Disp.}$$

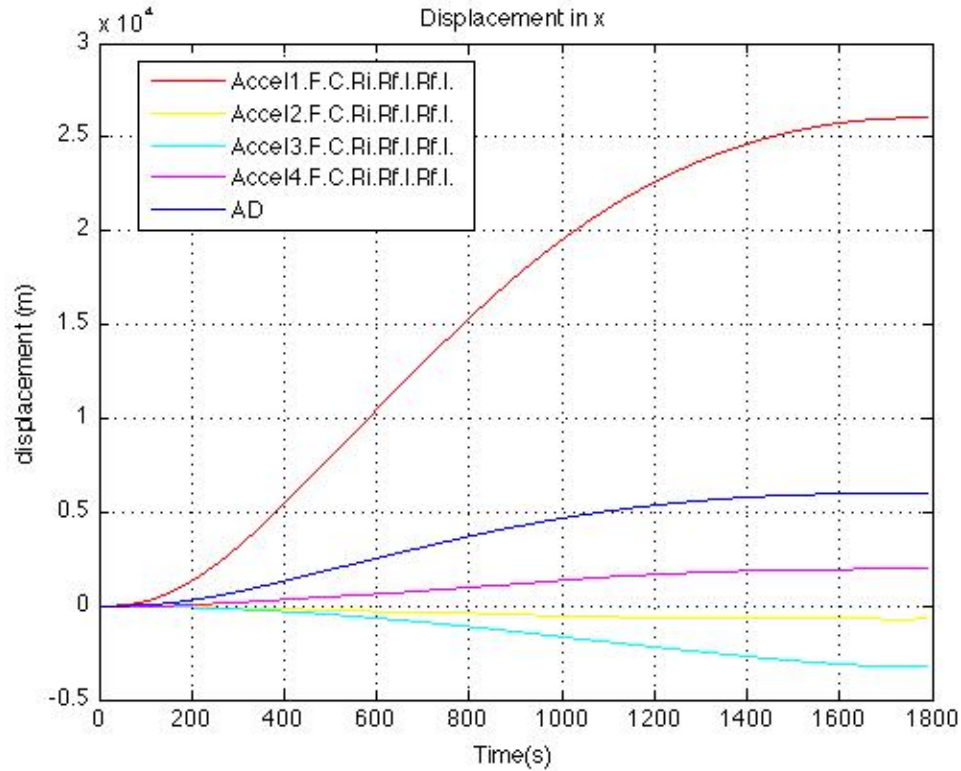


Figure 5-27: The Average of the Four Displacements

Fig. 5.27 shows the measured displacement signals of the four accelerometers (red, yellow, cyan, and magenta) and the Kalman filter (blue). Note the caption shown in the top-right of the plot in the box. The red line is denoted as “Accel1.F.C.Ri.Rf.I.Rf.I.” This means that the Kalman filter was applied on the first accelerometers signal, then it had a 20 second cut, C, and next the two acceleration rotation formulas were applied, Ri, and, Rf. Then it was integrated once, I, and the velocity rotation formula was applied, VRf. Then it was integrated again, I. Similarly, the other three accelerometers. The blue line is denoted as “AD” which means that the four displacement signals were averaged, AD.

5.2.1 Proposed Methods

The proposed method was to design a number of Kalman filter algorithms that were used to combine the multiple acceleration signals into a single output. Three different Kalman filter algorithms were designed:

5.2.1.1 The Parallel Design [38]

The first Kalman filter algorithm design combined multiple accelerometer measurement signals in parallel. All the output measurements formed a an output vector, as shown in Fig. 5.28:

$$Z_k = \begin{bmatrix} Z_{k,1} \\ Z_{k,2} \\ Z_{k,3} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ Z_{k,I} \end{bmatrix}$$

Figure 5-28: Parallel Design Output Vector

Where:

$Z_{k,i}$: The output measurement

I: The number of output measurements

Z_k : Is an m-vector, which has all the output readings and from it is called the output of the system

The process noise covariances ($Q_{k,1}, Q_{k,2}, \dots$) for all the measurement signals were assumed to be very small constant values, which were equal to $Q_{k,i} = 1 \times 10^{-5}$ v. The measurement error covariance for each measurement signal was equal to the measurement's variance. It was also assumed that all the measurement error covariance values were not correlated, and therefore the measurement error covariance could be described by Fig. 5.29 as:

$$R_k = \begin{bmatrix} R_{k,1} & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & R_{k,2} & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & R_{k,3} & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_{k,I} \end{bmatrix}$$

Figure 5-29: The Error Covariance Matrix

Where:

$R_{k,i}$: Is the measurement error covariance for each output signal

R_k : Is an $n \times m$ matrix for the all the measurement error covariances and it is called the system's measurement error covariance

I: The number of output measurements

The next step was to design the Kalman filter equations, which were as follows:

The state:

$$\hat{x}_k = \hat{x}_k^- + \sum_{i=1}^I K_{k,i} (Z_{k,i} - H_{k,i} \hat{x}_k^-) \quad (5.7)$$

The gain:

$$K_{k,i} = P_k H_{k,i}^T R_{k,i}^{-1} \quad (5.8)$$

The covariance:

$$P_k^{-1} = (P_k^-)^{-1} + \sum_{i=1}^I H_{k,i}^T R_{k,i}^{-1} H_{k,i} \quad (5.9)$$

Where:

$\hat{x}_k$: Is the updated system state estimate for the system

$\hat{x}_k^-$: Is the system state estimate of the system from the previous step

$K_{k,i}$: Is the Kalman gain for the measurement output

$z_{k,i}$: Is the measurement output

P_k : Is the updated error covariance of the system

P_k^- : Is the error covariance of the system from the previous step

$H_{k,i}$: Is the output transition matrix

$R_{k,i}$: Is the measurement error covariance

The error covariance was obtained first from Equation (5.3). The inverse of the updated error covariance of the system was equal to the inverse of the previous error covariance of the system plus the sum of the output transition matrix transpose, multiplied by the inverse of the error covariance matrix, multiplied by the output transition matrix for all the output readings. After the error covariance of the system is obtained it is substituted into Equation (5.2) and multiplied by the output transition matrix and the inverse of the covariance error for each measurement readings to get a gain for each measurement reading. As Equation (5.1) illustrates, the output transition matrix was multiplied by the previous system state estimate was then the result was subtracted from the measurement

output reading, and then it was multiplied by the measurement gain. The whole value was summed for each measurement reading and then the total was added to the previous system state estimate and the updated system state estimate is finally obtained. Fig. 5.30 shows an illustration of the designed filter.

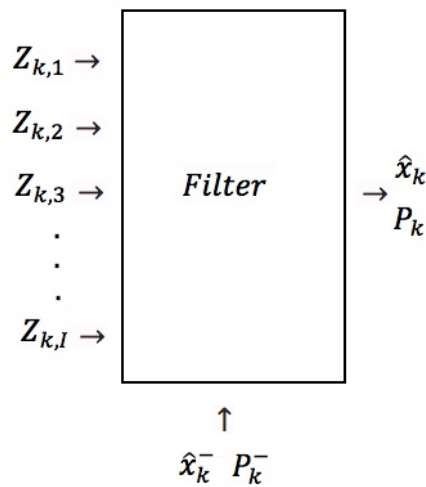


Figure 5-30: The Parallel Design Diagram

In this design, all the measurement outputs inputted to the filter as well as the initial system state and the error covariance estimate, and the output of the filter was the updated system state and the error covariance estimate.

Fig. 5.31 shows the four accelerometers output readings and the parallel design Kalman filter (PF) as indicated in the graph. It was observed that the drift between 0 and 800 seconds was almost gone and the filter behaviour was much better comparing to the normal average.

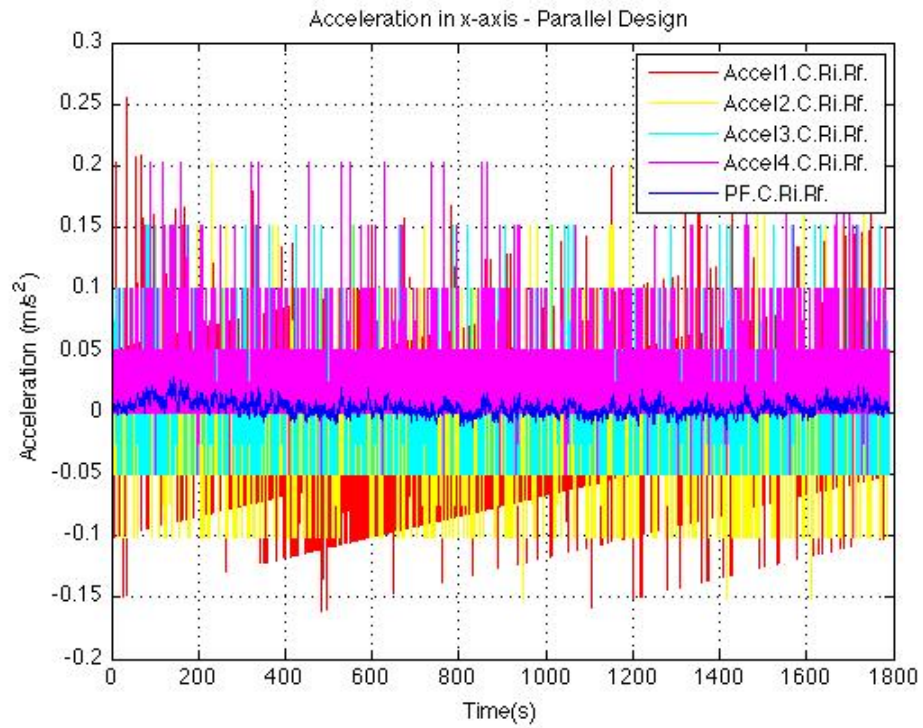


Figure 5-31: The Parallel Design Acceleration

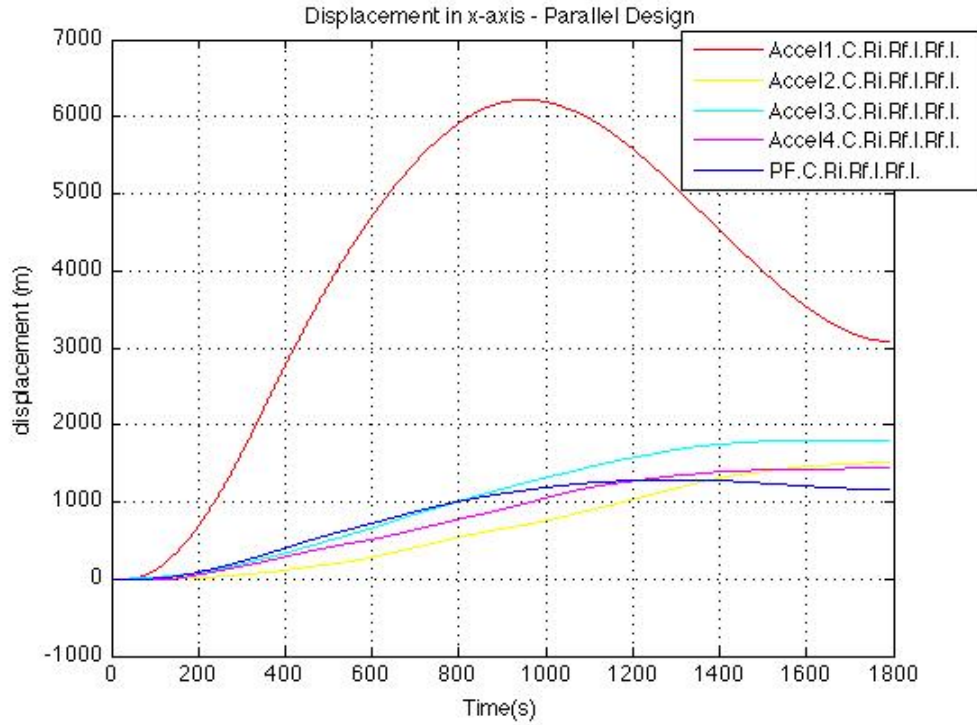


Figure 5-32: The Parallel Design Displacement

Fig. 5.32 shows the displacement graph of the four measurement readings and the parallel design filter displacement. It was also shown that the filter gave the best value.

5.2.1.2 The Series Design [38]

In this design, the first output measurement was implemented into the filter along with the initial system state and error covariance estimate, in which they were considered to be the system state and error covariance estimate at measurement zero. Then they were updated and implemented into the filter again along with the second output measurement and so on. After the last output measurement was implemented into the filter with the updated system state and error covariance from the previous step, the final system state and error covariance were obtained and they were considered the system state and the error covariance estimate as illustrated in Fig. 5.33.

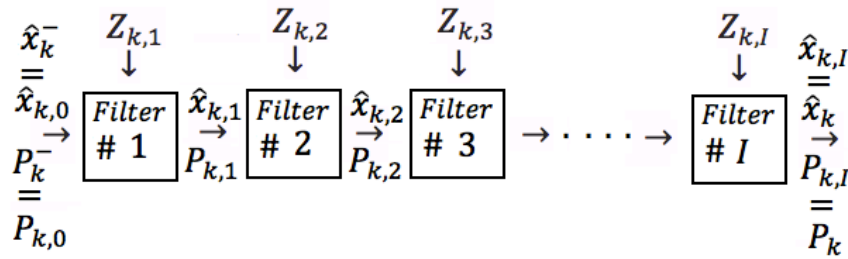


Figure 5-33: The Series Design Diagram

Then the Kalman filter equations were applied as follows:

$$P_{k,0} = P_k^- \quad (5.10)$$

$$\hat{x}_{k,0} = \hat{x}_k^- \quad (5.11)$$

$$\hat{x}_{k,i} = \hat{x}_{k,i-1} + K_{k,i}(Z_{k,i} - H_{k,i}\hat{x}_{k,i-1}) \quad (5.12)$$

$$K_{k,i} = P_{k,i} H_{k,i}^T R_{k,i}^{-1} \quad (5.13)$$

$$P_{k,i}^{-1} = P_{k,i-1}^{-1} + H_{k,i}^T R_{k,i}^{-1} H_{k,i} \quad (5.14)$$

$$i = 1, 2, \dots, I \quad (5.15)$$

$$P_k = P_{k,I} \quad (5.16)$$

$$\hat{x}_k = \hat{x}_{k,I} \quad (5.17)$$

Where:

$P_{k,0}$: The initial error covariance estimate

P_k^- : The error covariance estimate from the previous step

$\hat{x}_{k,0}$: The initial system state estimate

$\hat{x}_k^-$: The system state estimate from the previous step

$\hat{x}_{k,i}$: The updated system state estimate

$K_{k,i}$: The gain of the measurement

$Z_{k,i}$: The output readings of the measurement

$H_{k,i}$: The output transition matrix

$P_{k,i}$: The updated error covariance estimate

$R_{k,i}$: The measurement error covariance

P_k : The error covariance estimate of the system

$P_{k,I}$: The error covariance estimate of the final measurement

$\hat{x}_K$: The system state estimate of the system

$\hat{x}_{k,I}$: The system state estimate of the final measurement

First, the initial system state and error covariance estimate were equalled to the system state and error covariance estimate from the pervious step. Then the updated error covariance estimate was obtained, in which the inverse of the updated error covariance

estimate was equal to the previous measurement of the error covariance estimate, plus the result of multiplying the transpose of the output transition matrix, the inverse of the measurement error covariance and the output transition matrix. Then the gain was obtained by multiplying the updated error covariance estimate, the transpose of the output transition matrix and the inverse of the measurement error covariance. Finally, the previous measurement of the system state estimate was multiplied by the output transition matrix, then they were subtracted from the output measurement and the result was multiplied by the Kalman gain and added the previous measurement of the system state estimate to obtain the updated system state estimate.

