The Pennsylvania State University

The Graduate School

College of Engineering

KALMAN FILTER BASED ESTIMATION OF INERTIAL MEASUREMENT UNIT PARAMETERS IN A PORTABLE BIOMECHANICAL ASSESSMENT SUITE (PBAS)

A Thesis in

Electrical Engineering

by

Bharath Ramaswamy

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2011

The thesis of Bharath Ramaswamy was reviewed and approved* by the following:

H. Joseph Sommer III

Professor of Mechanical Engineering
Thesis Co-Advisor

Kenneth Jenkins

Professor and Head of Electrical Engineering
Thesis Co-advisor

Jeffery Schiano

Associate Professor of Electrical Engineering

*Signatures are on file in the Graduate School

ABSTRACT

Musculoskeletal disorders such as low back pain, joint injuries and repetitive strain injuries have affected a sizeable percentage of population in the US and are generally related to occupational tasks such as lifting. There have been several attempts to study the biomechanical aspects of such occupational tasks and quantify safe working practice. A Portable Biomechanical Assessment Suite (PBAS) was designed and tested to be worn by workers during their regular occupational activities and provide useful data for analysis. Kinematic data is measured by Inertial Measurement Units (IMUs) consisting of a three-axis accelerometer, a two-axis gyroscope, and a single-axis gyroscope. A Kalman Filter algorithm has been developed to smooth the data obtained from the IMU. Three different Kalman Filter algorithms, namely, Linear Kalman filter, Extended Kalman Filter and Unscented Kalman filter were implemented for this purpose and their performance has been compared based on the root mean square estimation error. Additionally, data sent from the IMU is sometimes lost due to radio packet collisions. A forward backward algorithm was developed to interpolate the missing values due to IMU dropouts.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The author would like to express his sincere thanks to Dr. H.J. Sommer III for his advice and support throughout this project. His words of encouragement and constant mentoring have immensely helped me in carrying out this research.

The concept of a Portable Biomechanical Assessment Suite (PBAS) was developed by Dr. Frank Buczek, at the National Institute for Occupational Safety and Health (NIOSH), Morgantown, WV. The author gratefully acknowledges financial support from NIOSH under PO 212-2009-M-31391.

Words fail you when you need them the most. The author has no words but a deep sense of gratitude and affection to acknowledge his parents Mrs. R. Vijayalakshmi and Mr. K. Ramaswamy and sister Vidhya for all the sacrifice, motivation and their sheer presence in the author's life. In the spirit of a famous hymn (originally in Sanskrit) which says:
`In other lands I am honored; in my country I am fortunate; in the ways of good conduct there is none that excels me- thus one may think; but if one's mind be not attached to the lotus feet of the Guru, what thence, what thence, what thence, what thence?", the author, with a bowed head and folded hands, dedicates any small achievement from his side to the lotus feet of His Holinesses of Kanchi, the spiritual gurus.

Last, but not the least, the author acknowledges the perennial friendship and moral support from Adith, Anjum, Bharath, Ganesh, Gautham and Harish.

Chapter 1

Occupational Disorders

The ultimate aim of this project is to reduce the incidence of musculoskeletal disorders related to occupational lifting tasks. Musculoskeletal disorders such as low back pain, joint injuries and repetitive strain injuries affect a sizeable percentage of the population in the United States. As early as the eighteenth century, the cause of these disorders was related to occupational etiological factors. However epidemiologic methods were not applied to study this problem (Bernard et al., 1997) until the 1970s. This has given rise to numerous biomechanical models to describe different occupational activities like lifting. The input data, required to validate these models, have been obtained through different experimental setups. In order to obtain realistic data to assess the risks involved in occupational activities, there is a need for compact and accurate instrumentation to be worn by workers and provide data directly from the worksite. The prototype of a Portable Biomechanical Assessment Suite (PBAS) has been developed and can be worn by workers while performing their occupational activities.

More specifically, the project aims at developing a sensor suite which can be worn by workers to measure the loading on the lower back during occupational lifting tasks in the workplace. This data can then be used to develop intervention strategies and keep workers safe. The PBAS has immediate application by measuring and quantifying the loads subjected during repetitive activities. This system can help improve the existing musculoskeletal models by providing data on body kinematics and external loading in actual field conditions, which was previously unavailable.

Chapter 2

Portable Biomechanical Assessment suite (PBAS)

The PBAS was designed to measure the stress on the musculoskeletal system in terms of body motion (kinematics) and forces causing or resulting from the motion (kinetics) during a full work day. This data can be used directly in biomechanical models that estimate forces on important structural areas of the body (e.g., spinal discs). It can also be used in current lifting equations to determine safe combinations of lifting loads and the frequency of lifts as well as in assessing repetitive motion of body segments.

This prototype focuses on the input to two widely accepted biomechanical safety models associated with lifting tasks (Freivalds et al., 1984 and Waters et al., 1993). Kinematic data needed for these models includes the motion of the head, thorax, pelvis, upper extremities, and lower extremities. Only motion in the sagittal plane will be measured. The kinetic data needed for these models are the external forces applied to the body at the hands and feet.

Kinematic data are measured by Inertial Measurement Units (IMUs) consisting of a three-axis accelerometer, a two-axis gyroscope, and a single-axis gyroscope as shown in Figure 1. The IMUs measure two acceleration signals and one gyroscope signal in the sagittal plane or two acceleration signals and one gyroscope signal in the coronal plane and send the data via a wireless network protocol. Data is collected by a microprocessor and written to a secure digital (SD) memory card for post processing.

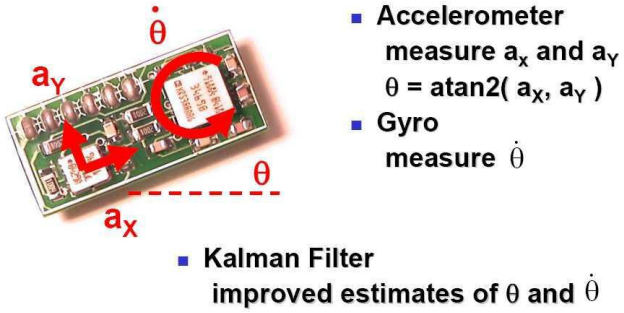

Figure 1: The Inertial Measurement Unit (IMU).

The data obtained from the IMUs is affected by the presence of noise. This necessitates the use of a computationally efficient filter to smooth the data and estimate the IMU parameters. The noise is assumed to follow a Gaussian distribution. This project utilizes a Kalman Filter for parameter estimation. The Kalman Filter is a set of mathematical equations that perform computations recursively on the mean estimates of the state of a system, to minimize the mean of the squared error. The filter supports estimation of past, present and future states even without accurate knowledge of the system model. An Unscented Kalman Filter has been designed for a planar IMU model to estimate the angular position provided by the accelerometers and the angular velocity measured by the gyroscopes present in the IMU.

The design and implementation of the PBAS prototype has been carried out by M.L.Boyd (Boyd 2010) as a part of his Master's thesis research. The project also involved performing various tests on the prototype and recording the data from the various IMU sensors. The primary objective of this work involved smoothing the data obtained by Boyd and estimating the same using a Kalman Filter. The subsequent sections explain briefly the IMU setup and the tests performed.

2.1 Design and Implementation of PBAS

Ten IMUs were constructed to record measurements on humans. Each IMU consists of an Xbee radio module and a six degree of freedom Razor board made by Sparkfun. The Xbee radio and Razor are described below. An interface board was designed to connect the Xbee to the Razor. Figure 2 shows the front, inside and back views of a finished IMU.

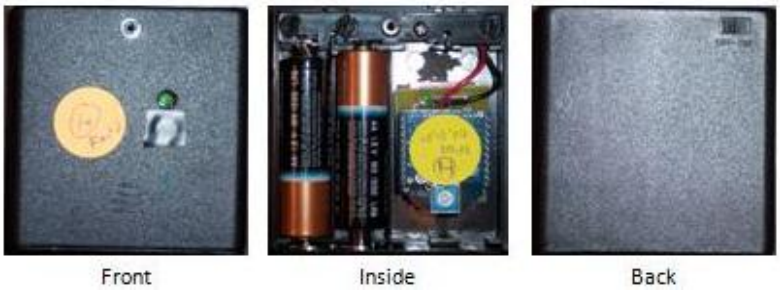

Figure 2: Finished IMU

The IMUs begin sampling when they receive a command from a Pacer, which consists of an Xbee radio module and a PIC16F688 microcontroller. The Pacer is shown in Figure 3. The PIC is coded to send the start commands out of its USART (Universal Asynchronous Receiver Transmitter) to the Xbee when either the start button is pushed or the TTL trigger sees a high-to-low transition. These commands are addressed to each specific Xbee in the network.

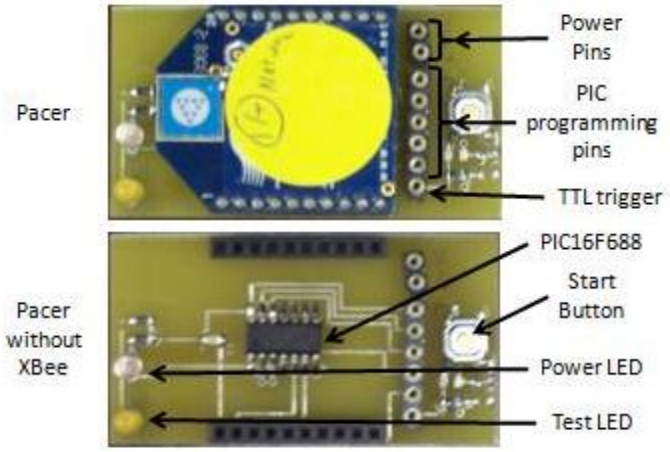

Figure 3: IMU Pacer device

To collect the data, a PKG-U device from Digi International is connected to the USB port of a laptop. The PKG-U device is used to program and communicate with an Xbee and is shown in Figure 4.

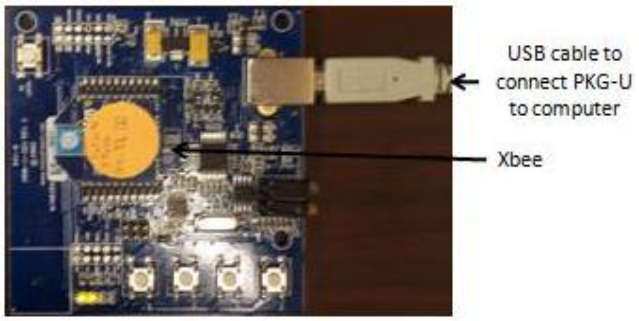

Figure 4: PKG-U device from Digi International

The 6DOF Razor from Sparkfun Electronics(2010), shown in Figure 5, is made up of a three-axis accelerometer, a two-axis gyroscope to measure pitch and roll, and a single-axis gyroscope to measure yaw. The analog outputs from the three chips are broken out to two headers where they can be easily connected to the Xbee analog-to-digital converter (ADC) pins through the interface board. The Razor includes all of the filtering resistors and capacitors and requires a three volt power supply.

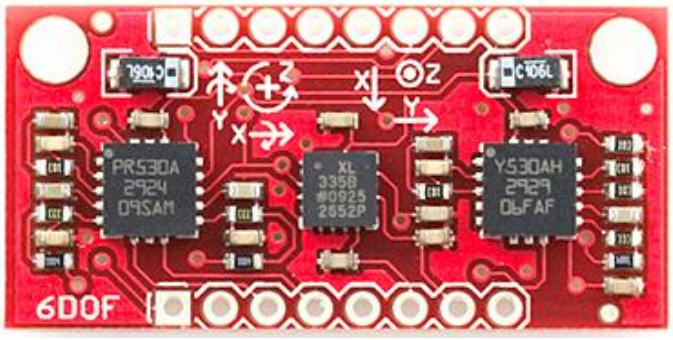

Figure 5: 6DOF Razor from Sparkfun Electronics

The Razor uses the ADXL335 accelerometer which measures accelerations within a range of ±3 g. The chip has a user selected bandwidth by selecting the value of three capacitors. The bandwidth of the chip in this study was 50 Hz. For the PBAS, it is used to measure the angular position of the body segments by measuring the static acceleration of gravity in each direction.

The two-axis gyroscope used on the Razor is the LPR530AL manufactured by ST Microelectronics. It can measure ±300 °/s with positive being the counterclockwise direction, and is capable of sensing angular rates up to a bandwidth of 140 Hz. This chip has two outputs for

each axis, a non-amplified and four times amplified signal. For the IMUs in the PBAS, the non-amplified signal is used.

The single-axis gyroscope used on the Razor is the LY530ALH also made by ST Microelectronics. It can measure up to ±300 °/s with a bandwidth of 140 Hz. This chip also has two separate outputs with the non-amplified signal being used.

The PBAS uses the Xbee OEM RF Modules made by Digi International for all radio communication. They use the 802.15.4 wireless radio protocol with a data rate of 250 kbps. The line-of-sight range for the modules is 300 ft with an operating frequency of 2.4 GHz. The modules in this application are programmed with a baud rate of 38400 Baud.

