Electrical Engineering Department

California Polytechnic State University

Final Project

Kalman Filtering of a Position Measurement

Omer Faruk Gumus

Kevin Zhipeng Yu

Nicholas Curt Garcia

Course Instructor: Dr. Jane Zhang

12/04/20

Introduction

Kalman filtering is a powerful tool that allows us to predict the states of a system with real time observations or measurements. Industries used it for variety of different applications. Things like object tracking and local/global navigation systems, economics prediction etc.

It is a relatively simple algorithm, but in order to better estimate the result with noisy measurement, we need to choose the parameter wisely. Our aim for this project is to implement Kalman filter and understand its parameter effects using MATLAB.

After completing this project, we will discover a basic concept of applying Kalman filter in tracking a moving object in a 2-D and 3-D direction using MATLAB. Also, we will show how to improve the process for the Kalman filtering.

Kalman Filtering Algorithm

In this project, we'll be working on a Discrete-Time Kalman filter. In order to process in a digital machine, we must take is sequentially as we implement it in MATLAB code. The main idea for Kalman filter is to use previous value to predict (forward project) the incoming value. The more measurements it accumulates, the more accurate it gets when returning the next result. It eventually ignores all the potential noise if any.

The Kalman Filter is would estimate the state of a system at instant *k* using the linear stochastic difference equation. To do this, the filter setup an assumption that the state of a system at instant *k* evolved from the prior state at *k – 1*. Below equation describe the prediction at instant *k*:

$$\boldsymbol{x}_k = \boldsymbol{\Phi x}_{k-1} + \boldsymbol{B u}_{k-1} + \boldsymbol{w}_k$$

Under certain restriction, using the Kalman filter for the tracking of moving objects require to setup a dynamic model for the object. It is always paired with the measurement model $\boldsymbol{z}_k$ that describes a relation between the state and measurement at the current step *k*. It is written as:

$$\boldsymbol{z}_k = \boldsymbol{H}\mathbf{x}_\mathrm{k} + \boldsymbol{v}_k$$

where:

- $\boldsymbol{\Phi}$, a matrix *nxn*, is the state transition matrix relating the previous time step *k-1* to the current state
- $\boldsymbol{B}$, a matrix *nxl*, is a control input matrix applied to the optional control input $\boldsymbol{u}_{k-1}$ which we won't use it in this project.
- $\boldsymbol{H}$, a matrix m*xn*, is a transformation matrix that transforms the state into the measurement domain
- $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ represent the process noise vector with the covariance $\boldsymbol{Q}$ and the measurement noise vector with the covariance $\boldsymbol{R}$, respectively. They are assumed statistically independence Gaussian noise with the normal probability distribution.

$$p(w) \sim N(0, \boldsymbol{Q})$$

$$p(v) \sim N(0, \boldsymbol{R})$$

Kalman Equations

To explain the Kalman equations, we must use two terms, a *priori* and a *posteriori* estimates. The $\widehat{x_k^-}$ and the $\boldsymbol{P}_k^-$ are the priori estimates for the state and error covariance respectively. The $\widehat{x_k}$ and the $\boldsymbol{P}_k^-$ are the posteriori estimates for the state and error covariance respectively.

The equations of Kalman filter are divided into two groups: the time update equations and measurement update equations. In some literature, this is sometimes called predictor-corrector or prediction-update.

The time update equations are responsible for projecting forward the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for improving a posteriori estimate by incorporating a new measurement into the priori estimate.

Prediction

In time update equation we need to calculate the predicted state estimate (a priori state estimates) $\widehat{x_k^-}$ and predicted error covariance (a priori error covariance estimates) $\boldsymbol{P}_k^-$. First, the a priori state estimate $\widehat{x_k^-}$ is predicted by using the state dynamic equation model that projects forward one step in time as follows:

$$\widehat{x_k^-} = \boldsymbol{\Phi}\mathbf{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1}$$

where: $\mathbf{x}_{k-1}$ is the previous estimated state (a posteriori state estimate).

Next, the error covariance matrix P_k^- is predicted by:

$$P_k^- = \Phi P_{k-1} \Phi^T + Q$$

where P_{k-1} is the previous estimated error covariance matrix and Q is the process noise covariance.

Update

During the update stage, we compute the Kalman gain K_k as follows:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

Where R is the measurement noise covariance.