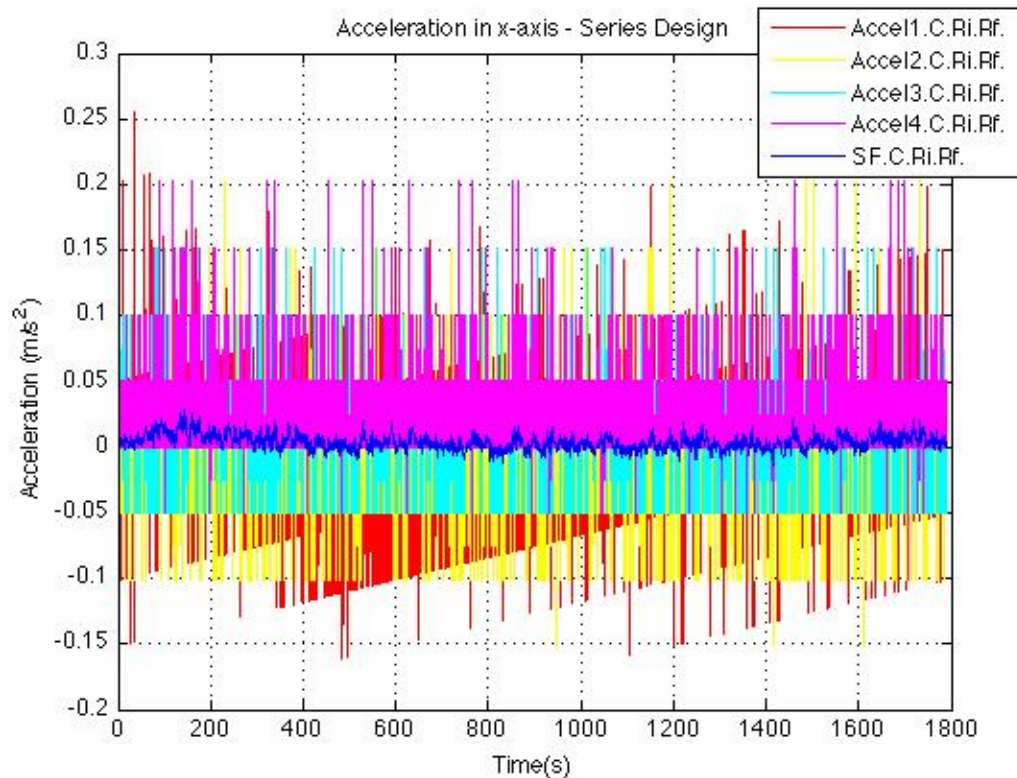


Figure 5-34: The Series Design Acceleration

Fig. 5.34 shows the four accelerometer's output readings and the series design of the Kalman filter, which was indicated by (SF) in the graph.

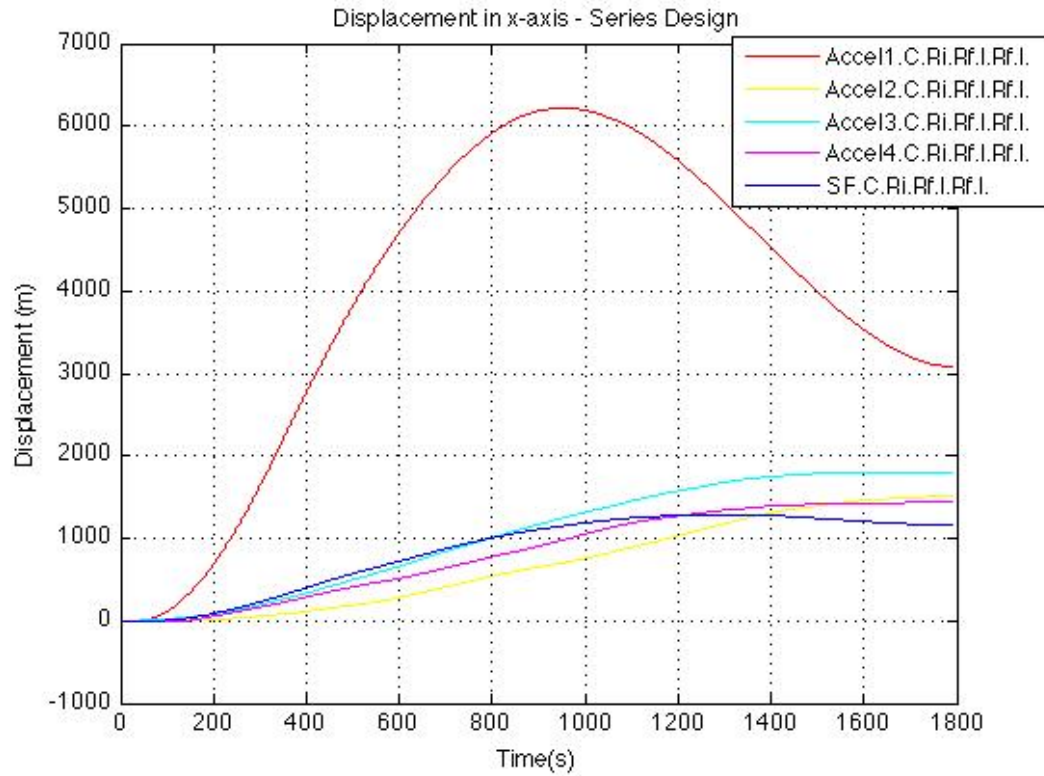


Figure 5-35: The Series Design Displacement

Fig. 5.35 shows the displacement of the four output readings in comparisons with the displacement of the series design Kalman filter's displacement.

5.2.1.3 The Compression Design [38]

The last design was the compression design, in which all the output measurements were compressed and a single output measurement was obtained, then it was

implemented into the Kalman filter and the output would be the final system state and error covariance estimate as shown in Fig. 5.36.

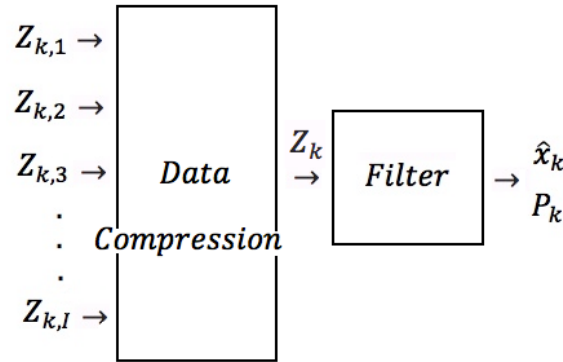


Figure 5-36: The Compression Design Diagram

The equations of the compression design was as follows:

$$Z_k = R_k \left(\sum_{i=1}^I R_{k,i}^{-1} Z_{k,i} \right) \quad (5.18)$$

$$R_k^{-1} = \sum_{i=1}^I R_{k,i}^{-1} \quad (5.19)$$

Where:

Z_k : The output measurement of the system

R_k : The error covariance estimate of the system

$R_{k,i}$: The error covariance estimate

$Z_{k,i}$: The output measurement

First, the error covariance estimate was calculates, in which the inverse of the total error covariance estimate was equal to the inverse of the total error covariance estimate.

The total output measurement can be obtained by taking the total values of the

multiplications of each measurement outputs by the inverse of each error covariance estimate. Then the final result was multiplied by the total error covariance estimate.

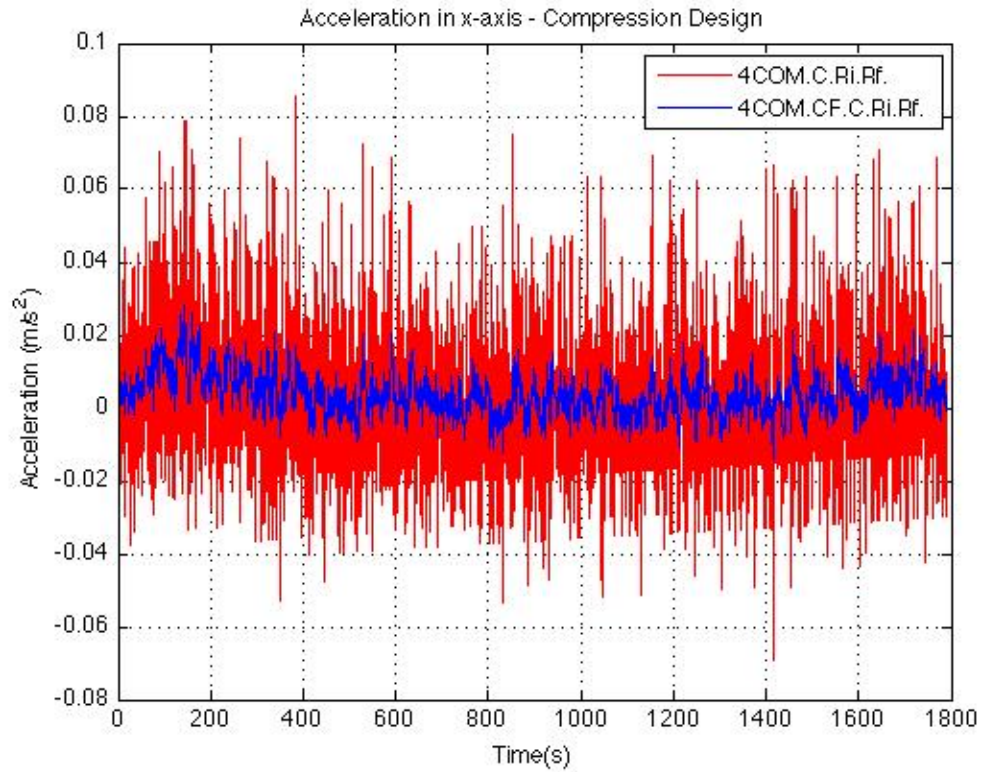


Figure 5-37: The Compression Design Acceleration

Fig. 5.37 shows the compression design of the Kalman filter. The red graph shows the single output signals, then when they were implemented into the Kalman filter, the blue graph produced.

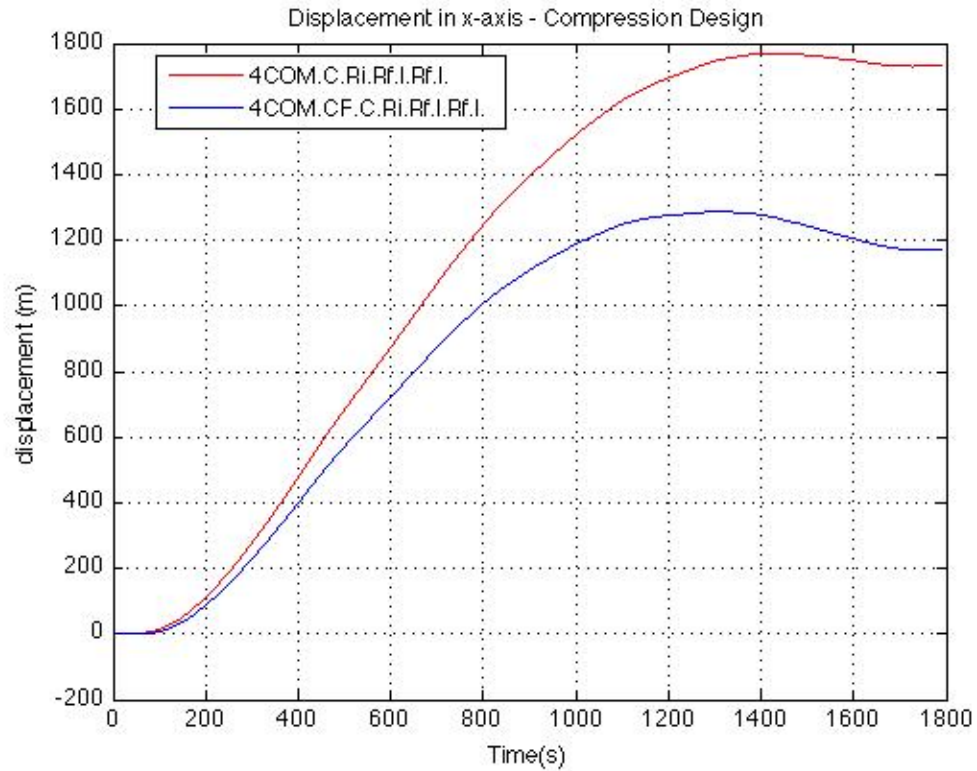


Figure 5-38: The Compression Design Displacement

Fig. 5.38 shows the displacement of the single output measurement, which it was indicated in red. And the displacement of the filtered signals using the compression design of Kalman filter and it was indicated in blue.

6 Results and Comparisons

A review of the experimental results is presented. First, the single sensor approach is considered, and secondly the multiple redundant sensors are considered. The results are presented in terms of “total estimated displacement” in units of meters, for different types of mathematical processing. Since all tests were conducted as static (motionless) tests, the true (actual) total displacement is equal to zero. Hence, those estimated displacement values that are closest to zero are considered as good results.

6.1 Single Sensor Approach

The single sensor approach results are presented in this section. Table 6.1 presents results of the total estimated displacements of 8 experiments from among the 48 experiments. The table shows four Raw Processing Techniques: (i) RiRfII, (ii) RiRfIVRfI, (iii) CRiRfII, (iv) CRiRfIVRfI. The letter codes for these notations are presented in Table 5.1. The technique that gives the estimated displacement values closest to the true displacement value (i.e. zero) is considered as the best technique. Also, Table 6.1 presents the best and the worst technique of the four Raw Processing Techniques in each experiment.

Exp. #	Raw Processing Techniques				Worst Technique out of the 4 Raw Processing Techniques		Best Technique out of the 4 Raw Processing Techniques	
	RiRfII	RiRfIVRfI	CRiRfII	CRiRfIVRfI				
1	-70,000	25,900	-15,300	39,400	RiRfII	70,100	CRiRfII	15,300
2	-75,500	-27,900	-19,800	-13,700	RiRfII	75,500	CRiRfIVRfI	13,700
3	4,290	-27,000	4,310	-26,600	RiRfIVRfI	27,000	RiRfII	4,290
4	116,000	11,800	6,630	-15,600	CRiRfIVRfI	116,000	CRiRfII	6,630
5	-66,900	-39,800	41,500	-12,500	RiRfII	66,900	CRiRfIVRfI	12,500
6	163,000	27,500	52,300	317	RiRfII	162,000	CRiRfIVRfI	317
7	-223,000	-54,300	-4,950	182	RiRfII	223,000	CRiRfIVRfI	182
8	200,000	13,400	89,300	-13,900	RiRfII	200,000	RiRfIVRfI	13,400

Table 6-1: Results of the Raw Processing Techniques and the Best and Worst Technique (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).

The Percentage of the Best Technique out of the 4 Raw Processing Techniques		Best Raw Processing Technique
RiRfII	1 out of 48	CRiRfIVRfI (58.3%)
RiRfIVRfI	6 out of 48	
CRiRfII	13 out of 48	
CRiRfIVRfI	28 out of 48	

Table 6-2: Results of the Best Raw Processing Techniques.

Table 6.2 presents the percentages of the best technique for the four Raw Processing Techniques out of the 48 experiments. In addition to that, the number of times each technique was the best was shown. The table also shows, the best Raw Processing Technique, which is the technique that had the highest percentage among the other techniques.

Exp. #	Moving Average Processing Techniques with 11-Sample Size			
	11MRiRfII	11MRiRfIVRfI	11MCRiRfII	11MCRiRfIVRfI
1	296,000	83,700	29,000	17,300
2	-448,000	-119,000	-2,630	-7,570
3	-632	-5,900	4,320	-4,590
4	-2,350,000	-587,000	11,500	349
5	12,000,000	2,980,000	-2,670	-1,480
6	11,700,000	2,910,000	5,640	1,520
7	11,700,000	2,900,000	4,840	2,630
8	11,400,000	2,840,000	10,700	-4,100

Table 6-3: Results of the Moving Average Processing Techniques with 11-Sample Size (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).