The PIC16F688 is 14-pin, 8-bit microcontroller used on the Pacer device of the PBAS. The PIC waits until it sees a high to low signal on one input pin and then sends the start sampling commands for all of the devices out of its USART to the Xbee radio for broadcasting. It is programmed to run at 38400 baud with an 8 MHz internal clock. This PIC allows for in-circuit programming by connecting five pins to the computer through a set of headers on the pacer device.

There are ten IMUs divided into two networks, with each network having its own Pacer and Collector. The networks are setup by programming the channel of each Xbee. IMUs A, B, C, D, and I form one network while IMUs E, F, G, H, and J form the other. Two networks are needed because of the amount of data coming into the collector from ten devices causing radio dropouts. The networks are setup as peer-to-peer networks on two separate channels that have been programmed into the Xbees.

Each Xbee is programmed with its own 64-bit and 16-bit source address. The 64-bit address is burned into the Xbee during manufacturing while the 16-bit is programmable. The ten 16-bit addresses are programmed with the hex number for the ASCII capital letters A through J.

When the IMU Xbees receive the start command from Pacer Xbees, they are designed to send an acknowledgment and all data back to the pacer Xbee. In order for the collector to see the data transmissions, all Xbees are configured in broadcast mode.

The Pacer device is made up of an Xbee radio module connected to a PIC16F688 microcontroller. This device is used to start the sampling of the IMUs by sending out remote AT commands. These commands are sent out of the PIC through its USART to the Xbee. This Xbee is programmed to be in API mode, which puts all data leaving or entering the Xbee into frames that define operations. The command for the IMUs to start sampling is sent to each IMU in the network. Each command has a destination address that specifies which Xbee is to receive the command. Figure 6 shows an example start command sent from the pacer device to IMU A.

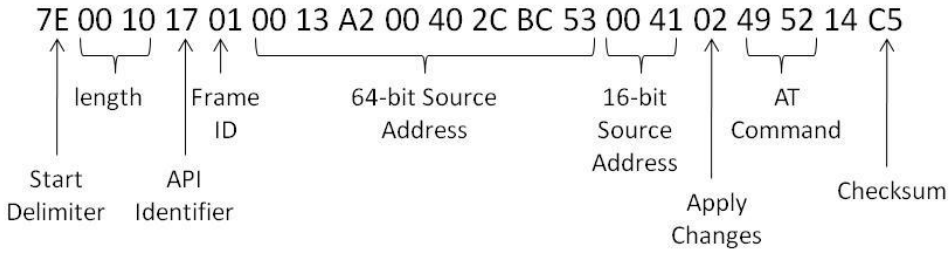

Figure 6: Remote AT command to start sampling of Xbee ADC lines

Once the start command is received by the Xbee on a IMU, an acknowledgment is sent back to the pacer followed by the data. By design of the network, the acknowledgment and data are supposed to be sent back to the device that sent the command. To allow the collector to see everything, all Xbees in the networks are set to broadcast mode. Figure 7 shows the acknowledgment frame sent from IMU A

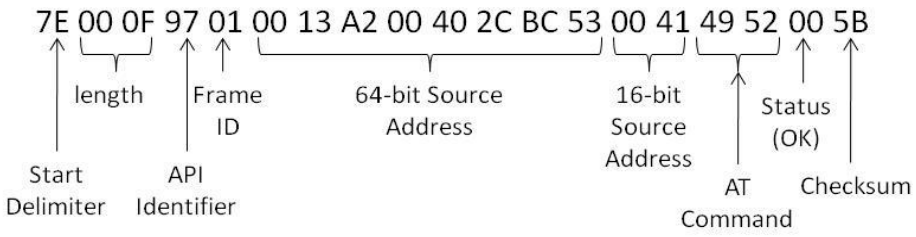

Figure 7: API acknowledgment frame

The data is received in packets by the PKG-U and read into HyperTerminal and saved in a binary file. There are two sets of data (one for each network) with five IMUs in each data set. After data collection is complete, a MATLAB program organizes the data into matrices for use.

A MATLAB program was created to combine the two data sets into four matrices, one for each signal and one for the calculated orientation angle of the IMUs. The program calls a subroutine twice, once for each data set, and combines the data. The subroutine first reads in the data from the binary file. It then cuts off the acknowledgment packets at the beginning by looking for the acknowledgment from the last IMU and keeping all data after it. This provides a start of good data where all IMUs are running. After that, the program goes through every packet to make sure it is the correct size and places all good packets into a large packet matrix where each row corresponds to a packet and each column corresponds to a byte in the packet. Table 1 shows a

sample of the packet matrix. For bad packets (too long or too short), it reads the IMU that it came from, puts its name in the correct columns of the matrix, and inserts NaN (not a number) into the rest of the columns.

Table 1: Sample packet matrix

126	0	26	131	0	66	64	0	3	38	0	2	4	1	248	1	187	2	4	1	247	1	187	2	4	1	247	1	187	162
126	0	26	131	0	67	69	0	3	38	0	2	1	1	247	1	185	2	1	1	247	1	185	2	1	1	247	1	185	172
126	0	26	131	0	68	66	0	3	38	0	1	244	1	249	1	186	1	244	1	249	1	187	1	244	1	249	1	186	206
126	0	26	131	0	73	66	0	3	38	0	2	3	1	245	1	154	2	3	1	245	1	154	2	2	1	245	1	154	7

IMU Name

D0 D1 D2 D0 D1 D2 D0 D1 D2

Sample 1 Sample 2 Sample 3

After checking all packets for the proper structure, the program checks for dropped packets. If one is missing, a dropout for that IMU is counted and NaN is inserted in the correct spot in the corresponding data vector. The dropped packets are later read by a subroutine where they are replaced by interpolated values based on the previous data.

A data vector is created for each signal along with a name vector that keeps track of the IMU data. This is done by extracting the data out of the packet matrix and placing it in its corresponding data vector. The same is done for the IMU names. These vectors are shown in Table 2.

Table 2: Sample IMU name and channel vectors

IMU Name	Channel 0	Channel 1	Channel 2	
65 (A)	513	501	448	← Sample 1
65	513	501	449	← Sample 2
65	513	501	449	← Sample 3
66 (B)	516	504	443	
66	516	503	443	
66	516	503	443	
67 (C)	513	503	441	
67	513	503	441	
67	513	503	441	
68 (D)	500	505	442	
68	500	505	443	
68	500	505	442	
73 (I)	515	501	410	
73	515	501	410	
73	514	501	410	

Finally, the program takes the data from the vectors and places it into the final data matrices. There are three matrices, one for each signal, with each column corresponding to one IMU. The data is inserted chronologically where it is sent back to the main program to be combined with the data from the second set. The fourth data matrix is for the calculated orientation angle of the IMUs and is created by taking the arctangent of each entry in the acceleration matrices.

2.2 Pendulum Test

A test was run at the NIOSH facility in Morgantown, WV to measure the x- and y-accelerations and z-angular velocity of three IMUs on a pendulum. The purpose of this test was to show the functionality of the system. A pendulum was used as pendulum motion is known analytically so the results could be validated easily. The setup for this test is shown in Figure 8. For the acceleration, the x-direction is perpendicular to the pendulum and the y-direction is along

the pendulum. The IMUs for this test were built on small breadboards to validate the circuit design and an example of one is shown in Figure 9.

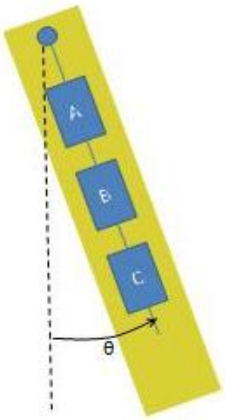

Figure 8: Pendulum test setup with breadboard IMUs

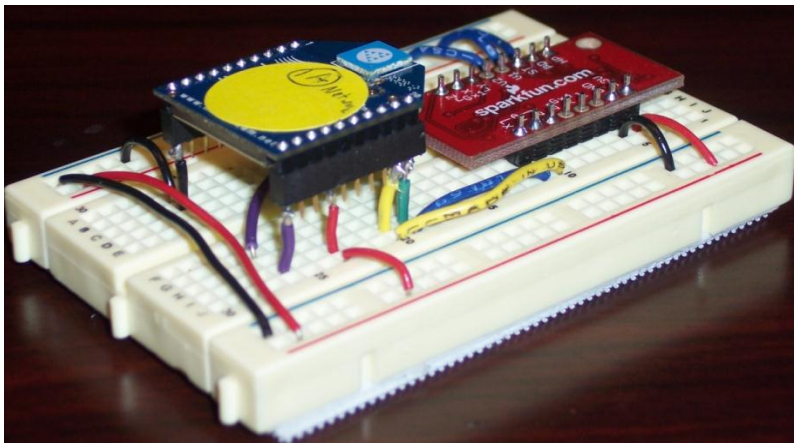

Figure 9: Breadboard IMU

The results of one run during testing are shown in Figure 10 through 12. Figure 10 shows the measured angular velocity of the three IMUs and came out as expected as the measured angular velocity should be the same for all IMUs. Figures 11and Figure 12 show the measured

acceleration in the x- and y-directions respectively. One might expect the acceleration signals to be the same, but this does not occur as explained below.

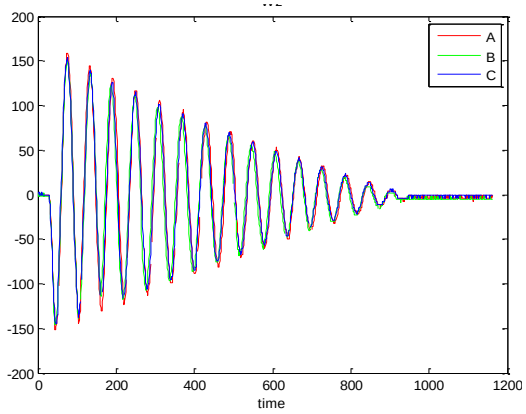

Figure 10: Angular velocity (degrees/second) vs. time for one NIOSH test

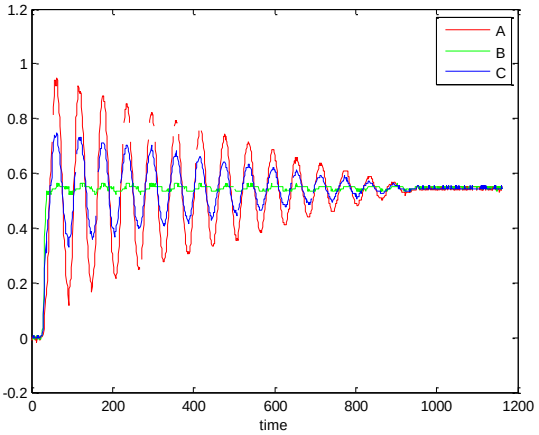

Figure 11: X-acceleration (g's) vs. time for one NIOSH test

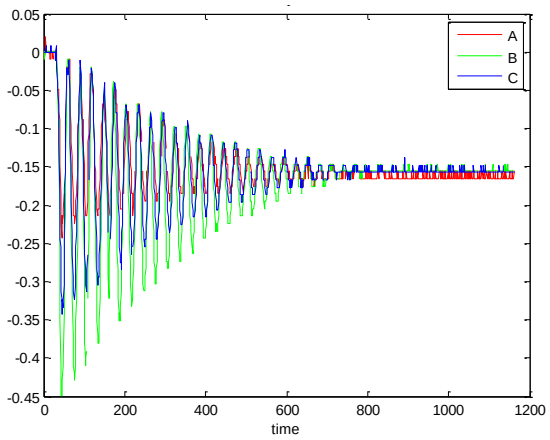

Figure 12: Y-acceleration (g's) vs. time for one NIOSH test

Separate tests were run to determine the cause of differences in the measured accelerations. These tests were run by switching the location of IMUs then measuring their signals. Four tests were run with the results and IMU locations being shown in Figures 13 through 16.