After that, we perform the actual measurement z_k.

In order to update the predicted state estimate x_k^-, we need to measure the measurement residual. It is the difference between the true measurement z_k and the previous estimated measurement $H\widehat{x_k^-}$. So, the measurement residual is $z_k - H\widehat{x_k^-}$.

To update the predicted state estimate called updated state estimate x_k is proceed by performing the summation of the previous updated state estimate x_k^- to the product of the Kalman gain and the measurement residual. It can be written as follows:

$$\widehat{x_k} = \widehat{x_k^-} + K_k (z_k - H\widehat{x_k^-})$$

After obtaining the updated state estimate, the filter calculates the updated error covariance P_k, which will be used in the next time step.

$$P_k = (I - K_k H) P_k^-$$

where I is an identity matrix.

The following table 1 is showing the summary of our Kalman filter algorithm.

Discrete Kalman Time update equations	Discrete Kalman filter measurement update equations.
$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$ $\widehat{x_k} = \widehat{x_k^-} + K_k (z_k - H\widehat{x_k^-})$ $P_k = (I - K_k H) P_k^-$	$\widehat{x_k^-} = \Phi \mathbf{x}_{k-1} + B u_{k-1}$ $P_k^- = \Phi P_{k-1} \Phi^T + Q$

Table 1: Kalman filter algorithm table.

Implementation of 2-D Position Measurement

For the 2-D measurement data the Kalman filter is implemented by determining the state space variables for position X, position Y, and their respective velocities. In Figures 1 and 2 the block diagram for the position processes are shown.

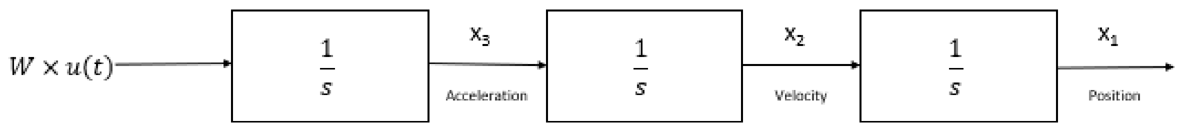

Figure 1: *Block diagram of white noise with magnitude W used to determine X position.*

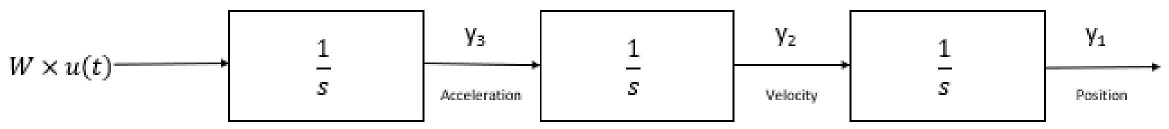

Figure 2: *Block diagram of white noise with magnitude W used to determine Y position.*

Transfer functions are used to determine the continuous-time model of the random process.

$$\frac{X_1(s)}{X_2(s)} = \frac{1}{s} \quad \therefore \dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2$$

$$\frac{X_2(s)}{X_3(s)} = \frac{1}{s} \quad \therefore \dot{\boldsymbol{x}}_2 = \boldsymbol{x}_3$$

Since Y has a symmetrical process to X the following is also true:

$$\frac{Y_1(s)}{Y_2(s)} = \frac{1}{s} \quad \therefore \dot{\boldsymbol{y}}_1 = \boldsymbol{y}_2$$

$$\frac{Y_2(s)}{Y_3(s)} = \frac{1}{s} \quad \therefore \dot{\boldsymbol{y}}_2 = \boldsymbol{y}_3$$

Therefore, the continuous-time state equations are:

$$\begin{bmatrix}\dot{x}_1\\ \dot{y}_1\\ \dot{x}_2\\ \dot{y}_2\end{bmatrix}=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}x_1\\ y_1\\ x_2\\ y_2\end{bmatrix}+\begin{bmatrix}\frac{\Delta t^2}{2}&0\\0&\frac{\Delta t^2}{2}\\ \Delta t&0\\0&\Delta t\end{bmatrix}\begin{bmatrix}x_3\\ y_3\end{bmatrix}$$

To determine the discrete-time state equations $\boldsymbol{\Phi}$, $\boldsymbol{Q}$, and $\boldsymbol{H}$ matrices must be determined.