Exp. #	Moving Average Processing Techniques with 21-Sample Size			
	21MRiRfII	21MRiRfIVRfI	21MCRiRfII	21MCRiRfIVRfI
1	315,000	84,500	47,900	18,400
2	-463,000	-123,000	34,100	703
3	-3,440	-3,590	5,630	-1,280
4	-2,450,000	-612,000	-141,000	-39,000
5	12,500,000	3,100,000	604,000	150,000
6	12,200,000	3,020,000	622,000	156,000
7	12,200,000	3,020,000	593,000	148,000
8	11,900,000	2,960,000	590,000	145,00

Table 6-4: Results of the Moving Average Processing Techniques with 21-Sample Size (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).

Exp. #	Moving Average Processing Techniques with 41-Sample Size			
	41MRiRfII	41MRiRfIVRfI	41MCRiRfII	41MCRiRfIVRfI
1	322,000	84,500	186,000	51,500
2	-473,000	-121,000	-220,000	-58,500
3	-2,910	-3,350	1,700	-2,180
4	-2,500,000	-616,000	-1,320,000	-329,000
5	12,700,000	3,120,000	6,640,000	1,650,000
6	12,400,000	3,040,000	6,480,000	1,610,000
7	12,400,000	3,040,000	6,480,000	1,600,000
8	12,100,000	2,980,000	6,340,000	1,570,000

Table 6-5: Results of the Moving Average Processing Techniques with 41-Sample Size (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).

Tables 6.3, 6.4, 6.5 present the results of 8 experiments from among 48 experiments. These tables show Moving Average Processing Techniques for three sample sizes: 11-sample size, 21-sample size and 41-sample size. These Moving Average Processing Techniques were as follows: (i) 11MRiRfII, (ii) 11MRiRfIVRfI, (iii) 11MCRiRfII, (iv) 11MCRiRfIVRfI, (v) 21MRiRfII, (vi) 21MRiRfIVRfI, (vii) 21MCRiRfII, (viii) 21MCRiRfIVRfI, (ix) 41MRiRfII, (x) 41MRiRfIVRfI, (xi) 41MCRiRfII, (xii) 41MCRiRfIVRfI. The letter codes for these notations are described in Table 5.1.

Table 6.6 presents the percentages of the best technique of the Moving Average Processing Techniques, with the different sample sizes, out of the 48 experiments. In addition to that, the number of times each technique was the best was presented. The

table also shows, the best Moving Average Processing Technique out of the 12 techniques, which is the technique that had the highest percentage.

The Percentage of the Best Technique out of the 12 Moving Average processing Techniques		Best Moving Average Processing technique
11MRiRfII	1 out of 48	11MCRiRfIVRfI (77%)
11MRiRfIVRfI	0 out of 48	
11MCRiRfII	4 out of 48	
11MCRiRfIVRfI	37 out of 48	
21MRiRfII	0 out of 48	
21MRiRfIVRfI	0 out of 48	
21MCRiRfII	0 out of 48	
21MCRiRfIVRfI	5 out of 48	
41MRiRfII	0 out of 48	
41MRiRfIVRfI	1 out of 48	
41MCRiRfII	0 out of 48	
41MCRiRfIVRfI	0 out of 48	

Table 6-6: Results of the Best Moving Average Processing Techniques.

Exp. #	Kalman Single Processing Techniques			
	FRiRfII	FRiRfIVRfI	FCRiRfII	FCRiRfIVRfI
1	-20,800	2,020	31,300	15,200
2	-84,500	-26,300	42,200	6,020
3	-25,900	-12,200	6,370	-3,960
4	93,100	20,700	-10,500	-5,430
5	-72,700	-18,800	-13,900	-4,230
6	162,000	39,500	13,100	2,490
7	-207,000	-51,200	-7,720	-1,530
8	169,000	36,800	5,910	-3,860

Table 6-7: Results of the Kalman Single Processing Techniques (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).

Table 6.7 presents results of 8 experiments from among the 48 experiments. The table shows four Kalman Single Processing Techniques: (i) FRiRfII, (ii) FRiRfIVRfI, (iii) FCRiRfII, (iv) FCRiRfIVRfI. The letter codes for these notations are presented in Table 5.1. The percentage of the best techniques out of the four Kalman Single Processing Techniques and also the best Kalman Single Processing Technique among the four techniques was presented in Table 6.8

The Percentage of the Best Techniques out of the 4 Kalman Single Processing Techniques		Best Kalman Single Processing Technique
FRiRfII	0 out of 48	FCRiRfIVRfI (79.1%)
FRiRfIVRfI	6 out of 48	
FCRiRfII	4 out of 48	
FCRiRfIVRfI	38 out of 48	

Table 6-8: Results of the Best Kalman Single Processing Techniques.

Summary of Single Sensor Approach			
Approach	Raw Process Techniques	Moving Average Processing Techniques	Kalman Single Processing Techniques
Best Technique	CRiRfIVRfI	11MCRiRfIVRfI	FCRiRfIVRfI
Average	13,500 m	8,330 m	8,780 m
Standard Deviation	12,500 m	13,300 m	14,700 m

Table 6-9: Summary of the Single Sensor Approach (All Experiments are 1800 Seconds Duration).

Table 6.9 presents a summary of the different approaches: (i) The Raw Processing Technique, (ii) The Moving Average Processing Technique (iii) The Kalman Single Processing Technique. The best technique in each approach was shown. In addition, the average displacement estimate and the standard deviation of best technique in each approach were presented.

6.2 Multi-Sensor Approach

The Multi-sensor approach results are presented in this section. Table 6.10 presents Multi-Sensor Processing Techniques of 2 experiments out of 12 experiments. In each experiments, 4 measured acceleration signals are combined and processed in four different techniques: (i) AFCRiRfIVRfI, (ii) FACRiRfIVRfI, (iii) FCRiRfIAVVRfI, (iv) FCRiRfIVRfIAD. The letter codes for these notations are presented in Table 5.1.

Experiment #	Measured Acceleration Signals	Multi-Sensor Processing Techniques			
		AFCRiRfIVRfI	FACRiRfIVRfI	FCRiRfIAVVRfI	FCRiRfIVRfIAD
1	Accel. 1	15,900	6,620	1,370	1,370
	Accel. 2				
	Accel. 3				
	Accel. 4				
2	Accel. 1	4,300	2,110	-197	-197
	Accel. 2				
	Accel. 3				
	Accel. 4				

Table 6-10: Results of the Multi-Sensor Processing Techniques (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).

Experiment #	Measured Acceleration Signals	Kalman Multi-Sensor Processing Techniques		
		PFCRiRfIVRfI	SFCRiRfIVRfI	CFCRiRfIVRfI
1	Accelerometer 1	-765	-765	-765
	Accelerometer 1			
	Accelerometer 1			
	Accelerometer 1			
2	Accelerometer 1	-2,340	-2,340	-2,340
	Accelerometer 1			
	Accelerometer 1			
	Accelerometer 1			

Table 6-11: Results of the Kalman Multi-Sensor Processing Techniques (All Values in Units of Meters) (All Experiments are 1800 Seconds Duration).

Table 6.11 presents the Kalman Multi-Sensor Processing Techniques of 2 experiments out of 12 experiments. In each experiments, 4 measured acceleration signals are combined and processed in four different techniques using the Kalman filter: (i) PFCRiRfIVRfI, (ii) SFCRiRfIVRfI, (iii) CFCRiRfIVRfI. The letter codes for these notations are presented in Table 5.1.

Summary of Multi-Sensor Approach		
Approach	Multi-Sensor Processing Techniques	Kalman Multi-Sensor Processing Techniques
Best Technique	VFCRiRfIRfI = DFCRiRfIRfI	PFCRiRfIRfI = SFCRiRfIRfI = CFCRiRfIRfI
Maximum	23,500 meter	10,500 meter
Minimum	197 meter	13 meter
Average of the experiments	6,600 meter	3,140 meter
Standard Deviation	7,450 meter	2,950 meter

Table 6-12: Summary of Multi-Sensor Approach (All Experiments are 1800 Seconds Duration).

Finally, Table 6.12 presents a summary of the Multi-Sensor different approaches: (i) The Multi-Sensor Processing Techniques, (ii) The Kalman Multi-Sensor Processing Techniques. The best technique in each approach was shown. In addition, the maximum and minimum estimated displacement, the average estimated displacement of the 12 experiments and the standard deviation of the best technique in each approach were presented.

7 Discussion

The discussion of results is split into two main parts. First, discussion on the use of a “single sensor” for the proposed RPES, and second a discussion on the use of “multiple redundant sensors” for the RPES. Also, some comments on the effect of temperature variation on the results, and the effects of test duration on the results.

7.1 Single Sensor Approach

For the “single sensor” three approaches were investigated to estimate displacement including: (i) use of boundary conditions alone, (ii) use of moving average techniques & boundary conditions, and (iii) use of Kalman Filtering techniques & boundary conditions. With all these techniques, double integration was performed to create an estimate for the total displacement.

7.1.1 Use of Boundary Conditions

First, considerations of various ways to apply the boundary conditions (as described in Table 1.1) to the collected accelerometer data were considered. Four processing techniques were investigated, which were: (i) RiRfII, (ii) RfRiIVRfI, (iii) CRiRfII, (iv) CRiRfIVRfI, where the meaning of the ‘letters’ for these processes is listed in Section 5.1.2. A comparison was conducted to: determine the best Raw Process technique developed, observe the effects of the boundary conditions, and observe the effect of removing the first 20 seconds of the output data. The comparisons were based upon the estimated displacement values of each raw process technique, for a single experiment.

Each of these four raw process techniques was applied 48 times for all the experiments. For each of these 48 trials, the raw process technique that had the closest value to zero displacement was noted. In other words, “was it raw process technique (i), (ii), (iii) or (iv) that was best?” for each of the 48 trials. Finally, the raw process technique that was closest most often out of 48 trials was considered to be the best raw process technique. Table 7.1 lists these results. As can be seen from the table, the raw process technique that had the closest estimate to the true value (i.e. zero) was CRiRfIVRfI, which was closest 28 times out of 48 experiments. The worst raw process technique was RiRfII, which only had the closest estimate to the true value, once out of 48 experiments.

The Percentage of the Best Technique out of the 4 Raw Processing Techniques		Best Raw Processing Technique
RiRfII	1 out of 48	CRiRfIVRfI (58.3%)
RiRfIVRfI	6 out of 48	
CRiRfII	13 out of 48	
CRiRfIVRfI	28 out of 48	

Table 7-1: Results of the Best Raw Processing Techniques.

Note that the best raw process technique CRiRfIVRfI, employs a C-process, which is a “cut” of the first 20 seconds of acceleration data.

- 1) The effect of removing the first 20 seconds had improved the displacement estimate, in comparison to the raw process techniques without removing the first 20 seconds.
- 2) Additionally, there was a difference between the raw process techniques in the last two rows of Table 7.1. The raw process technique of row 3 used only two boundary conditions; Ri and Rf, while the raw process technique of row 4 used three, Ri, Rf, and VRf. The first and second boundary conditions had removed

the offset error, as discussed previously. However, the effect of the third boundary condition was unknown, because there was no relative relationship between the acceleration and the displacement. In other words, if you were to change a parameter in the acceleration output values, one can predict the change in the velocity output values, but not the change in the displacement output values. Therefore, the third boundary condition would lead to either better or worse results. Based on the result of the 48 experiments, it was found that the third boundary conditions had improved the displacement readings significantly.

7.1.2 Use of Moving Averages & Boundary Conditions

Next, consideration of various ways to use both moving averages and boundary conditions was considered. Four moving average processing techniques were investigated, which were: (i) 11MRiRfII, (ii) 11MRiRfIVRfI, (iii) 11MCRiRfII, (iv) 11MCRiRfIVRfI, where 11M means the Moving Average Filter with 11-Sample Size, 21M means the Moving Average Filter with 21-Sample Size, and 41M means the Moving Average Filter with 41-Sample Size.

This comparison was done to determine if any improvements to the displacement estimate could be made by applying the moving average filter with 11-sample size, 21-sample size and 41-sample size. Also, in each sample size, another comparison was conducted to see the effect of removing the first 20 seconds of acceleration data, and applying the third boundary condition with the moving average filter.

- 1) A comparison between the moving average filters with 11-sample size with other techniques. In this comparison, the same comparison as before was conducted,

however, each technique was filtered out using the moving average with 11-sample size and the results were as following:

The Percentage of the Best Technique out of the 4 Moving Average filter with 11-Sample Size		The Best Moving Average 11-Sample Size Processing Technique
11MRiRfII	1 out of 48	11MCRiRfIVRfI (85.42%)
11MRiRfIVRfI	1 out of 48	
11MCRiRfII	5 out of 48	
11MCRiRfIVRfI	41 out of 48	

Table 7-2: Comparison between the Moving Average Processing Techniques with 11-Sample Size

Table 7.2 shows that the most values that were closest to the true value were the values when applying the Moving average filter with 11-sample size, removing the first 20 seconds and applying the three boundary conditions with 41 values. The numbers of values that were closest to the true values after applying the moving average filter with 11-sample size and removing the first 20 seconds but without applying the third boundary condition were 5 values. 1 value that was closest to the true value in both moving average process techniques: 1) When applying the moving average filter with 11-sample size and the third boundary condition and without removing the first 20 seconds. 2) When applying the moving average filter with 11-sample size and without the third boundary condition and without removing the first 20 seconds.

The Percentage of the Best Technique out of the 4 Moving Average filter with 21-Sample Size		The Best Moving Average 21-Sample Size Processing Technique
21MRiRfII	0 out of 48	21MCRiRfIVRfI (100.00%)
21MRiRfIVRfI	0 out of 48	
21MCRiRfII	0 out of 48	
21MCRiRfIVRfI	48 out of 48	

Table 7-3: Comparison between the Moving Average Processing Techniques with 21-Sample Size

Table 7.3 shows a similar comparison as the previous one but this time, the moving average filter with 21-sample size was applied. In all the 48 experiments, the values closest to the true values were for the moving average process technique when applying the moving average filter with 21-sample size, applying the third boundary condition and removing the first 20 seconds of the readings. While other moving average process techniques, none of the values were closest to the true value in any experiment.

The Percentage of the Best Technique out of the 4 Moving Average filter with 41-Sample Size		The Best Moving Average 41-Sample Size Processing Technique
41MRiRfII	0 out of 48	41MCRiRfIVRfI (95.83%)
41MRiRfIVRfI	1 out of 48	
41MCRiRfII	1 out of 48	
41MCRiRfIVRfI	46 out of 48	

Table 7-4: Comparison between the Moving Average Processing Techniques with 41-Sample Size

Table 7.4 shows a third comparison conducted when applying the moving average filter with 41-sample size. The table illustrated that the when applying the moving average filter with 41-sample size, removing the first 20 seconds and applying the third boundary condition, the number of values that represent the closest values to the true value were 46 out of 48 experiments. If the first 20 seconds were not removed and the third boundary condition was not applied 1 value out of the 48 experiments was obtained to be the closest value to the true value. And no values were obtained to be the closest value to the true value when applying the moving average filter with 41-sample sizes, not removing the first 20 seconds and not applying the third boundary condition.

After conducted these comparisons, the best moving average process techniques from each comparison were obtained and, therefore, they were to be compared with each other to acquire the best moving average filter among the three sample sizes.