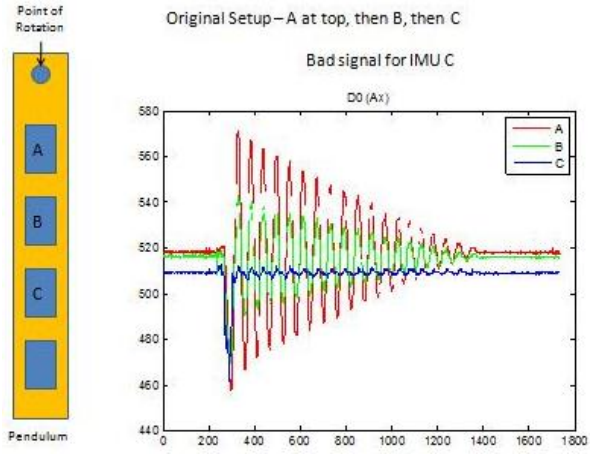

Figure 13: Test 1 setup and x-acceleration signal vs. time

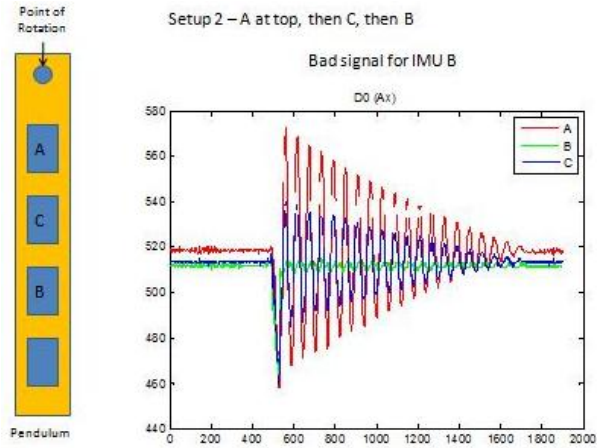

Figure 14: Test 2 setup and x-acceleration signal vs. time

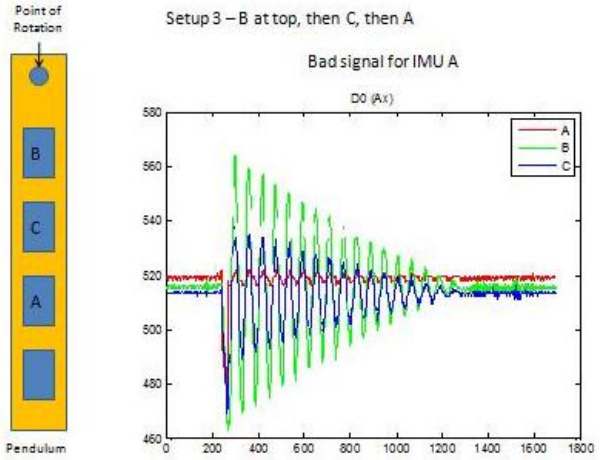

Figure 15: Test 3 setup and x-acceleration signals vs. time

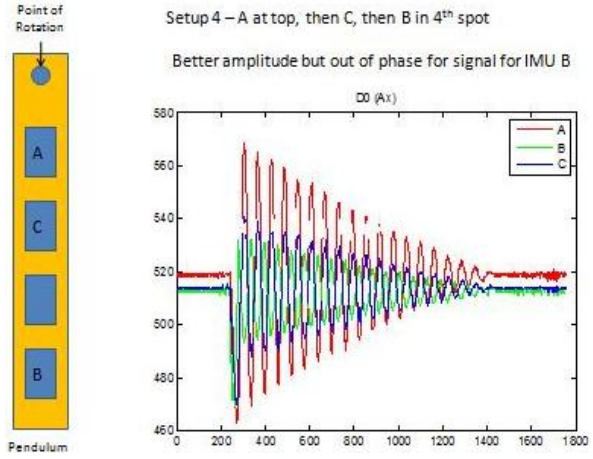

Figure 16: Test 4 setup and x-acceleration signal vs. time

From the results, the location of the IMU along the pendulum determines its measured acceleration. This phenomenon can be explained by investigating pendulum motion as described below:

θ = pendulum angle

θ_0 = initial angle of pendulum at release

f = frequency of oscillation dictated from the length of the pendulum, mass, and mass moment

t = time

$\dot{\theta}$ = angular velocity of pendulum

$\ddot{\theta}$ = angular acceleration of pendulum

A_x = local horizontal acceleration of IMU on pendulum

g = static acceleration due to gravity

r = distance from pivot of pendulum to IMU

$$\theta = \theta_0 \cos(2\pi ft)$$

$$\dot{\theta} = -\theta_0 (2\pi f)\sin(2\pi ft)$$

$$\ddot{\theta} = -\theta_0 (2\pi f)^2 \cos(2\pi ft)$$

$$A_{x,g} = g \sin\theta$$

$$A_{x,g} = g\theta$$

$$A_{x,tan} = r\ddot{\theta}$$

$$A_{x,tot} = g\theta_0 \cos(2\pi ft) - r\theta_0 (2\pi f)^2 \cos(2\pi ft)$$

$$A_{x,tot} = (g - r(2\pi f)^2)\,\theta_0 \cos(2\pi ft)$$

$$\text{For } A_{x,tot} = 0 \longrightarrow r = \frac{g}{(2\pi f)^2}$$

For f = 0.8224 Hz, calculated from the data, an IMU located 14.46 inches from the pivot will have $A_{x,tot} = 0$. This is why IMU B is out of phase in test setup 4 shown in Figure 16. A Kalman Filter will help alleviate this problem.

After production of the IMUs was complete, a second pendulum test was done with five IMUs. The IMUs were placed on the pendulum as shown in Figure 17 and tests were run in a similar fashion as in the NIOSH pendulum testing with the addition of initial calibration of the IMUs being done before this testing. The IMUs were calibrated to find the voltage offsets for the zero values of the sensors.

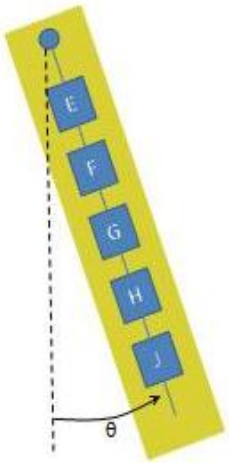

Figure 1: Pendulum test setup with finished IMUs

The results of the testing are shown in Figure 18 and Figure 19. Figure 18 shows the calculated orientation angle of the IMUs and should be the same for all IMUs as they swing together on the pendulum. This does not occur because the angle is calculated using the accelerations, with the same phenomenon occurring as in the previous testing. Figure19 shows the measured angular velocity of the IMUs. The signals are in phase and have the same amplitude as expected with discrepancies in the mean value being attributed to slight errors in the calibration. These test results shown had a dropout rate of 1.27% and a damaged packet rate of 0.96%. The results of this test helped to validate the sensor system.

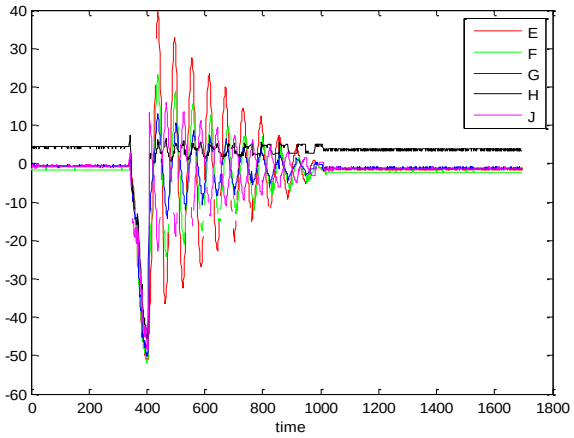

Figure 18: Calculated angle of pendulum in degrees vs. time

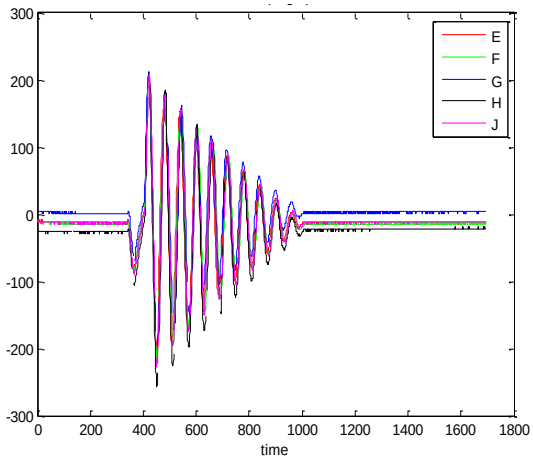

Figure 19: Measured angular velocity (deg/s) vs. time

After the data was validated for all sagittal plane IMUs, two IMUs were reprogrammed to be coronal plane IMUs. The coronal IMUs will be used on the pelvis and head. The setup for this test is shown in Figure 20.

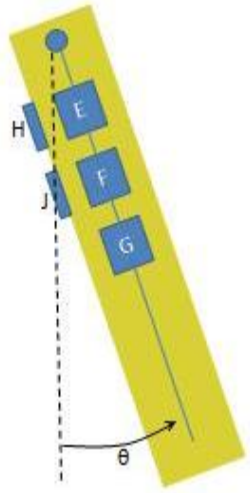

Figure 20: Pendulum test setup with sagittal and coronal IMUs

Two coronal IMUs (H and J) were programmed to send the z-acceleration, y-acceleration and y-angular velocity signals which should give the same measurements as the sagittal IMUs when aligned on the side of the pendulum as in Figure 20. The results of the test are shown in Figures 21 through 24. The z-acceleration and y-angular velocity of the coronal IMUs should be the same as the x-acceleration and z-angular velocity of the sagittal IMUs, respectively.

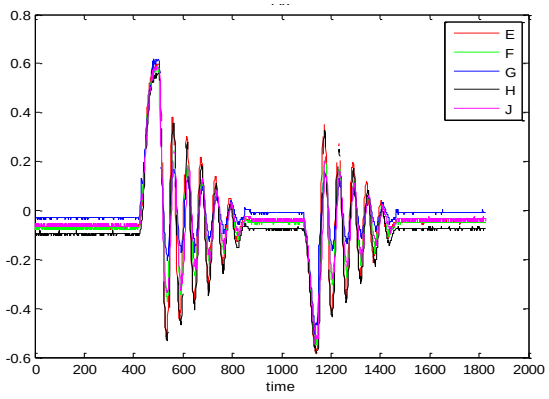

Figure 21: X-acceleration (g's) vs. time of IMUs

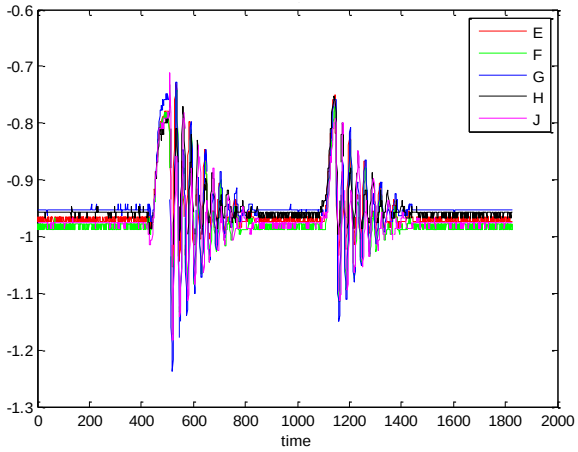

Figure 22: Y-acceleration (g's) vs. time of IMUs

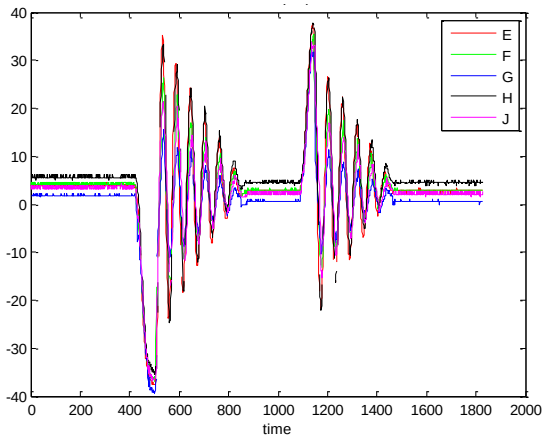

Figure 23: Measured angular position (degrees) of IMUs vs. time

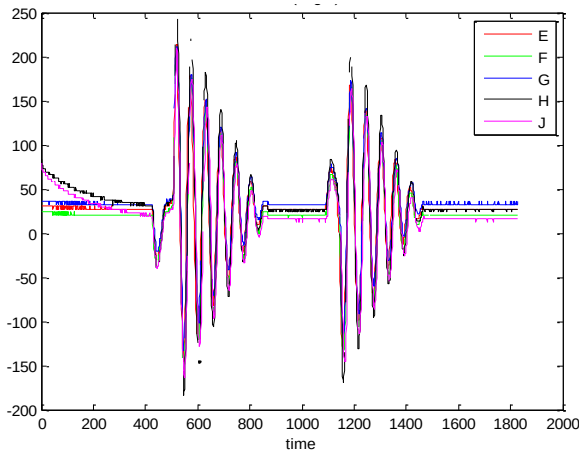

Figure 24: Angular velocity (deg/s) vs. time of IMUs

Figure 21 and Figure 22 show the x- and y-acceleration of each IMU respectively. The results of this test came out as expected as the acceleration signals matched up with each other much better than in the previous tests. This is a result of the IMU placement as the IMUs were grouped closer to the pivot of the pendulum and farther away from the point where the acceleration becomes zero. Figure 23 shows the calculated angle of each IMU by taking the inverse tangent of the ratio of x- to y-acceleration. Figure 24 shows the measured angular velocity of each IMU and came out as expected.

From the plots, the coronal IMUs match the sagittal IMUs and thus validate the use of them for the head and pelvis. The damaged packet rate for this test was 0.86% and the dropout rate was 0.61%, which are good results and show the data transmission of the network is unaffected by the use of coronal IMUs.

Chapter 3

Kalman Filter Based Estimation of PBAS Data

3.1 Introduction to Kalman Filtering

The idea of the Kalman Filter was introduced to the scientific community by R.E. Kalman in his famous paper (Kalman, 1960) describing a recursive algorithm to solve the discrete data filter problem. The important problem addressed by the Kalman Filter is of statistical nature and can be classified into three main sub-categories:

1) Prediction of a random signal
2) Separation of random signal from random noise
3) Detection of signals of known form affected by random noise

In order to address these problems, Weiner developed a linear dynamic system (Weiner Filter) accomplishing the detection, separation or prediction of random signals (Wiener, 1942.). The Weiner Filter is subject to limitations that challenged the practicality of its implementation:

1) The optimal estimator is specified by its impulse response which is not suited for practical computation and the implementation becomes increasingly difficult as the system becomes complex.
2) The generalizations and assumptions of the estimator required special modification for stationary processes, finite –memory etc.