Deriving Φ:

$$\Phi=L^{-1}[(sI-F)^{-1}]$$

$$\Phi=L^{-1}\left(\begin{bmatrix}s&0&-1&0\\0&s&0&-1\\0&0&s&0\\0&0&0&s\end{bmatrix}^{-1}\right)=L^{-1}\left(\begin{bmatrix}\frac{1}{s}&0&\frac{1}{s^2}&0\\0&\frac{1}{s}&0&\frac{1}{s^2}\\0&0&\frac{1}{s}&0\\0&0&0&\frac{1}{s}\end{bmatrix}\right)$$

$$\Phi=\begin{bmatrix}1&0&t&0\\0&1&0&t\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

We then evaluate Φ at time $t\ =\ \Delta t$.

$$\Phi=\begin{bmatrix}1&0&\Delta \mathrm{t}&0\\0&1&0&\Delta \mathrm{t}\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

Transformation matrix H

The measurement model for 2-D Kalman filter as follows,

$$\boldsymbol{z}_k=\boldsymbol{H}\mathbf{x}_{\mathrm{k}}+\boldsymbol{v}_k$$

In deriving the observation model, we assume that we're only measuring the positions and we are determining the velocities 0. However, in our models, we will estimate the velocities too. So, we can write the measurement model as follows:

$$\boldsymbol{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \end{bmatrix} + \boldsymbol{v}_k$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Process noise covariance matrix Q

The process noise covariance matrix $\boldsymbol{Q}$ or error in the state process is basically can be written as follows:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_x \sigma_{\dot{x}} \\ \sigma_x \sigma_{\dot{x}} & \sigma_{\dot{x}}^2 \end{bmatrix}$$

where σ_x and $\sigma_{\dot{x}}$ are the standard deviations of the position and the velocity, respectively.

We can define the standard deviation of position as the standard deviation of acceleration σ_a multiplied by $\frac{(\Delta t)^2}{2}$. The reason for this is because the $\frac{(\Delta t)^2}{2}$ is the effect that will have on the position. Therefore, by multiplying the standard deviation of the acceleration by $\frac{(\Delta t)^2}{2}$ we'll have the standard deviation of the position. Similarly, if we multiply the standard deviation of the acceleration by (Δt), we'll get the standard deviation of the velocity. So, we can write the process covariance noise $\boldsymbol{Q}$ as follows:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{(\Delta t)^4}{4} & \frac{(\Delta t)^3}{2} \\ \frac{(\Delta t)^3}{2} & (\Delta t)^2 \end{bmatrix} \sigma_a^2$$

Where σ_a^2 is the tuning magnitude of standard deviation of the acceleration.

The process noise covariance matrix $\boldsymbol{Q}$ for 2-D Kalman filter can be written as:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & 0 & \sigma_x \sigma_{\dot{x}} & 0 \\ 0 & \sigma_y^2 & 0 & \sigma_y \sigma_{\dot{y}} \\ \sigma_x \sigma_{\dot{x}} & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & \sigma_y \sigma_{\dot{y}} & 0 & \sigma_{\dot{y}}^2 \end{bmatrix}$$

We can rewrite the process noise covariance matrix for 2-D Kalman filter as:

$$Q = \begin{bmatrix} \frac{(\Delta t)^4}{4} & 0 & \frac{(\Delta t)^3}{2} & 0 \\ 0 & \frac{(\Delta t)^4}{4} & 0 & \frac{(\Delta t)^3}{2} \\ \frac{(\Delta t)^3}{2} & 0 & (\Delta t)^2 & 0 \\ 0 & \frac{(\Delta t)^3}{2} & 0 & (\Delta t)^2 \end{bmatrix} \sigma_a^2$$

Where the σ_a^2 is the magnitude of the standard deviation of the acceleration that is basically the process noise effecting on the process noise covariance matrix.

Measurement noise covariance matrix R

In 2-D Kalman filter, we suppose that the measurement positions x-and y- are both independent, so we can ignore any interaction between them so that the covariance *x* and *y* is 0. We look at only the variance in the *x* and the variance in the *y*.