The Percentage of the Best Technique out of the 3 Moving Average filter with the Different Sample Sizes		The Best Moving Average Technique
11MCRiRfIVRfl	41 out of 48	11MCRiRfIVRfl (85.42%)
21MCRiRfIVRfl	6 out of 48	
41MCRiRfIVRfl	1 out of 48	

Table 7-5: The Comparison between the Moving Average with Different Sample Sizes

Table 7.5 shows that the best moving average process techniques from the previous comparisons were when removing the first 20 seconds, and applying the third boundary condition for all moving average with different sample sizes. After another comparison was done amongst the different moving average with different sample size, it was found out that with 11-sample size, 41 values were the closet to the true value for the 48 experiments, 6 values when using the moving average with 21-sample size, and only 1 value when applying moving average filter with 41-sample size.

7.1.3 Use of Kalman Filter & Boundary Conditions

Next, consideration of various ways to use both a Kalman Filter and the boundary conditions was considered. Four Kalman single processing techniques were investigated, which were: (i) FRiRfII, (ii) FRiRfIVRfl, (iii) FCRiRfII, (iv) FCRiRfIVRfl, where the F-process represents application of the Kalman Filter, as described in Section 5.1.1.

The Percentage of the Best Technique out of the 4 Kalman Single Processing Techniques		The Best Kalman Single Processing Techniques
FRiRfII	0 out of 48	FCRiRfIVRfI (79.17%)
FRiRfIVRfI	5 out of 48	
FCRiRfII	5 out of 48	
FCRiRfIVRfI	38 out of 48	

Table 7-6: Comparison between the Kalman Single Processing Techniques

Table 7.6 shows that the most values that were closest to the true value were the Kalman single process technique of row 4. It used the Kalman filter; F, the three boundary conditions, Ri, Rf, and VRf and it employed the C-process, which is a ‘cut’ of the first 20 seconds of acceleration data. The numbers of values that were closest to the true values were 38 values. 5 values that was closest to the true value in both techniques, the Kalman single process technique of rows 2 and 3: 1) When applying the Kalman filter, F, and the third boundary condition, VRf, and without removing the first 20 seconds, C. 2) When applying the Kalman filter, F, and without the third boundary condition, VRf, and when removing the first 20 seconds, C.

7.1.4 Comparison of All Single Sensor Processing Methods

Finally, a comparison between all of the above methods was conducted. This was done by computing the average and the standard deviation of the 48 displacement estimates for all the process techniques, for the three cases of Section 7.1.1, 7.1.2 and 7.1.3, This included processing with raw acceleration data, processing with the moving average filter with 11, 21, and 41 sample sizes, and processing with the Kalman filter.

Summary of Single Sensor Approach			
Approach	Raw Process Techniques	Moving Average Processing Techniques	Kalman Single Processing Techniques
Best Technique	CRiRfIRfI	11MCRiRfIRfI	FCRiRfIRfI
Average	13,500 m	8,330 m	8,780 m
Standard Deviation	12,500 m	13,300 m	14,700 m

Table 7-7: Summary of the Single Sensor Approach (All Experiments are 1800 Seconds Duration).

Table 7.7 lists the average displacement and the standard deviation of the 48 experiments, for all the processing techniques. First, the best processing technique for each category had to be determined. This information is listed in Tables 7.1 through to 7.6. Therefore, for the raw process technique, the best technique was (CRiRfIVRfI), where the first 20 seconds of the data were removed and all three boundary conditions were applied. For the moving average process technique, the best technique was (11MCRiRfIVRfI), where the first 20 seconds of the data were removed, all three boundary conditions were applied, and the 11-sample size was used. When applying the Kalman filter, the best Kalman single process technique was FCRiRfIVRfI, where the first 20 seconds of the data was removed, and all three boundary conditions were applied. The average value of the 48 readings for the (CRiRfIVRfI) technique was about 13.5 kilometer as opposed to about 8.3 kilometers for the (11MCRiRfIRfI) and 8.78 kilometer for the (FCRiRfIVRfI) technique. In addition, the standard deviation of the (CRiRfIVRfI) technique was about 12.5 kilometer, 13.3 kilometer for the (11MCRiRfIVRfI) technique, and 14.66 kilometer for the (FCRiRfIVRfI) technique. It turned out that the moving average process technique with 11-sample gave the best average displacement, followed by the Kalman single process technique with no much difference, and finally the raw process technique with too much difference. On the other hand, the raw process

technique gave the best standard deviation followed by the moving average process technique with 11-sample size, then the Kalman single process technique. The average value of the raw process technique was anticipated to be the worse, however, the moving average process technique was better than the Kalman process technique because that by getting back to graph (51) it was observed that the moving average filter with 11-sample size's readings was closer to the zero – the true value – even though, it wasn't following the original data as the Kalman filter was.

Then the standard deviation of the raw process technique was less than when applying the moving average or Kalman filter, which illustrated the average deviation of the displacement readings. Then the moving average process technique came next and finally the Kalman filter process technique. The reason could be because in some raw displacement values, they changed between negative and positive values, which cancelled each other out and, in turn, reduced the deviation values.

7.2 Single Sensor Short-Duration Analysis

The previous section describes the performance of the single approach over a 30-minute duration (i.e. 1800 seconds). The reason for doing this is to allow us to better see the error and identify the drift over a long duration, and thereby develop a system that can handle all three types of errors. However, for the proposed application of surveying or construction use, as described in Section 1.1, a shorter duration of approximately 1 minute is more likely. Hence the purpose of this section is to show the performance of the system over a short-duration. Using the same data from Section 7.1, we will only consider the first 60 seconds.

Technique	F.Ri.Rf.I.VRi.I.	
Experiment Duration	60 Seconds Duration	1800 Seconds Duration
Average Displacement	33.8 m	35,700 m

Table 7-8: Comparison Between the Short-Duration and the Long-Duration Experiment

Table 7.8 shows an average displacement comparison of the technique F.Ri.Rf.I.VRi.I. between the short-duration experiment (60 seconds) and the long-duration experiment (1800 seconds). The average displacement of the 60-second-duration experiment is 33.8 meter as opposed to 35,700 meters in the 1800-second-duration experiment. To demonstrate the problem associated with long-term drift, we will calculate the average displacement after 60 seconds, from the 1800-second experiment. This can be done by dividing the 35,700 m average displacement by 1800 seconds, and then multiplying by 60 seconds, which is equal to 1190 meters. Therefore, when comparing the two results, we find that the short-duration experiment produces 97% better performance than the long-duration experiment, due to the fact there is less drift.

7.3 Multiple Sensors Approach

For the “multiple sensors approach,” four different measurement signals are combined together to construct a single signal for 1 experiment. Two approaches were investigated to combine the measurement signals and estimate the displacement, which include: (i) use of standard averaging techniques, and (ii) use of Kalman Filtering to combine the measurement signals. With all these techniques, double integration was performed to create an estimate for the total displacement. The total number of experiments is 12 experiments and in each experiment, there are four measurement acceleration signals.

7.3.1 Use of Standard Averaging Techniques

In this section different process techniques results were presented, in which the processing techniques was to combine the measurement signals using standard averaging techniques to get the best displacement estimate.

Table 7.8 presents the different multi-sensor processing techniques: (i) AFCRiRfVVRfI, (ii) FACRiRfIVVRfI, (iii) FCRiRfIAVVRfI, (iv) FCRiRfIVVRfIAD. The notations are defined in Table 5.1. In this table, a comparison between these four multi-sensor processing techniques was conducted by comparing the average displacement estimate and the standard deviation. As seen in the table, the multi-sensor processing technique in the second column had the highest average displacement (18,700 m) and the highest standard deviation (16,400 m), whereas, the multi-sensor processing technique in the first column had the second highest average displacement (15,400 m) and standard deviation (16,000 m). The multi-sensor processing technique in the third and fourth columns were exactly the same and had the lowest and the best average displacement estimate (6,600 m) and standard deviation (7,450 m). The average displacement and standard deviation were the same when taking the velocity average and the displacement average because of the linear relationship between the velocity and the displacement.

Multi-Sensor Processing Techniques			
AFCRiRfVVRfI	FACRiRfIVVRfI	FCRiRfIAVVRfI	FCRiRfIVVRfIAD
Average displacement:	Average displacement:	Average displacement:	Average displacement:
15,400 m	18,700 m	6,600 m	6,600 m
Standard deviation: 16,000 m	Standard deviation: 16,400 meter	Standard deviation: 7,450 meter	Standard deviation: 7,450 meter

Table 7-9: Comparison between the Multi-Sensor Processing Techniques (All Experiments are 1800 Seconds Duration).

7.3.2 Use of Kalman Multi-Sensor Processing Techniques

In this section different process techniques results were presented. The Kalman filter was designed to process a number of measurement signal to get a single output. These designs are: (i) The parallel design, (ii) The series design, (iii) The compression design. The objective is to optimally reduce the error covariance and get best displacement estimate.

Table 7.9 presents the different Kalman multi-sensor processing techniques: (i) PFCRiRfIRfI, (ii) SFCRiRfIRfI, (iii) CFCRiRfIRfI. The definition of the letter code is found in Table 5.1. In this table, a comparison between the three Kalman multi-sensor processing techniques was conducted by comparing the average displacement estimate and the standard deviation. As seen in the table, the Kalman multi-sensor processing techniques in the three columns are equal, which the average displacement is equal to 3,140 m, and the standard deviation equals 2,950 m.

Kalman Multi-Sensor Processing Techniques		
PFCRiRfIRfI	SFCRiRfIRfI	CFCRiRfIRfI
Average displacement:	Average displacement:	Average displacement:
3,140 m	3,140 m	3,140 m
Standard deviation	Standard deviation	Standard deviation
2,950 m	2,950 m	2,950 m

Table 7-10: Comparison between the Kalman Multi-Sensor Processing Techniques (All Experiments are 1800 Seconds Duration).

Summary of Multi-Sensor Approach		
Approach	Multi-Sensor Processing Techniques	Kalman Multi-Sensor Processing Techniques
Best Technique	VFCRiRfIRfI = DFCRiRfIRfI	PFCRiRfIRfI = SFCRiRfIRfI = CFCRiRfIRfI
Maximum	23,500 meter	10,500 meter
Minimum	197 meter	13 meter
Average of the experiments	6,600 meter	3,140 meter
Standard Deviation	7,450 meter	2,950 meter

Table 7-11: Summary of Multi-Sensor Approach (All Experiments are 1800 Seconds Duration).

Table 7.10 shows different comparisons between the two approaches: (i) Multi-Sensor Processing Techniques, (ii) Kalman Multi-Sensor Processing Techniques, where the best techniques of the two approaches were presented:

- 1) The maximum displacement estimate value of the best multi-sensor processing technique was 23,500 m. On the other hand, the maximum displacement estimate value of the best Kalman multi-sensor processing technique was 10,500 m. That is about 55% improvement.
- 2) The minimum displacement estimate value of the best multi-sensor processing technique was 197 m. On the other hand, the minimum displacement estimate value of the best Kalman multi-sensor processing technique was 13 m. That is about 93% improvement.
- 3) The average displacement estimate value of the best multi-sensor processing technique was 6,600 m. Likewise; the average displacement estimate value of the best Kalman multi-sensor processing technique was 3,140 m. That is an improvement of 52%.
- 4) The standard deviation estimate value of the best multi-sensor processing technique was 7,450 m. Similarly; the standard deviation estimate value of the best Kalman multi-sensor processing technique was 2,950 m. That is an improvement of 60%.

7.4 Other Observations

- I. The effect of ambient temperature on the accelerometers' measurement output was investigated. The effect of temperature varies from one accelerometer brand to another, depending on the accelerometer's specifications and characteristics, and can be found on the accelerometer's data sheet.

Fig. A. 5 in Appendix-A shows the relationship between the temperature and the accelerometer's output voltage for the ADXL203 accelerometer, which is the one used in these experiments. It is noted that within a temperature range of +5 and -5 degrees of the room temperature (25°C), which is within 20°C to 30°C, the corresponding voltage output was about the same, where the relationship was approximately a flat line. The output voltage changes from about 2.502 volts to about 2.482 volts, over the temperature range of -40°C to 125°C.

- II. Fig. A.6 in Appendix-A shows the relationship between the accelerometer's sensitivity and the temperature. It shows that for temperatures between 20°C and 30°C, there is no influence on the accelerometer's sensitivity and, therefore, the temperature effect can be neglected for the experiments of this work.

8 Contributions & Future Work

8.1 Contributions

The main contributions of this work are as follows:

- 1) The proposed RPES application, and specifically the proposed five boundary conditions of Table 1.1 that allow for improved displacement estimations, in comparison to applications without boundary conditions.
- 2) Investigation of various signal processing techniques which combine the boundary conditions with moving averages or Kalman filtering for the single accelerometer.
- 3) Investigation of combining multiple, redundant accelerometer output using averaging or Kalman Filtering, to create an improved displacement estimate.

8.2 Future Work

In order to further develop the proposed RPES device, some additional work can be investigated in the following areas:

- 1) The device hardware can be upgraded to use more sensitive components, in order to improve the performance and to reduce the noise. This includes:
 - a. Upgrading the A/D resolution. The microcontroller AT90USBKey had a 10-bit analog-to-digital resolution. This could be improved by using an A/D with a higher resolution, such as: the MCP3221 with 12-bit [39] or the ADADC71 with 16-bit resolution [40].

- b. Use a microcontroller with more than four A/D converter channels, which is the limit of the existing microcontroller. The existing system only has four A/D channels. A system with many high-resolution, A/D converters can be used to investigate the performance of five or more redundant accelerometers, on the estimated displacement.
 - c. The use of high performance inertial sensors with higher sensitivity and lower white Gaussian noise. This should greatly reduce the noise and improve the estimations of displacement.
 - d. Using multiple identical sensors would reduce the drift and offset errors, however, since they have the same white noise, perhaps, the improvement will be limited. Therefore, using non-identical sensors with different white noise errors would, perhaps, reduce the white noise, i.e. different manufacturer or different model numbers.
- 2) Further investigation is required to test the RPES with variable acceleration vs. time values. In other words, where the relationship between the acceleration and time is non-linear. Since Kalman filters can only be applied to linear system equations, new Kalman filter must be developed to handle non-linear systems by linearizing the mean and covariance, which is called the extended Kalman filter or EKF [37].
- 3) In the practical application of the proposed RPES device, the true acceleration is completely unknown, and the RPES device is expected to figure out/estimate that displacement. However, one of the input parameters for the Kalman filter used in this work is an estimate of the true acceleration. However, this would not be

available if the device is used in practice. Hence, further investigation should be conducted on designing a Kalman filter with an unknown input. In this respect, Gillijns and Moor [41] had optimally developed a filter that simultaneously estimates the input and the state of a linear system, where the relation between the unknown input and the state are interconnected. Hsieh [42] has investigated the state estimation of linear time-varying discrete system with an unknown input, where he proposed a robust two stage Kalman filter, which is unaffected by the unknown input. In [43] Kitanidis has illustrated the use of the linear minimum-variance estimation to determine mean areal precipitation with unknown input. The precipitation is modeled as a stochastic system with unknown mean and time varying. The average of the space is estimated with minimum variance and unbiased with any value of the unknown mean.

Conclusion

The scope of this work was to investigate the use of MEMS inertial sensors for relative positioning estimation. The performance of a single accelerometer was examined with a set of static tests using different filtering techniques. This included the investigation of: applying boundary conditions specific to this application, applying moving average filters, and applying a Kalman filter. These boundary conditions were to zero the first, last values of the acceleration and displacement values. After applying the boundary conditions and the different processing techniques, comparisons were done. It was found that application of the boundary conditions alone significantly improved the estimated displacement, in comparison to not using them. After conducting single sensor experiment 48 times, it was found that the performance could be further increased by the use of moving average filter. The use of applying a Kalman filter was also evaluated, which gave a good prediction to the original data with reduced white noise. However, after the analysis of all three single-sensor processing techniques, it was determined that the moving average filter with 11-sample size provided the best estimate of displacement.