The Kalman Filter utilizes Weiner's perspective of describing any complex system by a linear system of states represented by a set of first order difference equations excited by random white noise signals of known covariance. The state space approach for solving the Weiner problem is analogous to the famous noise-free optimal regulator problem solved by state and state

transition concepts (Kalman, 1960.). The Kalman Filter based estimation algorithm can be summarized as follows:

1) A mathematical description of the system whose states are to be estimated is obtained.
2) The equations to describe how the mean and covariance of the state propagate are implemented. These parameters are important in our case because :
   a. The mean of the state is the Kalman Filter estimate of the state.
   b. The covariance of the state is the covariance of the Kalman Filter estimate of the state
   This is called time update step of Kalman Filter algorithm.
3) Every time a measurement of outputs from the system is received, the mean and covariance of the states are updated. This is known as the measurement update step.

Consider a system represented by a linear stochastic difference equation as shown in equation 1:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \quad [1]$$

where $x \in \Re^n$ - n x 1 matrix representing the states of the system

$u \in \Re^l$ - l x 1 matrix representing the control input to the system

A is the n x n system matrix which relates the states at time k to the states at time k-1 and B is the n x l input matrix.

The output equation of the system is shown in equation 2:

$$z_k = Hx_k + v_k \quad [2]$$

where $z_k \in \Re^m$ represents the measurements from the system and H is the m x n matrix relating the states to the measurements.

The random variables w_k and v_k represent the process and measurement noise (respectively).

They are assumed to be independent of each other, white, and with Normal probability distributions as shown in Equation 3

$$P(w) = N(0, Q); \quad [3]$$

$$P(v) = N(0, R);$$

The error covariances Q and R usually change as the time progresses but these have been assumed to be constant throughout and have been calculated in the following sections.

The a-priori estimate of x is defined by $\hat{x}_k^-$ at the step k with the knowledge of the process before step k and $\hat{x}_k$ as the posteriori estimate at the time step k after receiving the measurement z_k.

The a-priori and posteriori errors are shown in Equation 4

$$e_k^- = x_k - \hat{x}_k^- \quad [4]$$

$$e_k = x_k - \hat{x}_k$$

The a-priori estimate error covariance and posteriori error covariance are shown in Equation 5.

$$P_k^- = E[e_k^- e_k^{-T}] \quad [5]$$

$$P_k = E[e_k e_k^T]$$

The idea of designing a Kalman Filter involves calculating a gain matrix K such that the posteriori estimate can be expressed as a linear combination of the a-priori estimate and the difference between actual and predicted measurements z_k and Hx_k respectively as shown in Equation 6.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - Hx_k) \quad [6]$$

The difference $z_k - Hx_k$ is called the innovation factor. It reflects the discrepancy between the estimation and actuality and the Kalman gain K is modified iteratively to reduce this

residual value.The gain K_k at every time instant k is derived by using the condition that the value of K_k minimizes the posteriori error covariance as shown in Equation 7.

$$K_k = P^-_k H^T H P^-_k(H^T + R)^{-1} \quad [7]$$

where the covariance is updated as: $P_k = (I - K_k H)P_k^-$

After each time and measurement update pair, the process is repeated with the previous a-posteriori estimates used to predict the new a-priori estimates. This recursive nature makes practical implementations much more feasible than an implementation of a Wiener Filter which is designed to operate on all of the data directly for each estimate. The Kalman Filter instead recursively conditions the current estimate based on the history of all the past measurements as shown in Figure 25.

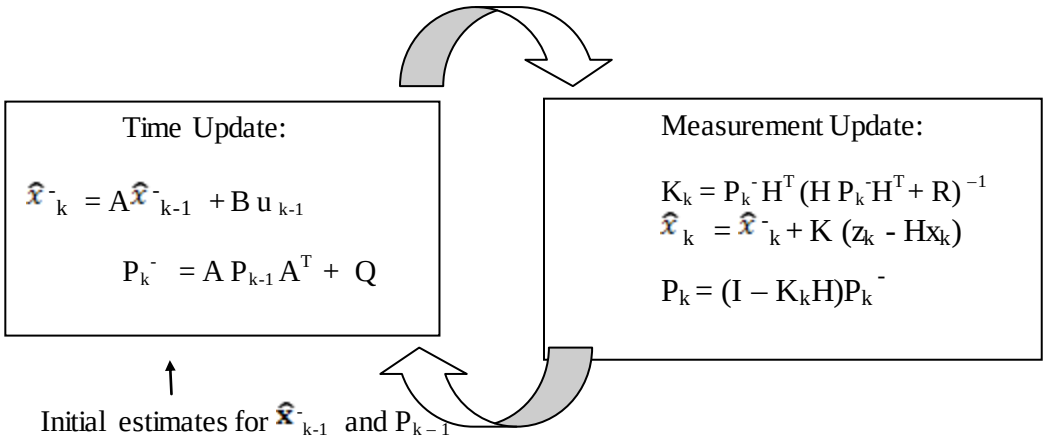

Figure 25: The flow diagram of the Kalman Filter equations

Under conditions where Q and R are almost constant, the values of P_k and K_k will stabilize quickly and remain constant. The determination of output noise covariance R is performed offline experimentally. The process covariance Q cannot be determined easily and has been assumed throughout the process. The value of process noise covariance can be tuned by system identification methods for better results of estimation.

3.2 Linear Kalman Filter

A Linear Kalman Filter (LKF) assumes the system to be linear and the noise distribution to be Gaussian. The planar IMU system model can be represented as shown in Equation 8 and 9:

$$\theta[K+1^-] = \theta[K^+] + \omega[K^+]*dt + W1[k+1] \quad [8]$$

$$\omega[K+1^-] = \omega[K-] + W2[k+1]$$

where dt is the sampling time, W1,W2 are process noise components.

$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad [9]$$

The covariance values were obtained by experiments performed on the IMUs.

The algorithm for LKF based estimation of IMU parameters is described below and the MATLAB code is presented in Appendix A.

1) The filter is initialized as : $\hat{x}_0^+ = E(x_0)$ $P0+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]$
2) The angular displacement (Θ) and velocity (ω) values are acquired
3) These values are sent to the Kalman Interpolation function described below to replace the missing values by the interpolated values.
4) The time update of $\hat{x}$ is performed according to the Equation 6.
5) The measurement update is performed by using the Θ and ω values at that instant.
6) The Kalman gain and covariance values are updated
7) The above steps are continued until all the required samples are estimated

The resulting angular displacement and velocity plots obtained after implementing the LKF of a typical IMU data sample are shown in Figures 26 and Figure 27 respectively. The IMU data was obtained from the pendulum tests described in Chapter 2. The plots show the measured data in blue and estimated data in red. The error between the measured and estimated data is very

low and cannot be seen in Figures 26 and 27. Therefore, the error plots for both of these parameters are represented in Figures 28 and 29 respectively.

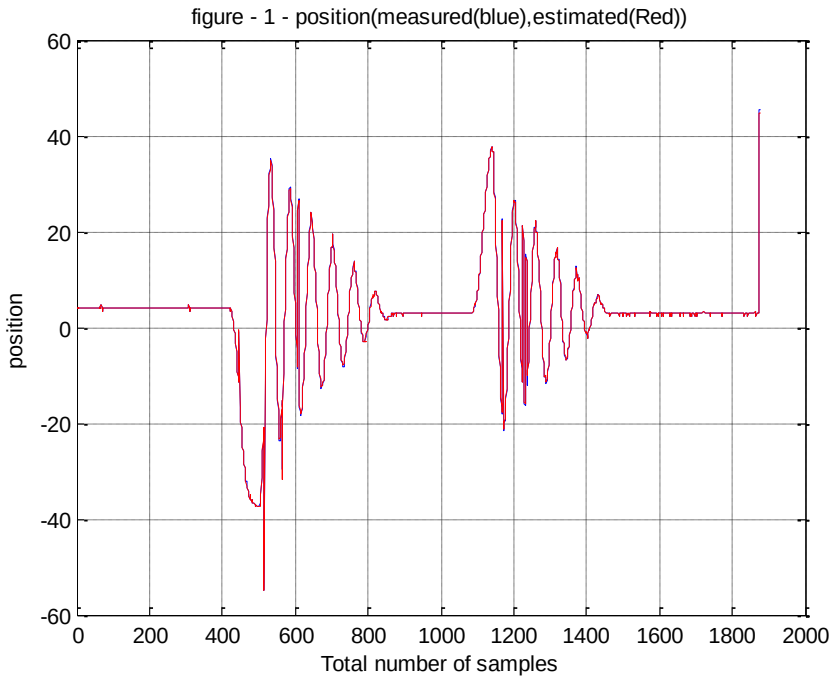

Figure 26: The plot between measured angular position and estimated angular position

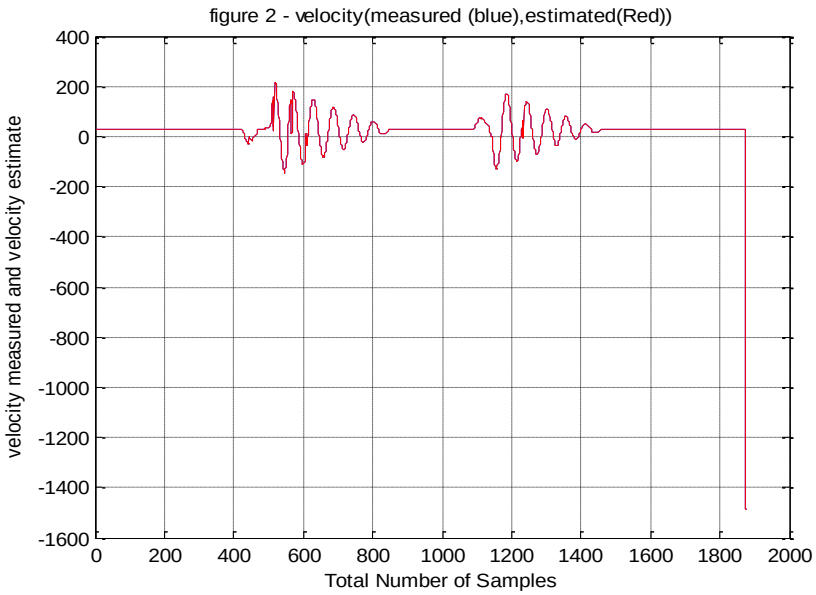

Figure 27: The plot between measured angular velocity and estimated angular velocity

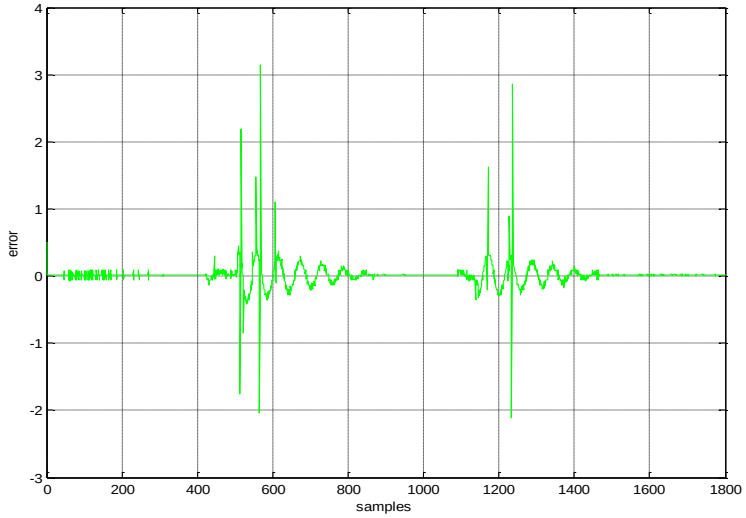

Figure 28: The plot of error between measured and estimated angular position

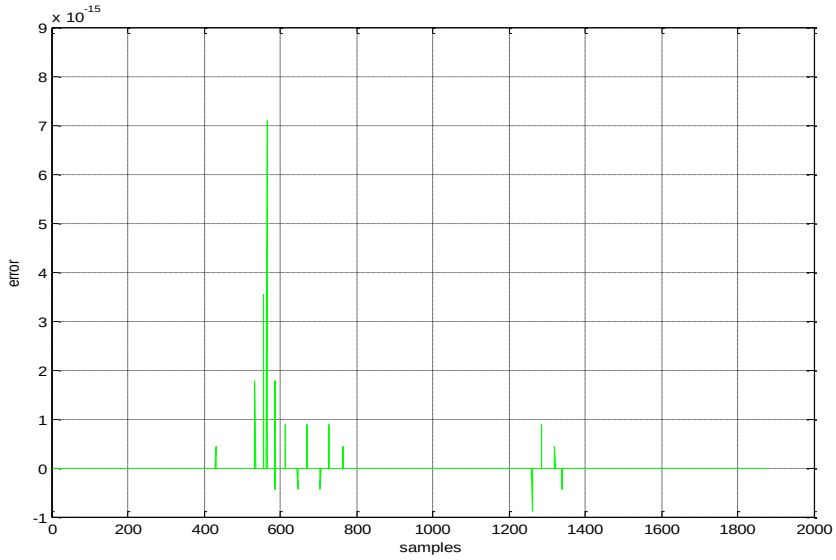

Figure 29: The plot of error between measured and estimated angular velocity

3.3 Extended Kalman Filter

The previous section explained the working of a Kalman Filter with an assumption on linearity of the system. The biomechanics model of movement is nonlinear and the parameters involved cannot be accurately estimated by the LKF. This gave rise to the idea of using an Extended Kalman Filter (EKF) for the IMU data. The following sections explain the idea behind an EKF and its implementation algorithm for the nonlinear IMU model defined in equation 10 and 11:

$$\Theta(k+1^{-}) = \tan\left(Ax(k^{+})/Ay(k^{+})\right) + W1 \qquad [10]$$

$$\boldsymbol{\omega}\,(k+1^{-}) = d(\Theta(k^{+}))/dt + W2 \qquad [11]$$

where Θ represents the angular displacement

ω represents the angular velocity

Ax and Ay represent the horizontal and vertical components of acceleration.