Then, the measurement noise covariance ***R*** can be written as follows:

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

Results:

The following parameters were set in the 2-D MATLAB code shown in the appendix:

Process noise magnitude W = 1;

Variance of X Var[X] = 1;

Variance of Y Var[Y] = 1;

The following results were produced:

Figure 3 below shows the 2-D position of the submarine. The measured position of the submarine is shown in blue and appears to be corrupted by noise. The Kalman filter estimate of the position of the submarine is shown in orange and appears to track the measured position. This indicates that the Kalman filter is overfitting the noise data and including it in the estimated position of the submarine. This is not good behavior because the Kalman filter is not filtering out the noise present in the measured data.

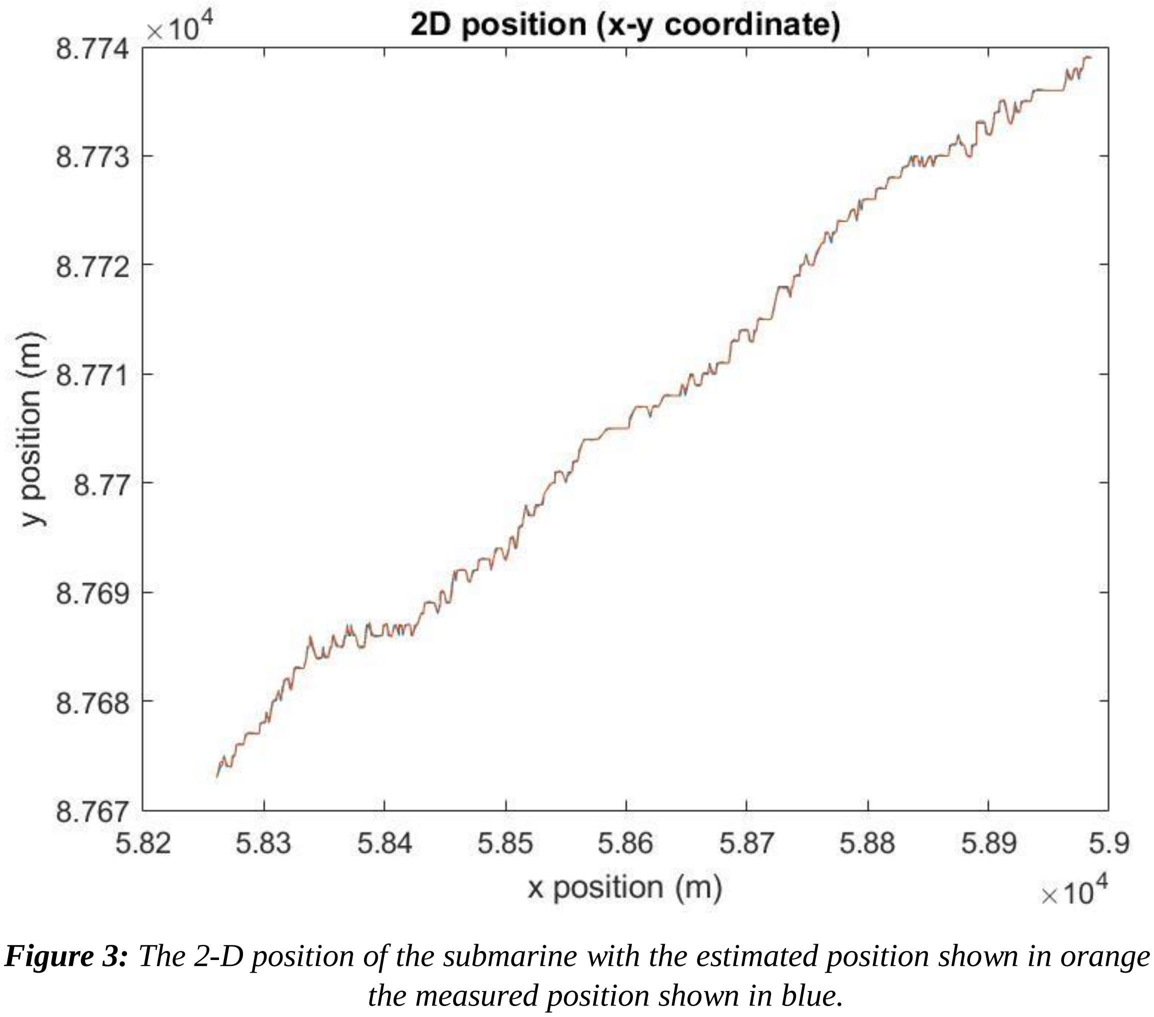

Figure 3: *The 2-D position of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 4 below shows the 2-D velocity of the submarine. The blue shows the measured velocity, and the orange shows the estimated velocity. For most of the path the estimated velocity is focused between 1m/s and 2m/s on the x-axis and 0m/s and 0.5m/s on the y-axis. The estimated velocity appears to track the measured velocity well with the current noise input magnitude of W=1.