The performance of redundant, multi-sensor measurement was also investigated. The investigation considered two main approaches. Firstly, simple averaging was performed at various stages of the processing, and it was found that the estimated displacement averages were identical for all cases. Secondly, three different designs of multi-sensor Kalman filters were implemented, which were: parallel, series and compression designs. They all gave the same results. When comparing simple averaging of multi-sensors with multi-sensor Kalman Filters, it was found the Kalman Filters had significantly increased performance.

The effect of the temperature was also investigated and it was observed that it had little effect over a short range of temperature difference for these particular accelerometers and, therefore, the temperature effect was ignored. Due to the accumulation of error over time when using inertial sensors as positioning measurement system, the best performance for such a device is for short period of time [15].

To conclude, a number of different processing techniques were attempted with single-sensor and multi-sensor configurations. Overall, it was found that the multi-sensor Kalman Filter combined with the use of boundary conditions, provided the best estimate of displacement, with results that were 2.6 times better than the best single-sensor estimates, and over 127 times better than double integration of the raw accelerometer data without boundary conditions. Although these increases in displacement estimates are good, improvements of at least an additional 10 times are required to meet the needs of the proposed RPES application. This may be achieved by the implementation of some of the recommendations made in this report.

Bibliography

- [1] "The Global Positioning System." Internet: <http://www.my-car-computer.com/global-positioning-system.html>, Jan. 01, 2011 [June 08, 2011].
- [2] Wikipedia. "Positioning system." Internet: http://en.wikipedia.org/wiki/Positioning_system, April 19, 2011 [May 09, 2011].
- [3] R.G. Brown and P.Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 3rd Edition, Wiley, 1996, pp. 496.
- [4] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech Print on Demand, 2004, pp. 299.
- [5] Wikipedia. "The Global Positioning System." Internet: http://en.wikipedia.org/wiki/Global_Positioning_System, July 26, 2011 [Feb. 20, 2011].
- [6] L. Greenemeler, "A Positioning System That Goes Where GPS Can't: Scientific American." Internet: <http://www.scientificamerican.com/article.cfm?id=indoor-positioning-system>, Jan. 23, 2008 [April 13, 2011].
- [7] C. Randell and H. Muller, "Low Cost Indoor Positioning System," *UbiComp'01*, 2001, pp. 42-28.
- [8] O.J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, 2007.
- [9] S.I. Roumeliotis, "Extended Kalman filter for frequent local and infrequent global sensor data fusion," *Proceedings of SPIE*, Pittsburgh, PA, USA: 1997, pp. 11-22.
- [10] C. Cappelle, D. Pomorski, and Y. Yang, "GPS/INS Data Fusion for Land Vehicle Localization," *The Proceedings of the Multiconference on Computational Engineering in Systems Applications*, Beijing, China: 2006, pp. 21-27.
- [11] D. Berdjag and D. Pomorski, "DGPS/INS data fusion for land navigation," In *Proceedings of the Seventh International Conference on Information Fusion 21*, 2004, pp. 881-887.
- [12] L. Drolet, F. Michaud, and J. Cote, "Adaptable sensor fusion using multiple Kalman filters," *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)* (Cat. No.00CH37113), Takamatsu, Japan, pp. 1434-1439.
- [13] A. Soloviev, "Tight Coupling of GPS and INS for Urban Navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, Oct. 2010, pp. 1731-1746.
- [14] A. Soloviev, D. Bates, and F. van Graas, "Tight Coupling of Laser Scanner and Inertial Measurements for a Fully Autonomous Relative Navigation Solution," *INSTITUTE OF NAVIGATION*, vol. 54, 2007, pp. 189-206.
- [15] G. Pang and H. Liu, "Evaluation of a Low-cost MEMS Accelerometer for Distance Measurement," *Journal of Intelligent and Robotic Systems*, vol. 30, Mar. 2009, pp. 249-265.
- [16] C. Lemaire, "Surface micromachined sensors for vehicle and personal navigation systems," *Proceedings of Conference on Intelligent Transportation Systems*, Boston, MA, USA , pp. 1068-1072.

- [17] H.H.S. Liu and G.K.H. Pang, "Accelerometer for mobile robot positioning," *IEEE Transactions on Industry Applications*, vol. 37, Jun. 2001, pp. 812-819.
- [18] Y.K. Thong, M.S. Woolfson, J.A. Crowe, B.R. Hayes-Gill, and R.E. Challis, "Dependence of inertial measurements of distance on accelerometer noise," *Measurement Science and Technology*, vol. 13, Aug. 2002, pp. 1163-1172.
- [19] L.A. Rocha, E. Cretu, and R.F. Wolffenbuttel, "Measuring and interpreting the mechanical-thermal noise spectrum in a MEMS," *Journal of Micromechanics and Microengineering*, vol. 15, Jul. 2005, p. S30-S38.
- [20] D.S. Bayard and S.R. Ploen, "High accuracy inertial sensors from inexpensive components," U.S. Patent US 6882964 B2, October 2, 2003.
- [21] K. Siercks, "Hand-Held Surveying Device and Surveying Method for Such a Surveying Device" U.S. Patent 2589998, June 15, 2006.
- [22] Wikipedia. "Trilateration." Internet: <http://en.wikipedia.org/wiki/Trilateration>, July 19, 2011 [Mar. 10, 2011].
- [23] B. Levitan and L. Harte, *GPS Quick Course 2nd Edition, Systems, Technology and Operation*, Fuquay-Varina N.C.: Althos Pub., 2009.
- [24] R.P. Feynman, "There's Plenty of Room at the Bottom. An Invitation to Enter a New Field of Physics," *Caltech Engineering and Science*, vol. 23:5, Feb. 1960, pp. 22-36.
- [25] H.C. Nathanson, W.E. Newell, R.A. Wickstrom, and J.R. Davis, "The resonant gate transistor," *IEEE Transactions on Electron Devices*, vol. 14, Mar. 1967, pp. 117-133.
- [26] Long-Sheng Fan, Yu-Chong Tai, and R.S. Muller, "IC-processed electrostatic micro-motors," *Technical Digest., International Electron Devices Meeting*, San Francisco, CA, USA: , pp. 666-669.
- [27] D.R. Sparks, "Application of MEMS technology in automotive sensors and actuators," *MHA '98. Proceedings of the 1998 International Symposium on Micromechatronics and Human Science. - Creation of New Industry - (Cat. No.98TH8388)*, Nagoya, Japan: , pp. 9-15.
- [28] C. Liu, *Foundations of MEMS*. New Jersey: Prentice Hall, 2005, pp. 530.
- [29] P.D. Groves, "Inertial Sensors," *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2008, pp. 97-120.
- [30] J.P. Lynch, A. Partridge, K.H. Law, T.W. Kenny, A.S. Kiremidjian, and E. Carryer, "Design of Piezoresistive MEMS-Based Accelerometer for Integration with Wireless Sensing Unit for Structural Monitoring," *Journal of Aerospace Engineering*, vol. 16, 2003, p. 108.
- [31] R. Christensen and N. Fogh, "Inertial Navigation System." Master project, Aalborg University, 2008.
- [32] A.J. Wheeler, A.R. Ganji, and V.V. (CON) Krishnan, *Introduction to Engineering Experimentation*, Prentice Hall, Denmark, 2009.
- [33] G. Vijayaraghavan, M. Brown, and M. Barnes, "Electrical noise and mitigation - Part 1: Noise definition, categories and measurement." Internet: <http://www.eetimes.com/design/industrial-control/4014226/Electrical-noise-and-mitigation--Part-1-Noise-definition-categories-and-measurement?pageNumber=0>, Dec. 16, 2008 [Jan. 04, 2011].

- [34] M.H. Rashid, *Microelectronic Circuits: Analysis and Design*, CL-Engineering, 1998.
- [35] P.S. Maybeck, "Introduction," in *Stochastic Models, Estimation, and Control*, Academic Press, 1979.
- [36] D. Simon, "Kalman Filtering," *Embedded Systems Programming*, vol. 14, Jun. 2001, pp. 72-79.
- [37] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *Department of Computer Science University of North Carolina at Chapel Hill*, Jul. 2006.
- [38] D. Willner, C. Chang, and K. Dunn, "Kalman filter algorithms for a multi-sensor system," *1976 IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes*, 1976, pp. 570-574.
- [39] Texas Instruments, "Low-Power 12-Bit Analog-To-Digital Converter with 12C Interface," Sep. 2009, p. 20.
- [40] Analog Devices, "Complete, High Resolution 16-Bit A/D Converter," 2005, p. 12.
- [41] S. Gillijns and B. Demoor, "Unbiased minimum-variance input and state estimation for linear discrete-time systems with direct feedthrough," *Automatica*, vol. 43, May. 2007, pp. 934-937.
- [42] Chien-Shu Hsieh, "Robust two-stage Kalman filters for systems with unknown inputs," *IEEE Transactions on Automatic Control*, vol. 45, Dec. 2000, pp. 2374-2378.
- [43] P. Kitanidis, "Unbiased minimum-variance linear state estimation," *Automatica*, vol. 23, Nov. 1987, pp. 775-778.

Appendix-A (RPES Components Specifications)

The Atmel AT90USBKey kit was used, which contains [44]:

- 128KB Flash Program Memory
- 8KB SRAM
- 4KB EEPROM
- 8 Channel 10-bit A/D-converter
- JTAG interface for on-chip-debug
- Up to 16MIPS throughput at 16MHz
- 2.7 – 5.5 Volt operation
- USB full/low speed device/host/OTG interface
- 4+1-ways joystick
- 2 Bi-Color LEDs
- Temperature sensor
- System clock: 8 MHz crystal
- Onboard reset button
- Onboard HWD button (force bootloader section execution)
- AT90USB1287 microcontroller

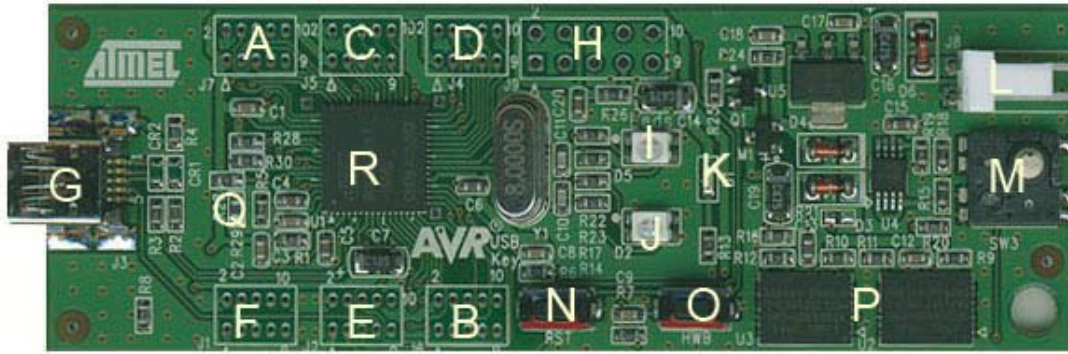


Figure A-1: AT90USBKey [45]

Label	Description
A-F	Ports A through F
G	USB MiniAB Connector
H	JTAG port
I-J	Bi-colored LEDs (Red/Green LEDs)
K	Power LED
L	External power connector
M	4+1 way joystick (four directions + push button)
N	On board reset button
O	On board HWB button to force bootloader section execution on reset
P	Data flash memory
Q	Temperature sensor
R	AT90USB1287

Table A-1: The AT90USBKey Components and Description [45]



Figure A-2: The Bluetooth Module (left) and Dongle (right) [46][47]

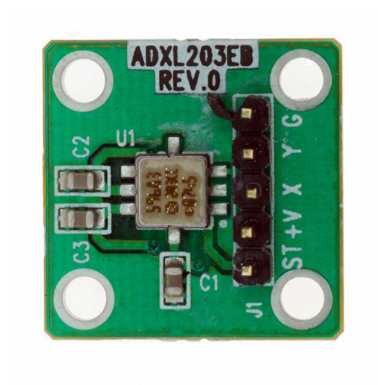


Figure A-3: The Dual Axis ADXL203 Accelerometer [“ADXL203EB Eval Bd $\pm 1.5g$ Dual Axis ADXL203 Accelerometer | eBay.”]

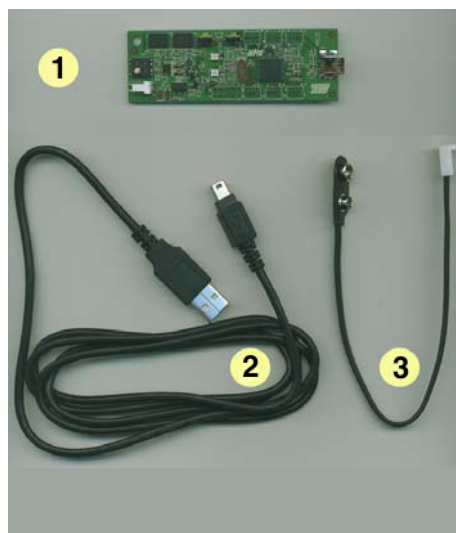


Figure A-4: The Connection Components of AT90USBKey [45]

The Components Description	The Specifications
• AT90USBKey micro-controller	• 8 MHz internal crystal oscillator clock, which can be operated on the following frequencies: 2 MHz, 8 MHz, 12 MHz, and 16 MHz • The clock is capable of 8-bit and 16-bit time resolution • 10-bit analog to digital resolution • 65-260 μm • Eight multiplexed single ended input channels • Optional left adjustment for ADC result readout • Free running or single conversion mode
• RN-41 Bluetooth module	• Roving networks • 1200 pbs baud rate speed • 100 meter rang • 2480 MHz frequency • Secure communications • 128-bit encryption
• Battery	• 9-Volt battery
• DBT-120 wireless Bluetooth 2.0 USB adaptor	• 2.1 Mbps signal rates • 4 dBm transmit power • 2.1 GHz frequency range • Up to 10 meter operating range
• Dual axis accelerometer evaluation ADXL203EB, which has the ADXL203 accelerometers	• High precision • Low power • Complete single and dual-axis • Full-scale range of $\pm 1.7 \text{ g}$ • Typical noise floor of $178 \mu\text{g}/\sqrt{\text{Hz}}$

Table A-2: The RPES Components Description

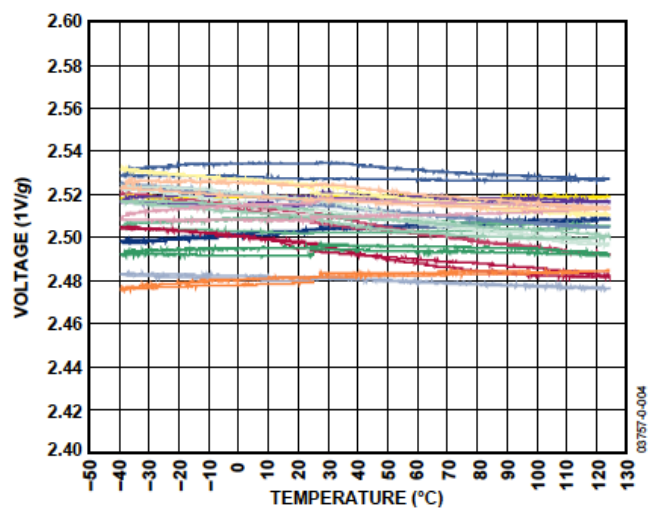


Figure A-5: The Voltage Vs. Temperature Graph [48]