The derivation of EKF was based on linearizing the nonlinear system around a nominal state trajectory. This nominal state trajectory can be determined by the Kalman Filter estimates of the state of a system. Therefore, the nonlinear system was linearized around the Kalman Filter estimate and the estimate is based on the linearized system. This idea was originally published by Stanley Schmidt (Schmidt, 1966.) in order to apply the Kalman Filter for nonlinear spacecraft navigation problems. The EKF equations are derived as follows [Simon, 2006.]

Given a non-linear system model as shown in Equation 12:

$$x_k = f_{k-1}(x_{k-1},u_{k-1},w_{k-1}); \quad [12]$$

$$y_k = h_k(x_k,v_k);$$

$$w_k \sim (0,Q_k);$$

$$v_k \sim (0,R_k);$$

A Taylor series expansion of the state equation is performed around $x_{k-1} = \hat{x}_{k-1}^{+}$ and $w_{k-1} = 0$ as shown in Equation 13.

$$x_k = f_{k-1}(\hat{x}_{k-1}^{+}, u_{k-1},0) + (\partial f_{k-1}/\partial x)(x_{k-1} - \hat{x}_{k-1}^{+}) + (\partial f_{k-1}/\partial \omega)\omega_{k-1} \quad [13]$$

$$= f_{k-1}(\hat{x}_{k-1}^{+}, u_{k-1},0) + F_{k-1}((x_{k-1}-\hat{x}_{k-1})^{+} L_{k-1}\omega_{k-1} \quad [1]$$

$$= F_{k-1}x_{k-1} + [f_{k-1}(\hat{x}_{k-1}^{+}, u_{k-1},0) - F_{k-1}\hat{x}_{k-1}^{+}] + L_{k-1}\omega_{k-1}$$

$$= F_{k-1}x_{k-1} + u_{k-1} + w_{k-1}$$

F_{k-1} and L_{k-1} are defined in the equation [13]. The known signal u_k and the noise signal w_k are defined in Equation 14.

$$u_k = f_k(x_{k-1}^{+}, u_{k-1},0) + F_{k-1}(x_{k-1} - x_{k-1}^{+}) + L_{k-1}w_{k-1} \quad [14]$$

$$w_k = (0, L_kQ_kL_k^{T})$$

The linearization is performed around $x_k = \hat{x}_k^{-}$ and $v_k = 0$ to obtain Equation 15.

$$y_k = h_k(\hat{x}_k^{-}, 0) - H_k\hat{x}_k^{-} \quad [15]$$

$v_k \sim (0. M_k R_k M^T_k)$

These equations represent a system of linear state space system and its corresponding measurement and can be estimated using standard Kalman Filter forms shown in equation 16.

$$P^-_k = F_{k-1} P^+_{k-1} F^T_{k-1} + L_{k-1} Q_{k-1} L^T_{k-1} \quad [16]$$

$$K_k = P^-_k H^T_k (H_k P^-_k H^T_k + M_k R_k M^T_k)^{-1}$$

$$\hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, 0)$$

$$z_k = h_k(\hat{x}_k^-, 0) - H_k \hat{x}_k^-$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^- - z_k)$$

$$= \hat{x}_k^- + K_k[y_k - h_k(\hat{x}_k^-, 0)]$$

$$P^+_k = (I - K_k H_k) P^-_k$$

The EKF algorithm is described below and MATLAB code is provided in Appendix B.

1) The nonlinear system and measurement equations are defined as follows:

   $x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})$;

   $y_k = h_k(x_k, v_k)$;

   $w_k \sim (0, Q_k)$;

   $v_k \sim (0, R_k)$;

2) Initialize the filter as follows.

   $\hat{x}_0^+ = E(x_0)$

   $P^+_0 = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$

3) For k = 1,2,…, do the following

   (a) Perform the following time-update equations:

   $P^-_k = F_{k-1} P^+_{k-1} F^T_{k-1} + L_{k-1} Q_{k-1} L^T_{k-1}$

$\hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, 0)$

Where the partial derivative matrices F_{k-1}, L_{k-1} are defined as

$F_{k-1} = \partial f_{k-1} / \partial x$ at $x = \hat{x}_{k-1}^+$

$L_{k-1} = \partial f_{k-1} / \partial \omega$ at $x = \hat{x}_{k-1}^+$.

(b) Perform the measurement update by initializing the iterated EKF estimate to the standard EKF estimate:

$\hat{x}^+_{k,0} = \hat{x}_k^-$

$P^+_{k,0} = P^-_k$.

For i = 0,1,…,N, evaluate the following equations (where N is the desired number of samples).

$H_{k,I} = \partial h / \partial x$ at $\hat{x}_{k,i}^+$

$M_{k,i} = \partial h / \partial v$ at $\hat{x}_{k,i}^+$

$K_{k,i} = P^-_k H^T_{k,i} (H_{k,i} P^-_k H^T_{k,i} + M_{k,i} R_k M^T_{k,i})^{-1}$

$P^+_{k,i+1} = (I - K_{k,i} H_{k,i}) P^-_k$

$\hat{x}^+_{k,i+1} = \hat{x}_k^- + K_k[y_k - h(\hat{x}_{k,i}^+) H_{k,i}(\hat{x}_k^- - \hat{x}_{k,i}^+)]$

(c) The final a posteriori state estimate and estimation error covariance are given as:

$\hat{x}^+_k = \hat{x}^+_{k,N+1}$

$\mathbf{P}^+_k = \mathbf{P}^+_{k,N+1}$

The resulting angular displacement and velocity plots obtained after implementing the EKF of the IMU data are shown in Figures 30 and 31 respectively. The IMU data was obtained from the pendulum tests described in Chapter 2. The plots show the measured data in blue and estimated data in red. The instantaneous error values between the measured and estimated data are very small and cannot be inferred from Figures 30 and 31. Therefore, the error plots for both these parameters are presented in Figures 32 and 33 respectively.

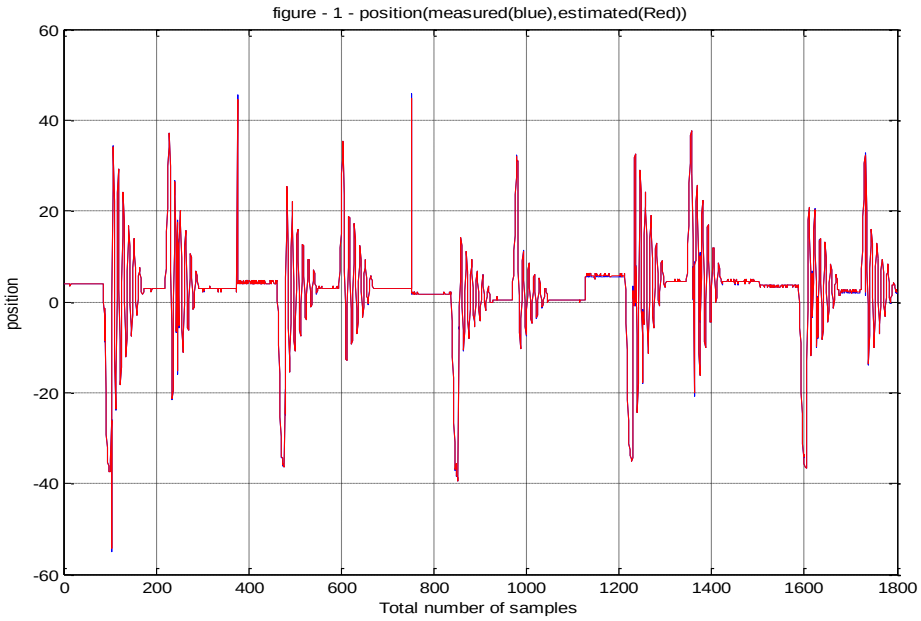

Figure 30: The plot of measured angular position and estimated angular position using EKF

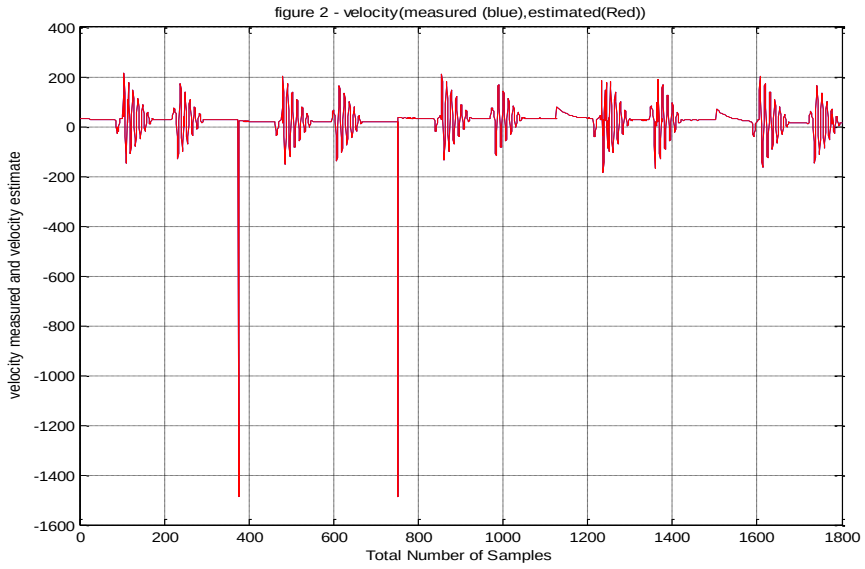

Figure 31: The plot of measured angular velocity and estimated angular velocity using EKF

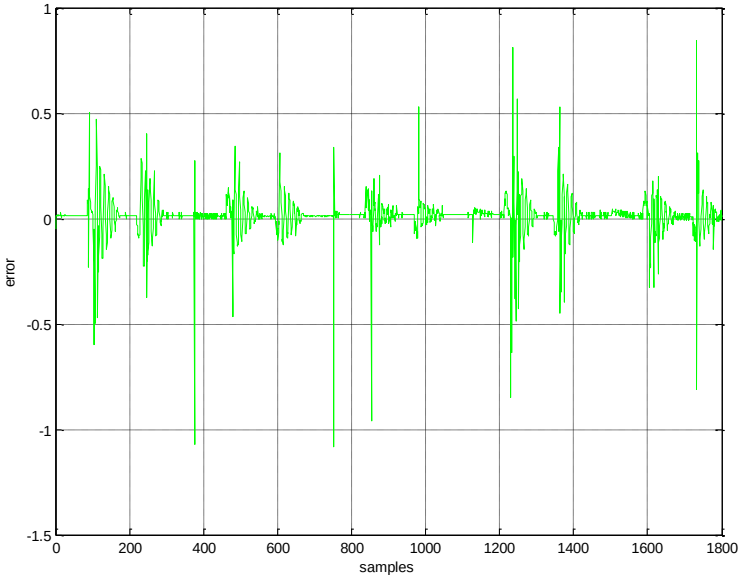

Figure 32: The plot of error between measured and estimated angular position

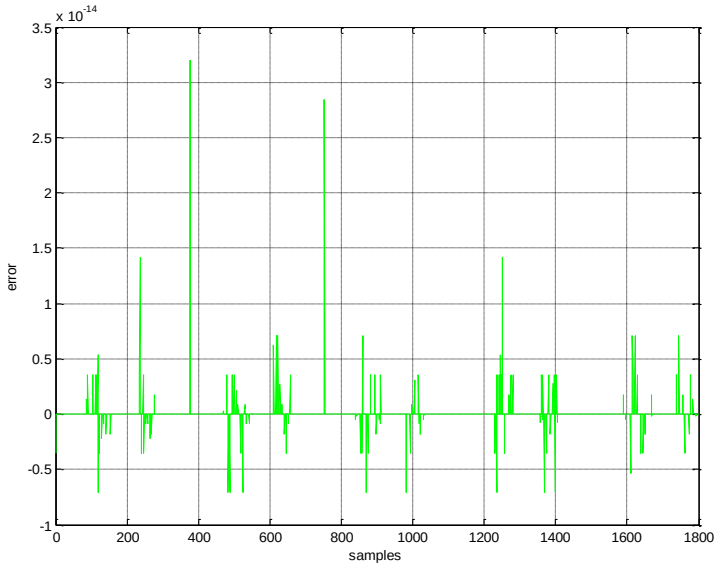

Figure 33: The plot of error between measured and estimated angular velocity

3.4 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) addresses the inaccuracies of linearity assumption in the system even better than the EKF filter. An important operation performed, in general, in the Kalman Filter is the propagation of a Gaussian random variable (GRV) through the system dynamics. In the EKF, the state distribution is approximated by a GRV, which is then propagated analytically through the first-order linearization of the nonlinear system. This can introduce large errors in the true posterior mean and covariance of the transformed GRV, which may lead to sub-optimal performance and sometimes divergence of the filter. The UKF addresses this problem by using a deterministic sampling approach. The state distribution is again approximated by a GRV, but is now represented using a minimal set of carefully chosen sample points. These sample points completely capture the true mean and covariance of the GRV, and when propagated through the true non-linear system, captures the posterior mean and covariance accurately to the 3rd order Taylor series expansion for any nonlinearity. The EKF, in contrast, only achieves first-order accuracy. Remarkably, the computational complexity of the UKF is the same order as that of the EKF.

The UKF algorithm is described below and the corresponding MATLAB code is provided in Appendix C.

Given a n-state discrete-time nonlinear system

$X_{k+1} = f(x_k,u_k,t_k) + w_k$

$Y_k = h(x_k,t_k) + v_k$

$W_k = N(0,Q_k)$

$V_k = N(0,R_k)$

1. The UKF is initialized as $\hat{x}_0^+ = E(x_0)$ $P0+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]$.

2. The time update is performed by the following tests

i. Choose sigma points $\hat{x}^{(i)}_{k-1} = \hat{x}^+_{k-1} + \tilde{x}^{(i)}$, i = 1,…,2n

ii. $\tilde{x}^{(i)} = (\sqrt{(n\, P_{k-1}^+)}_i^T$, i = 1,…,n

iii. $\tilde{x}^{(n+i)} = -(\sqrt{(n\, P_{k-1}^+)}_i^T$, i = 1,…,n

iv. Use the known nonlinear system to transform the sigma points into $\hat{x}^{(i)}_k$ vectors with appropriate changes .

$$\hat{x}^{(i)}_k = f(\hat{x}^{(i)}_{k-1}, u_k, t_k)$$

v. Combine the $\hat{x}^{(i)}_k$ vectors to obtain the a-priori state estimate at time k.

$$\hat{x}^-_k = 1/2n \sum_{i=1}^{2n} \hat{x}^{(i)}_k$$

vi. The a-priori error covariance is calculated

$$P_k^- = 1/2n \sum_{i=1}^{2n} (\hat{x}^{(i)}_k - \hat{x}^-_k)(\hat{x}^{(i)}_k - \hat{x}^-_k)^T + Q_{k-1}$$

3. Now that the time update has been performed, the measurement update is performed as follows:

a) $\hat{x}^{(i)}_k = \hat{x}^-_k + \tilde{x}^{(i)}$.

$\tilde{x}^{(i)} = (\sqrt{(n\, P_k^-)}_i^T$ i = 1,…,n

$\tilde{x}^{(n+i)} = -(\sqrt{(n\, P_k^-)}_i^T$ i = 1,…,n

This step can be skipped if desired. We can instead sigma points from the time update equation.

b) Use the nonlinear measurement equation to transform sigma points into:

$$\hat{Y}^{(i)}_k = h(\hat{x}^{(i)}_k, t_k)$$

c) Combine the $\hat{Y}^{(i)}_k$ to obtain the predicted measurement at time k.

$$\hat{Y}_k = 1/2n \sum_{i=1}^{2n} \hat{Y}^{(i)}_k$$

d) Estimate the covariance of the predicted measurement.

$$P_y = 1/2n \sum_{i=1}^{2n} (\hat{Y}^{(i)}_k - \hat{Y}_k)(\hat{Y}^{(i)}_k - \hat{Y}_k)^T + R_k$$

e) Estimate the cross covariance between $\hat{x}^-_k$ and $\hat{Y}_k$ based on the equation

$$P_{xy} = 1/2n \sum_{i=1}^{2n} (\hat{x}^{(i)}_k - \hat{x}^-_k)(\hat{Y}^{(i)}_k - \hat{Y}_k)^T$$

f) The measure update is performed using the normal Kalman Filter equations as shown

$$K_k = P_{xy} P_y^{-1}; \hat{x}^+_k = \hat{x}^-_k + K_k(Y_k - \hat{Y}_k); P_k^+ = P_k^- - K_k P_y K_k^T$$

The algorithm is iterated and the estimated values of x are plotted with actual values.

The resulting angular displacement and velocity plots obtained after implementing the UKF of the IMU data are shown in Figures 34 and 35 respectively. The IMU data was obtained from the pendulum tests described in Chapter 2. The plots show the measured data in blue and estimated data in red. The instantaneous error values between the measured and estimated data are very small and cannot be inferred from Figures 34 and 35. Therefore, the error plots for both these parameters are presented in Figures 36 and 37 respectively.

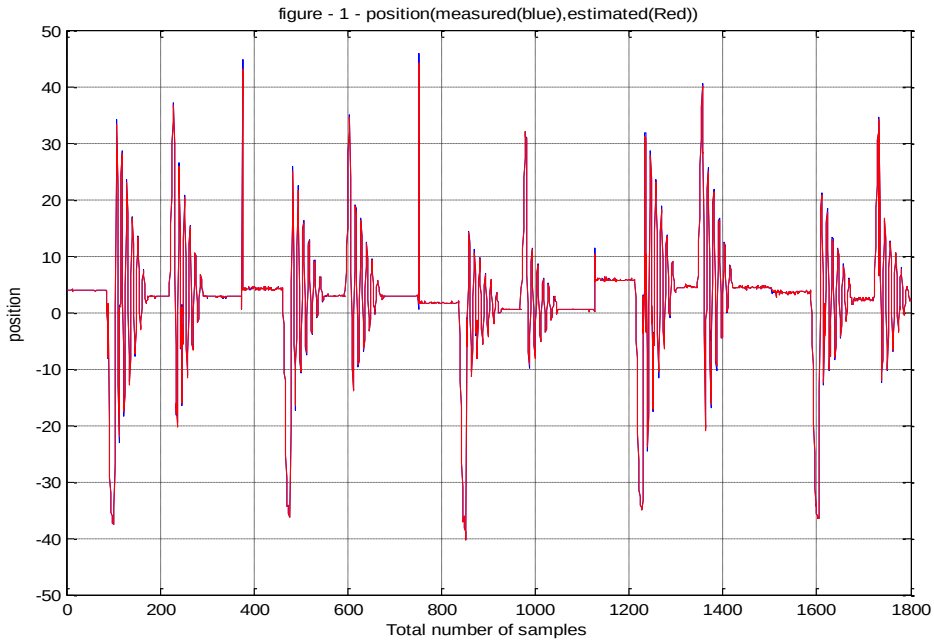

Figure 34: The plot of measured angular position and estimated angular position using UKF

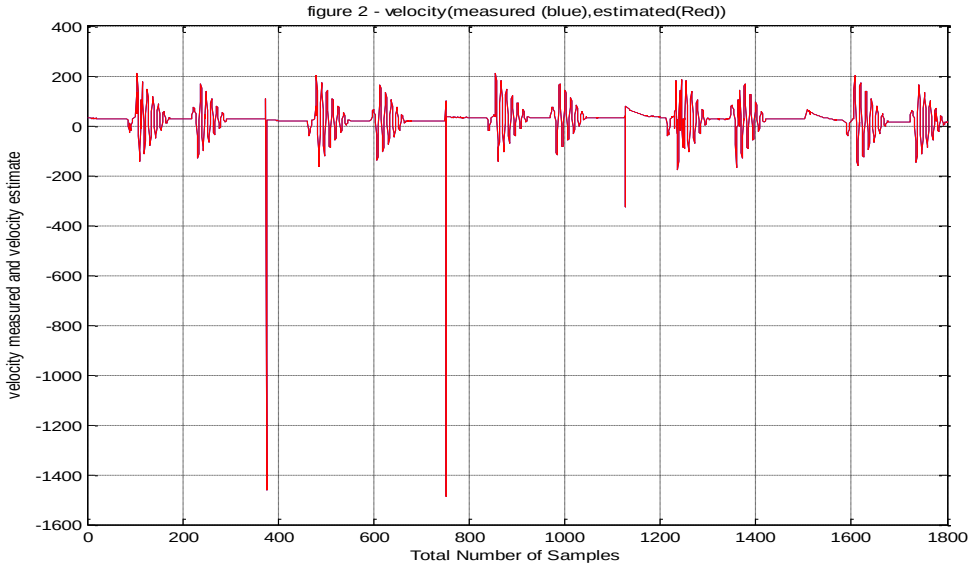

Figure 35: The plot of measured angular position and estimated angular position using UKF

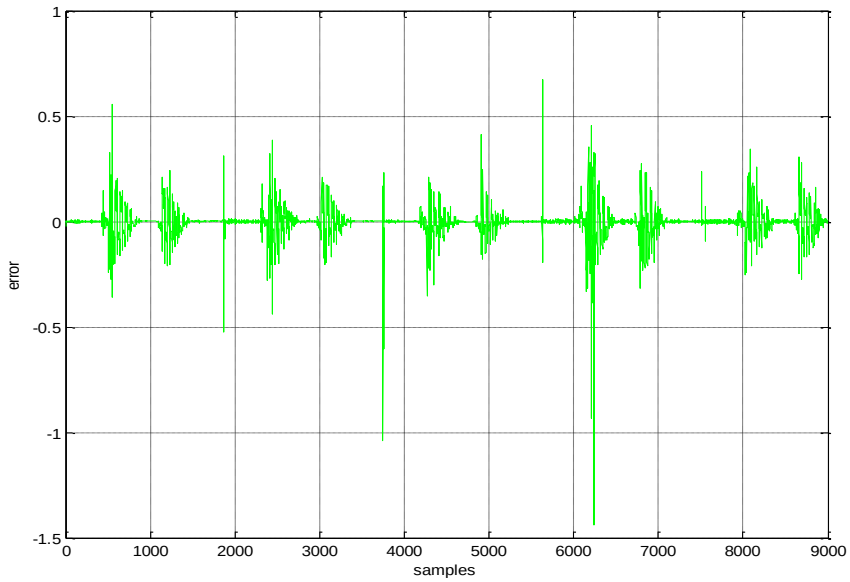

Figure 36: The plot of error between measured and estimated angular position

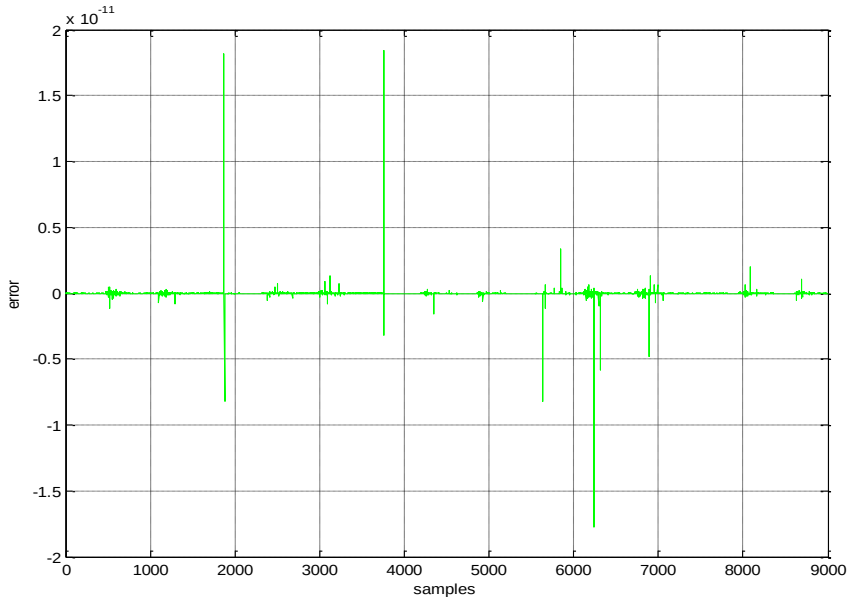

Figure 37: The plot of error between measured and estimated angular velocity

3.5 Interpolation Algorithm for IMU Dropouts:

The data obtained from IMUs is sometimes lost due to radio packet collisions. These IMU dropouts were represented by 'NaN' symbols in the original MATLAB program. Figure 38 shows a sample data set which has 'NaN' due to IMU dropouts. These points impose a challenge to the Kalman Filter estimation algorithm as the filter fails to progress after encountering the first 'NaN' in the data set.

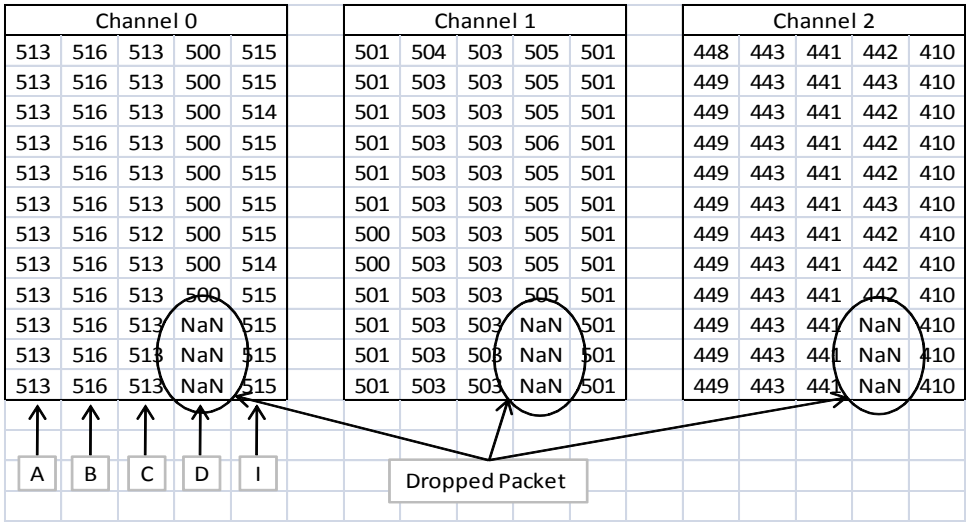

Figure 38: The data received from the IMU where dropouts have been denoted by 'NaN'.