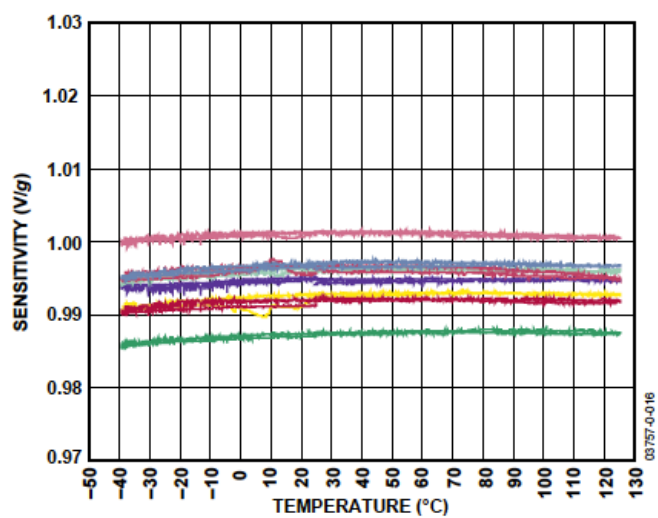


Figure A-6: The Sensitivity Vs. Temperature Graph [48]

Appendix-B (C-Code for the Micro-Controller AT90USB1287)

```

#include <avr/interrupt.h>
#include <avr/io.h>

// Global Variable
volatile char STATE;

void transmit_data(char char_out){
    while((UCSR1A & 0b00100000)
    != 0b00100000){}; //wait for empty transmit buffer

    //when UDRE1 (USART data register empty)
    is zero, make an endless loop until UDRE1 become 1
    UDR1 = char_out; //put data into
    buffer, sends the data

    return;
}

void digitalData(){

    ADMUX = 0b01100000; //ADC Multiplexer Selection
    Register

    //first 5 pins are set to 0 which means ADC0 PORTF (pin 1) is
    used as an input pin

    //the 6th pin is set to 0 to right adjust the result or getting the LSB
    of the value of ADC

    //pins 7 & 8 are set to 1 & 0 respectively for setting the reference
    votlage as an external voltage with 3.3v

    ADCSRA = 0b10000000; //the 8th pin is set to 1 to
    enable the ADC

    ADCSRA |= 0b01000000; //the 7th pin is set to 1 to
    start conversion

    while((ADCSRA & 0x40) == 0x40){}; //as long as pin 7
    is 1 that means the conversion is in progress

    //once it is 0 that means the conversion is completed

```



```

//ADC = Vin * 1023 (because ADC is 10-bit
register)/3.3 (Vref which was set earlier)
//convert ADC to binary and get the the first 8 most
significant bit MSB (ADCH)
char tmp = ADCL;
char junk = ADCH;

transmit_data(tmp);
transmit_data(junk);

ADMUX = 0b01100001; //first 5 pins are set to 1
which means ADC1 PORTF (pin 2) is used as an input pin

ADCSRA = 0b10000000; //the 8th pin is set to 1 to
enable the ADC

ADCSRA |= 0b01000000; //the 7th pin is set to 1 to
start conversion

while((ADCSRA & 0x40) == 0x40){}; //as long as pin 7
is 1 that means the conversion is in progress

//once it is 0 that means the conversion is completed

//ADC = Vin * 1023 (because ADC is 10-bit
register)/3.3 (Vref which was set earlier)
//convert ADC to binary and get the the first 8 most
significant bit MSB (ADCH)
char tmp1 = ADCL;
char junk1 = ADCH;

transmit_data(tmp1);
transmit_data(junk1);

ADMUX = 0b01100010; //first 5 pins are set to 2
which means ADC2 PORTF (pin 3) is used as an input pin

ADCSRA = 0b10000000; //the 8th pin is set to 1 to
enable the ADC

ADCSRA |= 0b01000000; //the 7th pin is set to 1 to
start conversion

while((ADCSRA & 0x40) == 0x40){}; //as long as pin 7
is 1 that means the conversion is in progress

```

```

//once it is 0 that means the conversion is completed

//ADC = Vin * 1023 (because ADC is 10-bit
register)/3.3 (Vref which was set earlier)
//convert ADC to binary and get the the first 8 most
significant bit MSB (ADCH)
char tmp2 = ADCL;
char junk2 = ADCH;

transmit_data(tmp2);
transmit_data(junk2);

return; //return the value of ADC
}

void mTimer(int count){
    /* Timer instructions */
    /* Sets timer 1 to run at CPU clock, disable all
function and use as pure timer */
    TCCR1B |= _BV(CS10);          /* _BV
sets the bit to logic 1)        */

/* Note the register is TCCR1B0 */
/*
TCCR1 is timer/counter control register 1 */
/* B
is the 'B' register and 0 is the 0th bit */
/*
CS is the pre-scaler */

int i=0;

TCCR1A = 0b00000010; //set the Timer/Conter
mode of operation to CTC

OCR1A = 1000; //set the output compare register for
1000 cycle
//system clock is 1 X 10 to the
power of -6
//==> 2000 (mTimer) X 1000
(OCR cycle) X 10 to the power of -6 = 2 seconds (for 2 LEDs)

```

```

//therefore 1 second for each
LED

TCNT1 = 0b00000000; //set the initial value of the
Timer/Counter 0x0000

//TIMSK1 = 0b00000010; //enable the output
compare interrupt

TIFR1 = 0b00000010; //clear the timer interrupt flag
and begin timer

while(i<count){ //poll the timer to determine when the
timer has reached 1000 cycle

    if((TIFR1 & 0x02) == 0x02
){

    TIFR1 = 0b00000010; //clear the interrupt flag by WRITING a ONE to
the bit

    i++; //increase the loop counter

    }

}

}

int main(){
    STATE = 0;

    cli(); // Disables all interrupts

    // Going to set up interrupt0 on PD0
    DDRD = 0b11111110;
    DDRF = 0b00000000; //set all pins of port F to input pins
                        //port F is used for analog to digital converter

    // Set up the Interrupt 0,3 options
    //External Interrupt Control Register A - EICRA (pg 94 and under the
EXT_INT tab to the right
    // Set Interrupt sense control to catch a rising edge
    EICRA |= 0b00000001;

```

```

//EICRA &= ~_BV(ISC01) & ~_BV(ISC00); /* These lines would undo the
above two lines */
//EICRA &= ~_BV(ISC31) & ~_BV(ISC30); /* Nice little trick */

```

```

// See page 96 - EIFR External Interrupt Flags...notice how they reset on
their own in 'C'...not in assembly
EIMSK |= 0b00000001;

```

```

// Enable all interrupts
sei(); // Note this sets the Global Enable for all interrupts

```

```

UCSR1A = 0b00000010; //set the register USART control and status
register A bit 1 to 1 (U2X1: double the USART transmission speed)
UCSR1B = 0b00001000; //set the register USART control and status
register B to zeros
UCSR1C = 0b00000110; //set the register USART control and status
register C bit 1 & 2 to 1 (UCSZ12 & UCSZ11: to make the character size = 8-bit)

```

```

UBRR1H = 0b00000000;
UBRR1L = 0b00001100; //baud rate (bps)=9600, U2X1=1 & clock
speed=1 MHz, hence, UBR=0b00001100=12

```

```

//char data_to_send; //decleare a variable to hold the digital value

```

```

PORTD = 0b10000000;

```

```

int x=0;

```

```

//mTimer(1000);
//}while(1);

do{
    if(STATE == 0)
        digitalData();

    mTimer(500);

    x++;
}while(x<=6000);

```

```

return(0);

```

```

}

```

```
/* Set up the External Interrupt 0 Vector */
ISR(INT0_vect){
    /* Toggle PORTD bit 0 */
    STATE = !STATE;
}
```

Appendix-C (MATLAB Code)

```

function kalman(duration, dt)

file = fopen('3600_13_Jan_11_4_3_2_1.txt', 'r');
data = fscanf(file, '%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d',
[16 inf]);

fclose(file);

data = data';

x1 = (data(:,2)*4)+(data(:,1)/64);
x2 = (data(:,6)*4)+(data(:,5)/64);
x3 = (data(:,10)*4)+(data(:,9)/64);
x4 = (data(:,14)*4)+(data(:,13)/64);

x1voltagearray = (x1*3.3)/1023;
x2voltagearray = (x2*3.3)/1023;
x3voltagearray = (x3*3.3)/1023;
x4voltagearray = (x4*3.3)/1023;

x1accel = [(9.80665*(x1voltagearray - 1.65))/0.626];
x2accel = [(9.80665*(x2voltagearray - 1.65))/0.626];
x3accel = [(9.80665*(x3voltagearray - 1.65))/0.626];
x4accel = [(9.80665*(x4voltagearray - 1.65))/0.626];

f = linspace(1/21, 1/21, 21);

m_x1accel2 = filter(f, 1, x1accel);
m_x2accel2 = filter(f, 1, x2accel);
m_x3accel2 = filter(f, 1, x3accel);
m_x4accel2 = filter(f, 1, x4accel);

x1_accel_measurement = [];
m_x1accel = [];
x1_accel_raw = [];
x1accelraw = [];
moving_ = [];
moving_1 = [];
moving_2 = [];

for t = 1:1:3601;
    data = m_x1accel2(t) - (m_x1accel2(1)*((t-3601)/-3600));

```

```

    x1_accel_measurement = [x1_accel_measurement, data];
end

for t = 1:1:3601;
    data = x1_accel_measurement(t) - (x1_accel_measurement(3601)*((t-1)/3600));

    m_x1accel = [m_x1accel, data];
end

for t = 1:1:3601;
    data = x1accel(t) - (x1accel(1)*((t-3601)/-3600));

    x1_accel_raw = [x1_accel_raw, data];
end

for t = 1:1:3601;
    data = x1_accel_raw(t) - (x1_accel_raw(3601)*((t-1)/3600));

    x1accelraw = [x1accelraw, data];
end

for t = 20:1:3601,

    m_ = m_x1accel2(t);
    moving_ = [moving_, m_];

end

for t = 1:1:3582;
    data = moving_(t) - (moving_(1)*((t-3582)/-3581));

    moving_1 = [moving_1, data];
end

for t = 1:1:3582;
    data = moving_1(t) - (moving_1(3582)*((t-1)/3581));

    moving_2 = [moving_2, data];
end

duration = 1800;
dt = 0.5;

v1 = std(x1accel, 1);

```

```

v2 = std(x2accel, 1);
v3 = std(x3accel, 1);
v4 = std(x4accel, 1);

```

```

A = 1;
B = 1;
C = 1;
x = 0;
x1_hat = 0;
x2_hat = 0;
x3_hat = 0;
x4_hat = 0;

```

```

Rv1 = v1^2;
Rv2 = v2^2;
Rv3 = v3^2;
Rv4 = v4^2;

```

```

Qw = 0.00001;
P1 = 1;
P2 = 1;
P3 = 1;
P4 = 1;

```

```

x1_true_accel = [];
x1_accel_est = [];
x1_measurement_noise1 = [];
x1_error = [];
x1_true_acce2 = [];
x1_accel_est2 = [];
x1_measurement_noise2 = [];
x1_accel_measurement = [];
x1_measurement_noise = [];
x1_accel_hat1 = [];
x1_accel_hat = [];
x1_rot_vel = [];
x1_noise_rot_vel = [];

```

```

x2_true_accel = [];
x2_accel_est = [];
x2_measurement_noise1 = [];
x2_error = [];
x2_true_acce2 = [];
x2_accel_est2 = [];
x2_measurement_noise2 = [];

```



```

x2_accel_measurement = [];
x2_measurement_noise = [];
x2_accel_hat1 = [];
x2_accel_hat = [];
x2_rot_vel = [];
x2_noise_rot_vel = [];

```

```

x3_true_accel = [];
x3_accel_est = [];
x3_measurement_noise1 = [];
x3_error = [];
x3_true_acce2 = [];
x3_accel_est2 = [];
x3_measurement_noise2 = [];
x3_accel_measurement = [];
x3_measurement_noise = [];
x3_accel_hat1 = [];
x3_accel_hat = [];
x3_rot_vel = [];
x3_noise_rot_vel = [];

```

```

x4_true_accel = [];
x4_accel_est = [];
x4_measurement_noise1 = [];
x4_error = [];
x4_true_acce2 = [];
x4_accel_est2 = [];
x4_measurement_noise2 = [];
x4_accel_measurement = [];
x4_measurement_noise = [];
x4_accel_hat1 = [];
x4_accel_hat = [];
x4_rot_vel = [];
x4_noise_rot_vel = [];

```

```

for t = 1:1:3601,

```

```

    u = 0;
    x1 = 0;
    x2 = 0;
    x3 = 0;
    x4 = 0;

```

```

    %x1_hat = x1_hat;
    %x2_hat = x2_hat;

```

```
%x3_hat = x3_hat;
%x4_hat = x4_hat;
```

```
P1 = P1 + Qw;
P2 = P2 + Qw;
P3 = P3 + Qw;
P4 = P4 + Qw;
```

```
K1 = P1*inv(P1 + Rv1);
K2 = P2*inv(P2 + Rv2);
K3 = P3*inv(P3 + Rv3);
K4 = P4*inv(P4 + Rv4);
```

```
x1_hat = x1_hat + K1*(x1accel(t) - x1_hat);
x2_hat = x2_hat + K2*(x2accel(t) - x2_hat);
x3_hat = x3_hat + K3*(x3accel(t) - x3_hat);
x4_hat = x4_hat + K4*(x4accel(t) - x4_hat);
```

```
P1 = (1 - K1)*P1;
P2 = (1 - K2)*P2;
P3 = (1 - K3)*P3;
P4 = (1 - K4)*P4;
```

```
x1_true_accel = [x1_true_accel, u];
x1_accel_est = [x1_accel_est, x1_hat];
x1_measurement_noise1 = [x1_measurement_noise1, x1accel(t)];
x1_error = [x1_error, P1];
```

```
x2_true_accel = [x2_true_accel, u];
x2_accel_est = [x2_accel_est, x2_hat];
x2_measurement_noise1 = [x2_measurement_noise1, x2accel(t)];
x2_error = [x2_error, P2];
```

```
x3_true_accel = [x3_true_accel, u];
x3_accel_est = [x3_accel_est, x3_hat];
x3_measurement_noise1 = [x3_measurement_noise1, x3accel(t)];
x3_error = [x3_error, P3];
```

```
x4_true_accel = [x4_true_accel, u];
x4_accel_est = [x4_accel_est, x4_hat];
x4_measurement_noise1 = [x4_measurement_noise1, x4accel(t)];
x4_error = [x4_error, P4];
```

```
end
```

```

for t = 20:1:3601,

    x1_ = x1_true_accel(t);
    x1_true_acce2 = [x1_true_acce2, x1_];

    y1_ = x1_accel_est(t);
    x1_accel_est2 = [x1_accel_est2, y1_];

    z1_ = x1_measurement_noise1(t);
    x1_measurement_noise2 = [x1_measurement_noise2, z1_];

```

```

end

```

```

for t = 20:1:3601,

    x2_ = x2_true_accel(t);
    x2_true_acce2 = [x2_true_acce2, x2_];

    y2_ = x2_accel_est(t);
    x2_accel_est2 = [x2_accel_est2, y2_];

    z2_ = x2_measurement_noise1(t);
    x2_measurement_noise2 = [x2_measurement_noise2, z2_];

```

```

end

```

```

for t = 20:1:3601,

    x3_ = x3_true_accel(t);
    x3_true_acce2 = [x3_true_acce2, x3_];

    y3_ = x3_accel_est(t);
    x3_accel_est2 = [x3_accel_est2, y3_];

    z3_ = x3_measurement_noise1(t);
    x3_measurement_noise2 = [x3_measurement_noise2, z3_];

```

```

end

```

```

for t = 20:1:3601,

    x4_ = x4_true_accel(t);
    x4_true_acce2 = [x4_true_acce2, x4_];

    y4_ = x4_accel_est(t);
    x4_accel_est2 = [x4_accel_est2, y4_];