One of the major challenges in the project involved pre-processing the data obtained to replace the dropped values and interpolate them with reasonably accurate values. Two different methods were adopted to address this problem:

i) Polynomial interpolation
ii) Forward-backward smoothing algorithm

The polynomial interpolation method makes use of samples before and after the IMU dropout. These samples are fit using MATLAB *polyfit* and *polyval* and the missing values are

replaced. In addition to the sample number and the corresponding value, the function also takes in the degree of polynomial to be used for interpolation as its input. Several different degrees of polynomials were tested to find the optimal order. Though this method has the advantage of an easy implementation, its accuracy and the optimal polynomial order required are subject to change with different data sets. This necessitates the need for a more accurate and rigorous approach towards solving the dropout interpolation problem.

The forward backward smoothing algorithm [Fraser et al, 1969] can be used to estimate the state x (k=m) based on measurements from k = 1 to k = N where N > m. The algorithm provides two estimates of x_m. The first estimate $\hat{x}_f$, is based on the standard Kalman Filter that operates from k=1 to k=m. The second estimate $\hat{x}_b$ is based on a Kalman Filter that runs backward in time from k=N back to k=m. The forward backward approach to smoothing combines the two estimates to form the optimal smoothed estimate. The forward estimate $\hat{x}_f$, the backward estimate $\hat{x}_b$ of the state and the smoothed estimate $\hat{x}$ can determined by equation 17

$$\hat{x} = K_f \hat{x}_f + K_b \hat{x}_b \qquad [17]$$

The covariance of the estimate can be found by the equation 18

$$P = (P_f^{-1} + P_b^{-1})^{-1} \qquad [18]$$

The interpolation algorithm is performed on the linear model of the system for the simplicity of implementation. The system model has already been described by equation 9. The algorithm can be summarized below and the MATLAB code is provided in Appendix D:

1) Initialize the forward Filter as:
   $\hat{x}^+_{f0} = E(x_0)$
   $P+_{f0} = E((x_0 - \hat{x}^+_{f0})(x_0 - \hat{x}^+_{f0})^T$.

2) For k = 1,..,m perform the following:
   $P^-_{fk} = F_{k-1} P^+_{f,k-1} F^T_{k-1} + Q_{k-1}$
   $K_{fk} = P^-_{fk} H^T_k R_k^{-1}$
   $\hat{x}^-_{fk} = F_{k-1} \hat{x}^+_{f,k-1}$

$$\hat{x}^{+}_{fk} = \hat{x}^{-}_{fk} + K_{fk}(y_k - H_k\hat{x}^{-}_{fk})$$

$$P^{+}_{fk} = (I - K_{fk}H_k)P^{-}_{fk}$$

The model for backward estimation is given by equation 19

$$x_{k-1} = F^{-1}_{k-1}\, x_k + F^{-1}_{k-1}\, w_{k-1+} \qquad [19]$$

$$= F^{-1}_{k-1}\, x_{k+} \, w_{b,k-1}$$

$$y_k = H_k\, x_k + v_k$$

$$w_{b,k} \sim (0, F_k^{-1} Q_k F^{-T}_k)$$

$$v_k \sim (0, R_k)$$

The algorithm steps can be repeated to obtain $\hat{x}^{+}_{bk}$ and P^{+}_{bk}. Equations 17 and 18 can be used to obtain the final estimate and covariance $\hat{x}^{+}_{k}$, P respectively. The algorithm has been implemented on the angular position data from the IMU. The figure 39 represents the IMU data with dropouts before interpolation and figure 40 represents the data after interpolation.

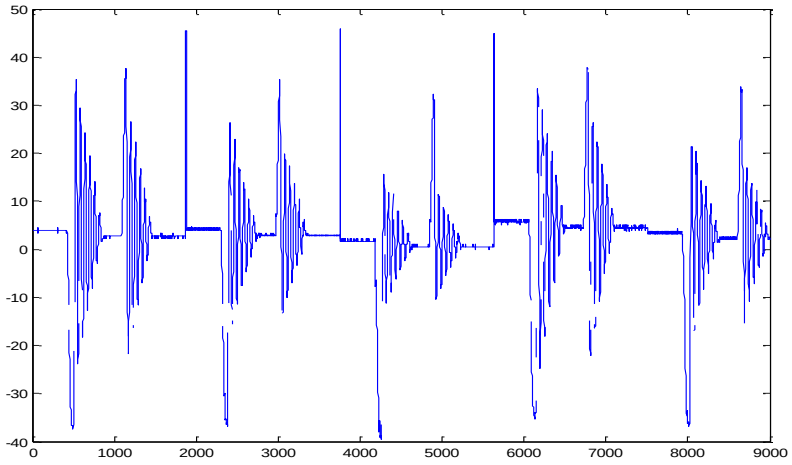

Figure 39: The plot of angular position before interpolation.

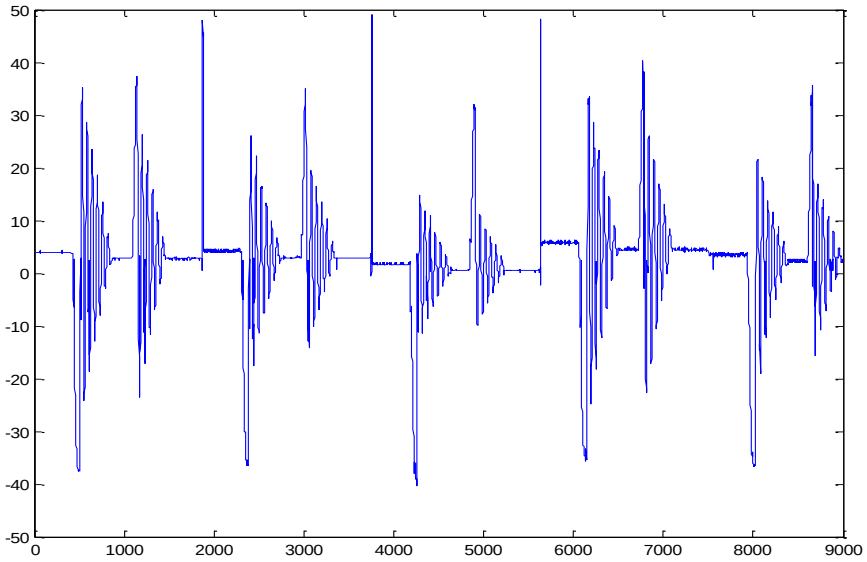

Figure 40: The plot of angular position after interpolation.

Chapter 4

Results and Conclusions

The PBAS system was setup and the data from IMUs were recorded while performing the pendulum tests. The LKF was implemented to process the IMU data based on the assumption of linearity of the model. The filter recorded a root mean square estimation error of 0.7503 radians for angular position and 4.65×10^{-16} radians per second for angular velocity. As the next stage of data processing, an EKF was implemented on the non-linear model of the system. This algorithm recorded a root mean square estimation error of 0.0991 radians for angular position and 1.45×10^{-15} radians per second for angular velocity. A UKF was then implemented to process the same set of data and was found to record a root mean square estimation error of 0.0554 radians for angular position and 3.71×10^{-13} radians per second for angular velocity. The values of Root Mean Square Errors (RMSE) are shown in Table 3. In all these implementations, a forward-backward smoothing algorithm was used to interpolate the data lost due to dropouts.

Table 3: The RMSE values for different Kalman Filtering algorithms

Algorithm	RMSE of Θ (radians)	RMSE of ω (radians/sec)
LKF	0.7503	4.65×10^{-16}
EKF	0.0991	1.45×10^{-15}
UKF	0.0554	3.71×10^{-13}

Based on the RMSE values, the UKF algorithm is the optimal solution to the problem of smoothing the IMU data without compromising on the computational complexity. The filter can also be implemented for the biomechanical model developed by Frievalds et al (1984) by modifying the nonlinear state equations accordingly. Therefore, the Unscented Kalman Filter can be used for estimation and smoothing of IMU data obtained while performing lifting tasks.

REFERENCES

Bernard BP, Musculoskeletal Disorders and Workplace Factors - A Critical Review of Epidemiologic Evidence for Work-Related Musculoskeletal Disorders of the Neck, Upper Extremity, and Low Back, *Centers for Disease Control and Prevention, National Institute for Occupational Safety and Health*, 1997

Boyd ML, Portable Biomechanical Assessment Suite (PBAS), Master's Thesis in Mechanical Engineering, The Pennsylvania State University, 2010

Fraser D, Potter J, The optimum linear smoother as a combination of two optimum linear filters, *Automatic Control, IEEE Transactions* , vol.14, no.4, pp. 387- 390, 1969

Freivalds A, Chaffin DB, Garg A, Lee KS, A dynamic evaluation of lifting maximum acceptable loads. *Journal of Biomechanics*, vol.17, no.4, pp. 251-262, 1984

Kalman RE, A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 1960

Schmidt SF, Advances in control systems, in Applications of State Space Methods to Navigation Problems, C. T. Leondes, Ed. New York: Academic, 1966

Simon D, Optimal State Estimation: Kalman, H∞, and Nonlinear Approaches, John Wiley& Sons, Inc., Hoboken, NJ, 2006

Sparkfun Electronics. IMU 6DOF Razor – Ultra – Thin IMU [Internet]. Boulder (CO): Sparkfun Electronics; [cited 2010 Jan 12]
Available from: http://www.sparkfun.com/commerce/product_info.php?products_id=9431.

Wan EA, Van Der Merwe R, The unscented Kalman Filter for nonlinear estimation. *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000, AS-SPCC, IEEE 2000* , pp.153-158, 2000

Waters TR, Putz-Anderson V, Garg A, Fine LJ, Revised NIOSH equation for the design and evaluation of manual lifting tasks, *Ergonomics*, vol.36, no.7, pp.749-776, 1993

Wiener N, The interpolation, extrapolation and smoothing of stationary time series, *Report of the Services 19*, Research Project DIC-6037 MIT, 1942

Appendix - A

Linear Kalman Filter code

```
% function linearkalman(duration,  dt)
clc;
duration = input('Enter duration: ');
dt = input('Enter interval: ');
imuno = input('Enter the no of imus used:');
ch = input('enter the IMU used : ');
order = input('enter the order of curve fitting : ');


a = [1 dt;0 1];
c = [1 0;0 1];

xhat = [0;0];
sz = [0.0263  0;0 0];
sw = eye(2);
 pr = sw;
 poshat = [];
 posmeas = [];
 velhat = [];
 velmeas = [];
thetaindex = [];
X = [];Y = [];Y1 = [];
for t = ch:duration;
   if isnan(theta(t))== 1;
      thetaindex = [thetaindex;t];
   end;
end;
thetafit = theta;Wzfit = Wz;
for i = 1:length(thetaindex);
   count = 0;
   j = thetaindex(i)-imuno;
   while (count <= 50);
      if isnan(thetafit(j))~= 1;
         count = count +1;
         X = [X;j];
         Y = [Y;thetafit(j)];
         Y1 =[Y1;Wzfit(j)];
         j = j-imuno;
      else
         j = j - imuno;
      end;
   end;
   count = 0;

    while (count <= 50);
```

```
        if isnan(thetafit(j))~=  1;
            count = count +1;
            X = [X;j];
            Y = [Y;thetafit(j)];
            Y1  = [Y1;Wzfit(j)];
            j = j+imuno;
        else
            j = j+imuno;
        end;
    end;
    p = polyfit(X,Y,order);
    p1 = polyfit(X,Y1,order);
    thetafit(thetaindex(i))  = polyval(p,thetaindex(i));
    Wzfit(thetaindex(i))  = polyval(p1,thetaindex(i));

end;
for t = ch:imuno:length(thetafit),
    y = [thetafit(t);Wzfit(t)];
     xhat = a*xhat;
     innvect = y - c*xhat;
     s = c*pr*c' + sz;
     k = a*pr*c'*inv(s);
     xhat = xhat  + k*innvect;
     pr = a*pr*a'- k*c*pr*a'+sw;
     posmeas =[posmeas;y(1)];
     poshat = [poshat;xhat(1)];
     velmeas =[velmeas;y(2)];
     velhat = [velhat;xhat(2)];
end
 close all;
 t = ch:imuno:length(thetafit);
 figure;
 plot(t,posmeas,'Bl',t,poshat,'R');
 grid;
 xlabel('Total  number of samples');
 ylabel('position');
 title ('figure  - 1 - position(measured(blue),estimated(Red))');
 figure;
 plot(t,poshat-posmeas,'G');
 grid;
 xlabel('samples');
 ylabel('error');
 t = ch:imuno:length(Wzfit);
 figure;
 plot(t,velmeas,'Bl',t,velhat,'R');
 grid;
 xlabel('Total  Number of Samples');
 ylabel('velocity  measured and velocity estimate');
 title('figure  2 - velocity(measured (blue),estimated(Red))')
  figure;
 plot(t,velhat-velmeas,'G');
 grid;
 xlabel('samples');
 ylabel('error');
```