```

```

    z4_ = x4_measurement_noise1(t);
    x4_measurement_noise2 = [x4_measurement_noise2, z4_];

end

close all;

time = 0:0.5:1790.5;
time1 = 0:dt:duration;

for t = 1:1:3582;
    data = x1_measurement_noise1(t) - (x1_measurement_noise1(1)*((t-3582)/-3581));

    x1_accel_measurement = [x1_accel_measurement, data];
end

for t = 1:1:3582;
    data = x1_accel_measurement(t) - (x1_accel_measurement(3582)*((t-1)/3581));

    x1_measurement_noise = [x1_measurement_noise, data];
end

for t = 1:1:3582;
    data = x1_accel_est(t) - (x1_accel_est(1)*((t-3582)/-3581));

    x1_accel_hat1 = [x1_accel_hat1, data];
end

for t = 1:1:3582;
    data = x1_accel_hat1(t) - (x1_accel_hat1(3582)*((t-1)/3581));

    x1_accel_hat = [x1_accel_hat, data];
end

x1_velocity = cumtrapz(time, x1_measurement_noise);
x1_vel_hat = cumtrapz(time, x1_accel_hat);

x1_displacement = cumtrapz(time, x1_velocity);
x1_pos_hat = cumtrapz(time, x1_vel_hat);

for t = 1:1:3582;
    data = x1_vel_hat(t) - (x1_vel_hat(3582)*((t-1)/3581));

```

```

    x1_rot_vel = [x1_rot_vel, data];
end

for t = 1:1:3582;
    data = x1_velocity(t) - (x1_velocity(3582)*((t-1)/3581));

    x1_noise_rot_vel = [x1_noise_rot_vel, data];
end

x1_rot_pos = cumtrapz(time, x1_rot_vel);
x1_noise_rot_pos = cumtrapz(time, x1_noise_rot_vel);

%-----

for t = 1:1:3582;
    data = x2_measurement_noise1(t) - (x2_measurement_noise1(1)*((t-3582)/-3581));

    x2_accel_measurement = [x2_accel_measurement, data];
end

for t = 1:1:3582;
    data = x2_accel_measurement(t) - (x2_accel_measurement(3582)*((t-1)/3581));

    x2_measurement_noise = [x2_measurement_noise, data];
end

for t = 1:1:3582;
    data = x2_accel_est(t) - (x2_accel_est(1)*((t-3582)/-3581));

    x2_accel_hat1 = [x2_accel_hat1, data];
end

for t = 1:1:3582;
    data = x2_accel_hat1(t) - (x2_accel_hat1(3582)*((t-1)/3581));

    x2_accel_hat = [x2_accel_hat, data];
end

x2_velocity = cumtrapz(time, x2_measurement_noise);
x2_vel_hat = cumtrapz(time, x2_accel_hat);

x2_displacement = cumtrapz(time, x2_velocity);
x2_pos_hat = cumtrapz(time, x2_vel_hat);

```

```

for t = 1:1:3582;
    data = x2_vel_hat(t) - (x2_vel_hat(3582)*((t-1)/3581));

    x2_rot_vel = [x2_rot_vel, data];
end

for t = 1:1:3582;
    data = x2_velocity(t) - (x2_velocity(3582)*((t-1)/3581));

    x2_noise_rot_vel = [x2_noise_rot_vel, data];
end

x2_rot_pos = cumtrapz(time, x2_rot_vel);
x2_noise_rot_pos = cumtrapz(time, x2_noise_rot_vel);

%-----

for t = 1:1:3582;
    data = x3_measurement_noise1(t) - (x3_measurement_noise1(1)*((t-3582)/-3581));

    x3_accel_measurement = [x3_accel_measurement, data];
end

for t = 1:1:3582;
    data = x3_accel_measurement(t) - (x3_accel_measurement(3582)*((t-1)/3581));

    x3_measurement_noise = [x3_measurement_noise, data];
end

for t = 1:1:3582;
    data = x3_accel_est(t) - (x3_accel_est(1)*((t-3582)/-3581));

    x3_accel_hat1 = [x3_accel_hat1, data];
end

for t = 1:1:3582;
    data = x3_accel_hat1(t) - (x3_accel_hat1(3582)*((t-1)/3581));

    x3_accel_hat = [x3_accel_hat, data];
end

x3_velocity = cumtrapz(time, x3_measurement_noise);
x3_vel_hat = cumtrapz(time, x3_accel_hat);

```

```

x3_displacement = cumtrapz(time, x3_velocity);
x3_pos_hat = cumtrapz(time, x3_vel_hat);

for t = 1:1:3582;
    data = x3_vel_hat(t) - (x3_vel_hat(3582)*((t-1)/3581));

    x3_rot_vel = [x3_rot_vel, data];
end

for t = 1:1:3582;
    data = x3_velocity(t) - (x3_velocity(3582)*((t-1)/3581));

    x3_noise_rot_vel = [x3_noise_rot_vel, data];
end

x3_rot_pos = cumtrapz(time, x3_rot_vel);
x3_noise_rot_pos = cumtrapz(time, x3_noise_rot_vel);

%-----

for t = 1:1:3582;
    data = x4_measurement_noise1(t) - (x4_measurement_noise1(1)*((t-3582)/-3581));

    x4_accel_measurement = [x4_accel_measurement, data];
end

for t = 1:1:3582;
    data = x4_accel_measurement(t) - (x4_accel_measurement(3582)*((t-1)/3581));

    x4_measurement_noise = [x4_measurement_noise, data];
end

for t = 1:1:3582;
    data = x4_accel_est(t) - (x4_accel_est(1)*((t-3582)/-3581));

    x4_accel_hat1 = [x4_accel_hat1, data];
end

for t = 1:1:3582;
    data = x4_accel_hat1(t) - (x4_accel_hat1(3582)*((t-1)/3581));

    x4_accel_hat = [x4_accel_hat, data];
end

```

```

x4_velocity = cumtrapz(time, x4_measurement_noise);
x4_vel_hat = cumtrapz(time, x4_accel_hat);

x4_displacement = cumtrapz(time, x4_velocity);
x4_pos_hat = cumtrapz(time, x4_vel_hat);

for t = 1:1:3582;
    data = x4_vel_hat(t) - (x4_vel_hat(3582)*((t-1)/3581));

    x4_rot_vel = [x4_rot_vel, data];
end

for t = 1:1:3582;
    data = x4_velocity(t) - (x4_velocity(3582)*((t-1)/3581));

    x4_noise_rot_vel = [x4_noise_rot_vel, data];
end

x4_rot_pos = cumtrapz(time, x4_rot_vel);
x4_noise_rot_pos = cumtrapz(time, x4_noise_rot_vel);

%-----

figure(1);
plot(time1, x1accel, 'r', time1, x1_true_accel, 'g');
grid
legend('Measured Acceleration', 'Actual Acceleration');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration in x-axis');

x_velocity1 = cumtrapz(time1, x1accel);
x_act_velocity1 = cumtrapz(time1, x1_true_accel);

figure(2);
plot(time1, x_velocity1, 'r', time1, x_act_velocity1, 'g');
grid
legend('Measured Velocity', 'Actual Velocity');
xlabel('Time(s)');
ylabel('Velocity (m/s)');
title('Velocity in x-axis');

x_displacement1 = cumtrapz(time1, x_velocity1);
x_act_displacement1 = cumtrapz(time1, x_act_velocity1);

```



```

figure(3);
plot(time1, x_displacement1, 'r', time1, x_act_displacement1, 'g');
grid
legend('Measured Displacement', 'Actual Displacement');
xlabel('Time(s)');
ylabel('Displacement (m)');
title('Displacement in x-axis');

```

```

figure(4);
plot(time1, x1accel, 'r', time1, x1_accel_est, 'b', time1, x1_true_accel, 'g');
grid
legend('Measured Acceleration', 'Kalman Filter', 'Actual Acceleration');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration in x-axis and Kalman Filter');

```

```

figure(5);
plot(time1, x1_error, 'r');
grid
legend('The Error Covariance');
xlabel('Time(s)');
ylabel('Error Covariance');
title('The Error Covariance');

```

```

figure(6);
plot(time, x1_measurement_noise2, 'r', time, x1_accel_est2, 'b', time,
x1_true_acce2, 'g');
grid
legend('Measured Acceleration', 'Kalman Filter', 'Actual Acceleration');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration in x-axis and Kalman Filter with the first 20 second is cut out');

```

```

x_act_velocity1_cut = cumtrapz(time, x1_true_acce2);

```

```

figure(7);
plot(time, x1_measurement_noise, 'r', time, x1_accel_hat, 'b', time,
x1_true_acce2, 'g');
grid
legend('C.Ri.Rf', 'F.C.Ri.Rf.', 'Actual Acceleration');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration in x-axis and Kalman Filter with boundary condition and the first
20 second is cut out');

```

```
figure(8);
plot(time, x1_velocity, 'r', time, x1_vel_hat, 'b', time, x_act_velocity1_cut, 'g');
grid;
legend('C.Ri.Rf.I.', 'F.C.Ri.Rf.I.', 'Actual Velocity');
xlabel('Time(s)');
ylabel('Velocity (m/s)');
title('Velocity in x-axis with the velocity when applying Kalman Filter');
```

```
x_act_displacement1_cut = cumtrapz(time, x_act_velocity1_cut);
```

```
figure(9);
plot(time, x1_displacement, 'r', time, x1_pos_hat, 'b', time,
x_act_displacement1_cut, 'g');
grid;
legend('C.Ri.Rf.I.I.', 'F.C.Ri.Rf.I.I.', 'Actual Displacement');
xlabel('Time(s)');
ylabel('displacement (m)');
title('Displacement in x-axis with the displacement when applying Kalman Filter');
```

```
figure(10);
plot(time, x1_velocity, 'r', time, x1_noise_rot_vel, 'r--', time, x1_vel_hat, 'b', time,
x1_rot_vel, 'b--', time, x_act_velocity1_cut, 'g');
grid;
legend('C.Ri.Rf.I.', 'C.Ri.Rf.I.Rf.', 'F.C.Ri.Rf.I.', 'F.C.Ri.Rf.I.Rf.', 'Actual Velocity');
xlabel('Time(s)');
ylabel('Velocity (m/s)');
title('Velocity in x-axis with the velocity when applying Kalman Filter with and
without velocity boundary conditions');
```

```
figure(11);
plot(time, x1_displacement, 'r', time, x1_noise_rot_pos, 'r--', time, x1_pos_hat, 'b',
time, x1_rot_pos, 'b--', time, x_act_displacement1_cut, 'g');
grid;
legend('C.Ri.Rf.I.I.', 'C.Ri.Rf.I.Rf.I.', 'F.C.Ri.Rf.I.I.', 'F.C.Ri.Rf.I.Rf.I.', 'Actual
Displacement');
xlabel('Time(s)');
ylabel('displacement (m)');
title('Displacement in x-axis with the displacement when applying Kalman Filter
with and without velocity boundary conditions');
```

```
figure(12);
plot(time1, x1accel, 'r', time1, x2accel, 'y', time1, x3accel, 'c', time1, x4accel, 'm');
grid;
legend('Accelerometer 1', 'Accelerometer 2', 'Accelerometer 3', 'Accelerometer
4');
xlabel('Time(s)');
```

```
ylabel('Acceleration (m/s^2)');
title('4 Accelerometers in x-axis');
```

```
figure(13);
plot(time, x1_measurement_noise, 'r', time, x2_measurement_noise, 'y', time,
x3_measurement_noise, 'c', time, x4_measurement_noise, 'm');
grid
legend('Accelerometer 1', 'Accelerometer 2', 'Accelerometer 3', 'Accelerometer
4');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('4 Accelerometers in x-axis with the boundary conditions');
```

```
figure(14);
plot(time1, x1accelraw, 'r', time1, m_x1accel, 'k', time1, x1_true_accel, 'g');
grid
legend('Ri.Rf.', 'M21.Ri.Rf.', 'Actual Acceleration');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Acceleration in x-axis');
```

```
figure(15);
plot(time, x1_measurement_noise, 'r', time, x1_accel_hat, 'b', time, moving_2, 'k',
time, x1_true_acce2, 'g');
grid
legend('C.Ri.Rf', 'F.C.Ri.Rf.', 'M21.C.Ri.Rf.', 'Actual Acceleration');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)')
title('Acceleration in x-axis and Kalman Filter with boundary condition and the first
20 second is cut out');
```

Parallel Design:

```
function kalman(duration, dt)
```

```
file = fopen('3600_13_Jan_11_4_3_2_1.txt', 'r');
data = fscanf(file, '%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d',
[16 inf]);
```

```
fclose(file);
```

```
data = data';
```

```
x1 = (data(:,2)*4)+(data(:,1)/64);
x2 = (data(:,6)*4)+(data(:,5)/64);
x3 = (data(:,10)*4)+(data(:,9)/64);
```

```

x4 = (data(:,14)*4)+(data(:,13)/64);

x1voltagearray = (x1*3.3)/1023;
x2voltagearray = (x2*3.3)/1023;
x3voltagearray = (x3*3.3)/1023;
x4voltagearray = (x4*3.3)/1023;

x1accel = [(9.80665*(x1voltagearray - 1.65))/0.626];
x2accel = [(9.80665*(x2voltagearray - 1.65))/0.626];
x3accel = [(9.80665*(x3voltagearray - 1.65))/0.626];
x4accel = [(9.80665*(x4voltagearray - 1.65))/0.626];

xaccel = [];

for t = 1:1:3601,

xaccel = [x1accel; x2accel; x3accel; x4accel];

end

duration = 1800;
dt = 0.5;

v1 = std(x1accel, 1);
v2 = std(x2accel, 1);
v3 = std(x3accel, 1);
v4 = std(x4accel, 1);

A = 1;
C = [1 1 1 1];
x_hat = 0;

Rv1 = v1^2;
Rv2 = v2^2;
Rv3 = v3^2;
Rv4 = v4^2;

k = [0; 0; 0; 0];

Rv = [Rv1 0 0 0; 0 Rv2 0 0; 0 0 Rv3 0; 0 0 0 Rv4];

Qw = 0.00001;
P = 1;

x_true = [];
x_hat_1 = [];

```

```

x_measurement = [];
x_error = [];

x_true_accel = [];
x_accel_est = [];
x_measurement_noise1 = [];
x_accel_measurement = [];
x_measurement_noise = [];
x_accel_hat1 = [];
x_accel_hat = [];
x_rot_vel = [];
x_noise_rot_vel = [];

for t = 1:1:3601,

    u = 0;

    %P = A * P * A' + Qw;
    P = P + Qw;

    %P = inv(inv(P) + ((transpose(C(:,1)) * inv(Rv1) * C(:,1)) + (transpose(C(:,2)) *
inv(Rv2) * C(:,2)) + (transpose(C(:,3)) * inv(Rv3) * C(:,3)) + (transpose(C(:,4)) *
inv(Rv4) * C(:,4))));
    P = inv(inv(P) + (inv(Rv1) + inv(Rv2) + inv(Rv3) + inv(Rv4)));

    %k(1,:) = P * transpose(C(:,1)) * inv(Rv1);
    %k(2,:) = P * transpose(C(:,2)) * inv(Rv2);
    %k(3,:) = P * transpose(C(:,3)) * inv(Rv3);
    %k(4,:) = P * transpose(C(:,4)) * inv(Rv4);
    k(1,:) = P * inv(Rv1);
    k(2,:) = P * inv(Rv2);
    k(3,:) = P * inv(Rv3);
    k(4,:) = P * inv(Rv4);

    %x_hat = x_hat + ((k(1,:) * (x1accel(t) - (C(:,1) * x_hat))) + (k(2,:) * (x2accel(t) -
(C(:,2) * x_hat))) + (k(3,:) * (x3accel(t) - (C(:,3) * x_hat))) + (k(4,:) * (x4accel(t) -
(C(:,4) * x_hat))));
    x_hat = x_hat + ((k(1,:) * (x1accel(t) - x_hat)) + (k(2,:) * (x2accel(t) - x_hat)) +
(k(3,:) * (x3accel(t) - x_hat)) + (k(4,:) * (x4accel(t) - x_hat)));

    x_true = [x_true, u];
    x_hat_1 = [x_hat_1, x_hat];
    x_measurement = [x_measurement, xaccel(t)];
    x_error = [x_error, P];

end