Appendix - B

Extended Kalman Filter code

```
clc;
duration = input('Enter duration: ');
dt = input('Enter interval: ');
imuno = input('Enter the no of imus used:');
ch = input('enter the IMU used : ');
posmeas = [];poshat = [];velmeas = [];velhat = [];thetaindex = [];thetafit1 = [];Wzfit1 = [];X = [];Y = [];Y1
= []; temp = zeros(3,3);temp1 = zeros(2,2); temp2 = zeros(3,2);
Xfd = [0;0];Xbd = [0;0]; Pfd=[];Pbd =[];sfd = [];sbd = [];P =[];Xsmooth = [];K = []; innvect = [];sw1 = [];
xhat = [0;0;0];xsig = [];ysig = [];yhat = [0;0];
sz = [0.0268 0;0 0];
sw = eye(3);sw1 = eye(2);
Pr = sw; Py = [];Pxy = []; Pfd = sw1; Pbd = sw1;
H = [1 0 0;0 1 0];

for t = 1:duration;
   if isnan(theta(t))== 1;
      thetaindex = [thetaindex;t];
   end;
end;
thetafit1 = theta;Wzfit1 = Wz;
for i = 1:length(thetaindex);
   count = 0;
   j = thetaindex(i)-imuno;
   while (count <= 70);
      if isnan(thetafit1(j))~= 1;
         count = count +1;
         X = [X;j];
         Y = [Y;thetafit1(j)];
         Y1 =[Y1;Wzfit1(j)];
         j = j-imuno;
      else
         j = j - imuno;
      end;
   end;
   count = 2;
   while (count <=70);
      A = [1 (Y(count)-Y(count-1))*dt;0 1];C = eye(2);
      Xfd = A*Xfd;
      out = [Y(count);Y1(count)];
      innvect = out - C*Xfd;
    sfd = C*Pfd*C' + sz;
    K = A*Pfd*C'*inv(sfd);
    Xfd = Xfd + K*innvect;
    Pfd = A*Pfd*A'- K*C*Pfd*A'+sw1;
   count = count +1;
   end;
   count =2;
    while (count <= 70);
```

```
        if isnan(thetafit1(j))~=  1;
            count = count +1;
            X = [X;j];
            Y = [Y;thetafit1(j)];
            Y1  = [Y1;Wzfit1(j)];
            j = j+imuno;
        else
            j = j+imuno;
        end;
    end;
    count =70;
     while  (count >=2);
         A = [1  (Y(count)-Y(count-1))*dt;0   1];C  = eye(2);
        Xbd = inv(A)*Xbd;
        out = [Y(count);Y1(count)];
        innvect = out - C*Xbd;
     sbd = C*Pbd*C' + sz;
     K = A*Pbd*C'*inv(sbd);
     Xbd = Xbd + K*innvect;
     Pbd= A*Pbd*A'- K*C*Pbd*A'+ sw1;
    count = count - 1;
    end;

    P = inv( inv(Pfd) + inv(Pbd));
    Xsmooth = P*( inv(Pbd)*Xbd + inv(Pfd)*Xfd);
    thetafit1(thetaindex(i))=  Xsmooth(1);
    Wzfit1(thetaindex(i))  = Xsmooth(2);
end;
[num1,den1]  = butter(1,0.8,'low');
poshat = filtfilt(num1,den1,poshat);
velhat =  filtfilt(num1,den1,velhat);

for t = ch:imuno:duration;
    F = [1 dt 0;0 1 0;sec(xhat(1)+  xhat(2)*dt)^2    (sec(xhat(1)+  xhat(2)*dt)^2)*dt  0];
    Pr  = F*Pr*F'  + sw;
    k = Pr*H'*inv((H*Pr*H'  + sz));
    xhat(1)  = xhat(1)  + dt*xhat(2);
    xhat(3)  = tan(xhat(1)  + xhat(2));
    y = [thetafit1(t);Wzfit1(t)];
    xhat  = xhat  + k*(y - H*xhat);
    Pr  = (eye(3)-  k*H)*Pr;
    posmeas =[posmeas;y(1)];
     poshat = [poshat;xhat(1)];
     velmeas  =[velmeas;y(2)];
     velhat = [velhat;xhat(2)];
end

 close all;
 t = 1:length(poshat);
 figure;
 plot(t,posmeas,'Bl',t,poshat,'R');
 grid;
 xlabel('Total  number of samples');
```

```
ylabel('position');
title ('figure - 1 - position(measured(blue),estimated(Red))');
figure;
plot(t,poshat-posmeas,'G');
grid;
xlabel('samples');
ylabel('error');

t = 1:length(velhat);
figure;
plot(t,velmeas,'Bl',t,velhat,'R');
grid;
xlabel('Total Number of Samples');
ylabel('velocity measured and velocity estimate');
title('figure 2 - velocity(measured (blue),estimated(Red))')
 figure;
plot(t,velhat-velmeas,'G');
grid;
xlabel('samples');
ylabel('error');
```

Appendix - C

Unscented Kalman Filter code

```
clc;
duration = input('Enter duration: ');
dt = input('Enter interval: ');
imuno = input('Enter the no of imus used:');
ch = input('enter the IMU used : ');
posmeas = [];poshat = [];velmeas = [];velhat = [];thetaindex = [];thetafit1 = [];Wzfit1 = [];X = [];Y
= [];Y1 = []; temp = zeros(3,3);temp1 = zeros(2,2); temp2 = zeros(3,2);
Xfd = [0;0];Xbd = [0;0]; Pfd=[];Pbd =[];sfd = [];sbd = [];P =[];Xsmooth = [];K = []; innvect =
[];sw1 = [];

xhat = [0;0;0];xsig = [];ysig = [];yhat = [0;0];
sz = [0.0263 0;0 0];
sw = eye(3);sw1 = eye(2);
Pr = sw; Py = [];Pxy = []; Pfd = sw1; Pbd = sw1;
H = [1 0 0;0 1 0];
for t = 1:duration;
  if isnan(theta(t))== 1;
    thetaindex = [thetaindex;t];
  end;
end;
thetafit1 = theta;Wzfit1 = Wz;

for i = 1:length(thetaindex);
  count = 1;
  j = thetaindex(i)-imuno;
  while (count <= 70);
    if isnan(thetafit1(j))~= 1;
      count = count +1;
      X = [X;j];
      Y = [Y;thetafit1(j)];
      Y1 =[Y1;Wzfit1(j)];
      j = j-imuno;
    else
      j = j - imuno;
    end;
  end;
  count = 2;
  while (count <=70);
    A = [1 (Y(count)-Y(count-1))*dt;0 1];C = eye(2);
    Xfd = A*Xfd;
    out = [Y(count);Y1(count)];
    innvect = out - C*Xfd;
   sfd = C*Pfd*C' + sz;
   K = A*Pfd*C'*inv(sfd);
   Xfd = Xfd + K*innvect;
   Pfd = A*Pfd*A'- K*C*Pfd*A'+sw1;
  count = count +1;
```

```
    end;
    count =2;
     while (count <= 100);
       if isnan(thetafit1(j))~=  1;
          count = count +1;
          X = [X;j];
          Y = [Y;thetafit1(j)];
          Y1  = [Y1;Wzfit1(j)];
          j = j+imuno;
       else
          j = j+imuno;
       end;
    end;
    count = 100;
     while (count >=2);
        A = [1  (Y(count)-Y(count-1))*dt;0   1];C  = eye(2);
       Xbd = inv(A)*Xbd;
       out = [Y(count);Y1(count)];
       innvect = out - C*Xbd;
     sbd = C*Pbd*C' + sz;
     K = A*Pbd*C'*inv(sbd);
     Xbd = Xbd + K*innvect;
     Pbd= A*Pbd*A'- K*C*Pbd*A'+ sw1;
    count = count - 1;
    end;

    P = inv( inv(Pfd) + inv(Pbd));
    Xsmooth = P*( inv(Pbd)*Xbd + inv(Pfd)*Xfd);
    thetafit1(thetaindex(i))=  Xsmooth(1);
    Wzfit1(thetaindex(i))  = Xsmooth(2);
end;
[num1,den1]  = butter(3,0.5,'low');
thetafit1 = filtfilt(num1,den1,thetafit1);
Wzfit1 =  filtfilt(num1,den1,Wzfit1);
for t = ch:imuno:duration;
    np = chol(3*Pr);
    for i = 1:3;
       xsig(:,i)  = xhat + np(i,:)';
       xsig(:,i+3)  = xhat - np(i,:)';
    end;
    for i = 1:3;
    xsig(i,1)  = xsig(i,1)  + dt*xsig(i,2);
    xsig(i,3)  = tan(xsig(i,1));
    end;
    xhat  = [0;0;0];
    for i = 1:6;
   xhat = xhat + xsig(:,i);
    end;
    xhat  = (xhat)/6;
    for i = 1:6;
   temp = temp + (xsig(:,i)-  xhat)*(xsig(:,i)-  xhat)';
   end;
    Pr = temp/6 + sw;
temp = zeros(3,3);
```

```
yhat =[0;0];
   for i = 1:6;
    ysig(:,i)  = H*xsig(:,i);
    yhat = yhat + ysig(:,i);
    end;
    yhat = yhat/6;

   for i = 1:6;
    temp1 = temp1 + (ysig(:,i)-  yhat)*(ysig(:,i)- yhat)';
    end;
    Py = temp1/6 + sz;
  temp1 = zeros(2,2);
    for i = 1:6;
    temp2 = temp2 + (xsig(:,i)-   xhat)*(ysig(:,i)-  yhat)';
    end;
    Pxy  = temp2/6;
  temp2 = zeros(3,2);
    k = Pxy*(Py^-1);
    y = [thetafit1(t);Wzfit1(t)];
    xhat  = xhat  + k*(y-yhat);
    Pr  = Pr  - k*Py*k';
    posmeas =[posmeas;y(1)];
     poshat = [poshat;xhat(1)];
     velmeas =[velmeas;y(2)];
     velhat = [velhat;xhat(2)];
end;

    close all;
 t = 1:length(poshat);
 figure;
 plot(t,posmeas,'Bl',t,poshat,'R');
 grid;
 xlabel('Total  number of samples');
 ylabel('position');
 title ('figure  - 1 - position(measured(blue),estimated(Red))');
 figure;
 plot(t,poshat-posmeas,'G');
 grid;
 xlabel('samples');
 ylabel('error');

 t = 1:length(velhat);
 figure;
 plot(t,velmeas,'Bl',t,velhat,'R');
 grid;
 xlabel('Total  Number of Samples');
 ylabel('velocity  measured and velocity estimate');
 title('figure  2 - velocity(measured (blue),estimated(Red))')
  figure;
 plot(t,velhat-velmeas,'G');
 grid;
 xlabel('samples');
 ylabel('error');
```

Appendix - D

Interpolation code:

```
for i = 1:length(thetaindex);
   count = 1;
   j = thetaindex(i)-imuno;
   while (count <= 70);
      if isnan(thetafit1(j))~= 1;
         count = count +1;
         X = [X;j];
         Y = [Y;thetafit1(j)];
         Y1 =[Y1;Wzfit1(j)];
         j = j-imuno;
      else
         j = j - imuno;
      end;
   end;
   count = 2;
   while (count <=70);
      A = [1 (Y(count)-Y(count-1))*dt;0 1];C = eye(2);
      Xfd = A*Xfd;
      out = [Y(count);Y1(count)];
      innvect = out - C*Xfd;
    sfd = C*Pfd*C' + sz;
    K = A*Pfd*C'*inv(sfd);
    Xfd = Xfd + K*innvect;
    Pfd = A*Pfd*A'- K*C*Pfd*A'+sw1;
   count = count +1;
   end;
   count =2;
    while (count <= 100);
      if isnan(thetafit1(j))~= 1;
         count = count +1;
         X = [X;j];
         Y = [Y;thetafit1(j)];
         Y1 = [Y1;Wzfit1(j)];
         j = j+imuno;
      else
         j = j+imuno;
      end;
   end;
   count = 100;
    while (count >=2);
       A = [1 (Y(count)-Y(count-1))*dt;0 1];C = eye(2);
      Xbd = inv(A)*Xbd;
      out = [Y(count);Y1(count)];
      innvect = out - C*Xbd;
```

```
    sbd = C*Pbd*C' + sz;
    K = A*Pbd*C'*inv(sbd);
    Xbd = Xbd + K*innvect;
    Pbd= A*Pbd*A'- K*C*Pbd*A'+ sw1;
   count = count - 1;
   end;

   P = inv( inv(Pfd) + inv(Pbd));
   Xsmooth = P*( inv(Pbd)*Xbd + inv(Pfd)*Xfd);
   thetafit1(thetaindex(i))=  Xsmooth(1);
   Wzfit1(thetaindex(i))  = Xsmooth(2);
end;
```