```

```

for t = 1:1:3601,

    x_ = x_true(t);
    x_true_accel = [x_true_accel, x_];

    y_ = x_hat_1(t);
    x_accel_est = [x_accel_est, y_];

    z_ = x_measurement(t);
    x_measurement_noise1 = [x_measurement_noise1, z_];

end

close all;

time = 0:dt:duration;

for t = 1:1:3601;
    data = x_measurement_noise1(t) - (x_measurement_noise1(1)*((t-3601)/-3600));

    x_accel_measurement = [x_accel_measurement, data];
end

for t = 1:1:3601;
    data = x_accel_measurement(t) - (x_accel_measurement(3601)*((t-1)/3600));

    x_measurement_noise = [x_measurement_noise, data];
end

for t = 1:1:3601;
    data = x_accel_est(t) - (x_accel_est(1)*((t-3601)/-3600));

    x_accel_hat1 = [x_accel_hat1, data];
end

for t = 1:1:3601;
    data = x_accel_hat1(t) - (x_accel_hat1(3601)*((t-1)/3600));

    x_accel_hat = [x_accel_hat, data];
end

x_velocity = cumtrapz(time, x_measurement_noise);
x_vel_hat = cumtrapz(time, x_accel_hat);

```

```

x_displacement = cumtrapz(time, x_velocity);
x_pos_hat = cumtrapz(time, x_vel_hat);

for t = 1:1:3601;
    data = x_vel_hat(t) - (x_vel_hat(3601)*((t-1)/3600));

    x_rot_vel = [x_rot_vel, data];
end

for t = 1:1:3601;
    data = x_velocity(t) - (x_velocity(3601)*((t-1)/3600));

    x_noise_rot_vel = [x_noise_rot_vel, data];
end

x_rot_pos = cumtrapz(time, x_rot_vel);
x_noise_rot_pos = cumtrapz(time, x_noise_rot_vel);

figure(1);
subplot(3,1,1);
plot(time, x_measurement_noise, 'r', time, x_accel_hat, 'b', time, x_true_accel,
'y');
grid;
legend('Ri.Rf.', 'F.Ri.Rf.', 'T. ');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Figure 1 - Acceleration in x');

subplot(3,1,2);
plot(time, x_velocity, 'r', time, x_noise_rot_vel, 'r--', time, x_vel_hat, 'b', time,
x_rot_vel, 'b--');
grid;
legend('Ri.Rf.I.', 'Ri.Rf.I.Rf', 'F.Ri.Rf.I.', 'F.Ri.Rf.I.Rf');
xlabel('Time(s)');
ylabel('Velocity (m/s)');
title('Figure 2 - Velocity in x');

subplot(3,1,3);
plot(time, x_displacement, 'r', time, x_noise_rot_pos, 'r--', time, x_pos_hat, 'b',
time, x_rot_pos, 'b--');
grid;
legend('Ri.Rf.I.I.', 'Ri.Rf.I.Rf.I.', 'F.Ri.Rf.I.I.', 'F.Ri.Rf.I.Rf.I. ');
xlabel('Time(s)');
ylabel('displacement (m)');
title('Figure 3 - Displacement in x');

```

```
XRiRfll = x_displacement(3601)
```

```
XRiRfIRfl = x_noise_rot_pos(3601)
```

```
XFRiRfll = x_pos_hat(3601)
```

```
XFRiRfIRfl = x_rot_pos(3601)
```

Series Design:

```
function kalman(duration, dt)
```

```
file = fopen('3600_13_Jan_11_4_3_2_1.txt', 'r');
data = fscanf(file, '%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d',
[16 inf]);
```

```
fclose(file);
```

```
data = data';
```

```
x1 = (data(:,2)*4)+(data(:,1)/64);
x2 = (data(:,6)*4)+(data(:,5)/64);
x3 = (data(:,10)*4)+(data(:,9)/64);
x4 = (data(:,14)*4)+(data(:,13)/64);
```

```
x1voltagearray = (x1*3.3)/1023;
x2voltagearray = (x2*3.3)/1023;
x3voltagearray = (x3*3.3)/1023;
x4voltagearray = (x4*3.3)/1023;
```

```
x1accel = [(9.80665*(x1voltagearray - 1.65))/0.626];
x2accel = [(9.80665*(x2voltagearray - 1.65))/0.626];
x3accel = [(9.80665*(x3voltagearray - 1.65))/0.626];
x4accel = [(9.80665*(x4voltagearray - 1.65))/0.626];
```

```
xaccel = [x1accel; x2accel; x3accel; x4accel];
```

```
for t = 1:1:3601,
```

```
xaccel = [x1accel x2accel x3accel x4accel];
```

```
end
```

```
duration = 1800;
```



```

dt = 0.5;

v1 = std(x1accel, 1);
v2 = std(x2accel, 1);
v3 = std(x3accel, 1);
v4 = std(x4accel, 1);

A = 1;
C = [1 1 1 1];
x_hat = 0;

Rv = [v1^2; v2^2; v3^2; v4^2];

k = [0; 0; 0; 0];

Qw = 0.00001;
P = 1;

x_true = [];
x_hat_1 = [];
x_measurement = [];
x_error = [];

x_true_accel = [];
x_accel_est = [];
x_measurement_noise1 = [];
x_accel_measurement = [];
x_measurement_noise = [];
x_accel_hat1 = [];
x_accel_hat = [];
x_rot_vel = [];
x_noise_rot_vel = [];

for t = 1:1:3601,

    u = 0;

    P = P + Qw;

    for i = 1:1:4,

        P = inv(inv(P) + inv(Rv(i,:)));

        k = P * inv(Rv(i,:));

        x_hat = x_hat + (k * (xaccel(t,i)-x_hat));

```

```

end

x_true = [x_true, u];
x_hat_1 = [x_hat_1, x_hat];
x_measurement = [x_measurement, xaccel(t)];
x_error = [x_error, P];

end

for t = 1:1:3601,

    x_ = x_true(t);
    x_true_accel = [x_true_accel, x_];

    y_ = x_hat_1(t);
    x_accel_est = [x_accel_est, y_];

    z_ = x_measurement(t);
    x_measurement_noise1 = [x_measurement_noise1, z_];

end

close all;

time = 0:dt:duration;

for t = 1:1:3601;
    data = x_measurement_noise1(t) - (x_measurement_noise1(1)*((t-3601)/-3600));

    x_accel_measurement = [x_accel_measurement, data];
end

for t = 1:1:3601;
    data = x_accel_measurement(t) - (x_accel_measurement(3601)*((t-1)/3600));

    x_measurement_noise = [x_measurement_noise, data];
end

for t = 1:1:3601;
    data = x_accel_est(t) - (x_accel_est(1)*((t-3601)/-3600));

    x_accel_hat1 = [x_accel_hat1, data];
end

```

```

for t = 1:1:3601;
    data = x_accel_hat1(t) - (x_accel_hat1(3601)*((t-1)/3600));

    x_accel_hat = [x_accel_hat, data];
end

x_velocity = cumtrapz(time, x_measurement_noise);
x_vel_hat = cumtrapz(time, x_accel_hat);

x_displacement = cumtrapz(time, x_velocity);
x_pos_hat = cumtrapz(time, x_vel_hat);

for t = 1:1:3601;
    data = x_vel_hat(t) - (x_vel_hat(3601)*((t-1)/3600));

    x_rot_vel = [x_rot_vel, data];
end

for t = 1:1:3601;
    data = x_velocity(t) - (x_velocity(3601)*((t-1)/3600));

    x_noise_rot_vel = [x_noise_rot_vel, data];
end

x_rot_pos = cumtrapz(time, x_rot_vel);
x_noise_rot_pos = cumtrapz(time, x_noise_rot_vel);

figure(1);
subplot(3,1,1);
plot(time, x_measurement_noise, 'r', time, x_accel_hat, 'b', time, x_true_accel,
'y');
grid;
legend('Ri.Rf.', 'F.Ri.Rf.', 'T. ');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Figure 1 - Acceleration in x');

subplot(3,1,2);
plot(time, x_velocity, 'r', time, x_noise_rot_vel, 'r--', time, x_vel_hat, 'b', time,
x_rot_vel, 'b--');
grid;
legend('Ri.Rf.I.', 'Ri.Rf.I.Rf', 'F.Ri.Rf.I.', 'F.Ri.Rf.I.Rf');
xlabel('Time(s)');
ylabel('Velocity (m/s)');
title('Figure 2 - Velocity in x');

```

```

subplot(3,1,3);
plot(time, x_displacement, 'r', time, x_noise_rot_pos, 'r--', time, x_pos_hat, 'b',
time, x_rot_pos, 'b--');
grid
legend('Ri.Rf.I.I.', 'Ri.Rf.I.Rf.I.', 'F.Ri.Rf.I.I.', 'F.Ri.Rf.I.Rf.I');
xlabel('Time(s)');
ylabel('displacement (m)');
title('Figure 3 - Displacement in x');

XRiRfI = x_displacement(3601)

XRiRfIRfI = x_noise_rot_pos(3601)

XFRiRfI = x_pos_hat(3601)

XFRiRfIRfI = x_rot_pos(3601)

```

Compression Design:

```

function kalman(duration, dt)

file = fopen('3600_13_Jan_11_4_3_2_1.txt', 'r');
data = fscanf(file, '%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d',
[16 inf]);

fclose(file);

data = data';

x1 = (data(:,2)*4)+(data(:,1)/64);
x2 = (data(:,6)*4)+(data(:,5)/64);
x3 = (data(:,10)*4)+(data(:,9)/64);
x4 = (data(:,14)*4)+(data(:,13)/64);

x1voltagearray = (x1*3.3)/1023;
x2voltagearray = (x2*3.3)/1023;
x3voltagearray = (x3*3.3)/1023;
x4voltagearray = (x4*3.3)/1023;

x1accel = [(9.80665*(x1voltagearray - 1.65))/0.626];
x2accel = [(9.80665*(x2voltagearray - 1.65))/0.626];
x3accel = [(9.80665*(x3voltagearray - 1.65))/0.626];
x4accel = [(9.80665*(x4voltagearray - 1.65))/0.626];

```

```

xaccel = [];

duration = 1800;
dt = 0.5;

v1 = std(x1accel, 1);
v2 = std(x2accel, 1);
v3 = std(x3accel, 1);
v4 = std(x4accel, 1);

v = ((v1 + v2 + v3 + v4)/4);

A = 1;
B = 1;
C = 1;
x = 0;
x1_hat = 0;
x2_hat = 0;
x3_hat = 0;
x4_hat = 0;

x_hat = ((x1_hat + x2_hat + x3_hat + x4_hat)/4);

Rv1 = v1^2;
Rv2 = v2^2;
Rv3 = v3^2;
Rv4 = v4^2;

Rv = inv(inv(Rv1) + inv(Rv2) + inv(Rv3) + inv(Rv4));

for t = 1:1:3601,

z = Rv * ((inv(Rv1) * x1accel(t)) + (inv(Rv2) * x2accel(t)) + (inv(Rv3) * x3accel(t))
+ (inv(Rv4) * x4accel(t)));

xaccel = [xaccel, z];

end

Qw = 0.00001;
P1 = 1;
P2 = 1;
P3 = 1;
P4 = 1;

P = ((P1 + P2 + P3 + P4)/4);

```

```

x_true = [];
x_hat_1 = [];
x_measurement = [];
x_error = [];

x_true_accel = [];
x_accel_est = [];
x_measurement_noise1 = [];
x_accel_measurement = [];
x_measurement_noise = [];
x_accel_hat1 = [];
x_accel_hat = [];
x_rot_vel = [];
x_noise_rot_vel = [];

for t = 1:1:3601,

    u = 0;
    x = 0;

    %x1_hat = x1_hat;

    P = P + Qw;

    K = P*inv(P + Rv);

    x_hat = x_hat + K*(xaccel(t) - x_hat);

    P = (1 - K)*P;

    x_true = [x_true, u];
    x_hat_1 = [x_hat_1, x_hat];
    x_measurement = [x_measurement, xaccel(t)];
    x_error = [x_error, P];

end

for t = 1:1:3601,

    x_ = x_true(t);
    x_true_accel = [x_true_accel, x_];

    y_ = x_hat_1(t);
    x_accel_est = [x_accel_est, y_];

```

```

    z_ = x_measurement(t);
    x_measurement_noise1 = [x_measurement_noise1, z_];

end

close all;

time = 0:dt:duration;

for t = 1:1:3601;
    data = x_measurement_noise1(t) - (x_measurement_noise1(1)*((t-3601)/-3600));

    x_accel_measurement = [x_accel_measurement, data];
end

for t = 1:1:3601;
    data = x_accel_measurement(t) - (x_accel_measurement(3601)*((t-1)/3600));

    x_measurement_noise = [x_measurement_noise, data];
end

for t = 1:1:3601;
    data = x_accel_est(t) - (x_accel_est(1)*((t-3601)/-3600));

    x_accel_hat1 = [x_accel_hat1, data];
end

for t = 1:1:3601;
    data = x_accel_hat1(t) - (x_accel_hat1(3601)*((t-1)/3600));

    x_accel_hat = [x_accel_hat, data];
end

x_velocity = cumtrapz(time, x_measurement_noise);
x_vel_hat = cumtrapz(time, x_accel_hat);

x_displacement = cumtrapz(time, x_velocity);
x_pos_hat = cumtrapz(time, x_vel_hat);

for t = 1:1:3601;
    data = x_vel_hat(t) - (x_vel_hat(3601)*((t-1)/3600));

    x_rot_vel = [x_rot_vel, data];
end

```

```

for t = 1:1:3601;
    data = x_velocity(t) - (x_velocity(3601)*((t-1)/3600));

    x_noise_rot_vel = [x_noise_rot_vel, data];
end

x_rot_pos = cumtrapz(time, x_rot_vel);
x_noise_rot_pos = cumtrapz(time, x_noise_rot_vel);

figure(1);
subplot(3,1,1);
plot(time, x_measurement_noise, 'r', time, x_accel_hat, 'b', time, x_true_accel,
'y');
grid;
legend('Ri.Rf.', 'F.Ri.Rf.', 'T. ');
xlabel('Time(s)');
ylabel('Acceleration (m/s^2)');
title('Figure 1 - Acceleration in x');

subplot(3,1,2);
plot(time, x_velocity, 'r', time, x_noise_rot_vel, 'r--', time, x_vel_hat, 'b', time,
x_rot_vel, 'b--');
grid;
legend('Ri.Rf.I.', 'Ri.Rf.I.Rf', 'F.Ri.Rf.I.', 'F.Ri.Rf.I.Rf');
xlabel('Time(s)');
ylabel('Velocity (m/s)');
title('Figure 2 - Velocity in x');

subplot(3,1,3);
plot(time, x_displacement, 'r', time, x_noise_rot_pos, 'r--', time, x_pos_hat, 'b',
time, x_rot_pos, 'b--');
grid;
legend('Ri.Rf.I.I.', 'Ri.Rf.I.Rf.I.', 'F.Ri.Rf.I.I.', 'F.Ri.Rf.I.Rf.I. ');
xlabel('Time(s)');
ylabel('displacement (m)');
title('Figure 3 - Displacement in x');

XRIrfill = x_displacement(3601)

XRIrflRfl = x_noise_rot_pos(3601)

XFRiRfill = x_pos_hat(3601)

XFRiRflRfl = x_rot_pos(3601)

```