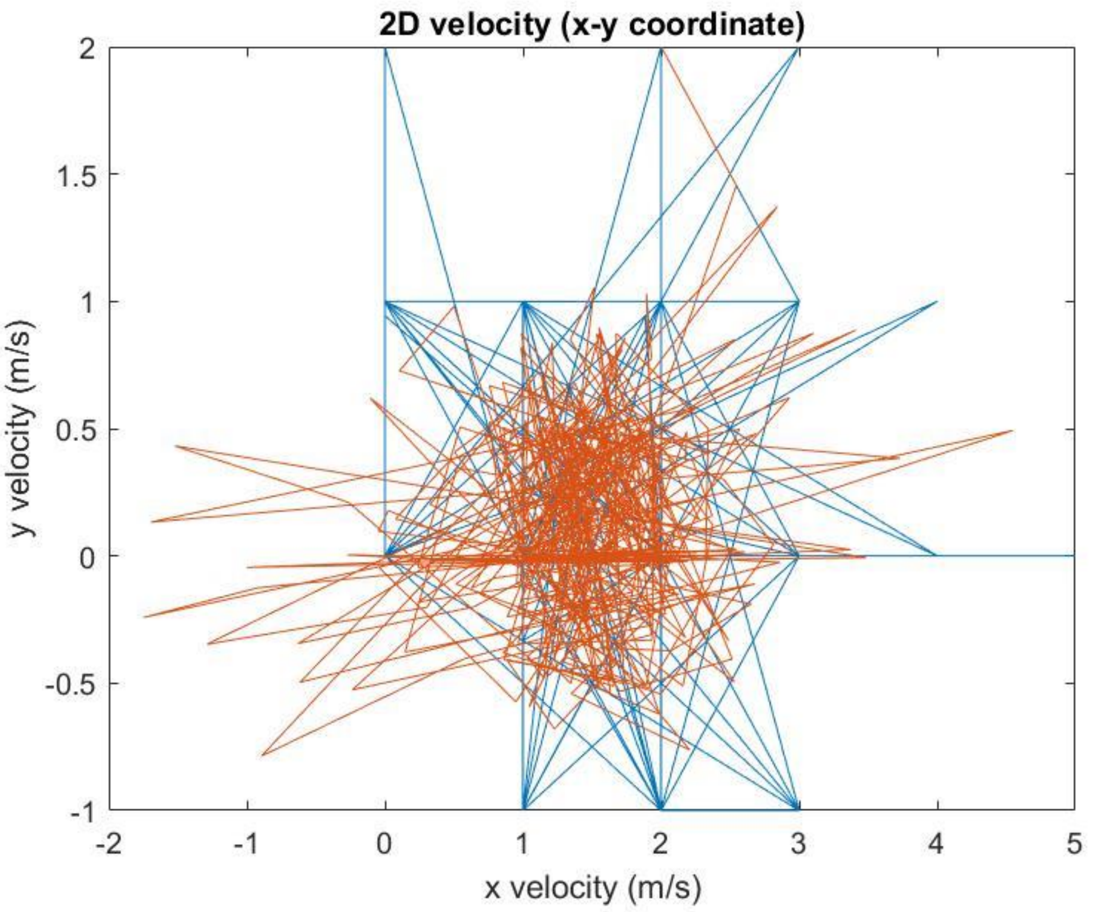

Figure 4: *The 2-D velocity of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 5 below displays the x position of the submarine over its course. The measured X position (blue) is tracked very closely with the Kalman filter's estimated X position (orange). This indicates that Kalman filter has too high of a white noise input magnitude of W=1. The Kalman filter's estimated X position is not filtering out much of the noise present in the measured X position. Therefore, the estimated X position is overfitted to the measured X position.

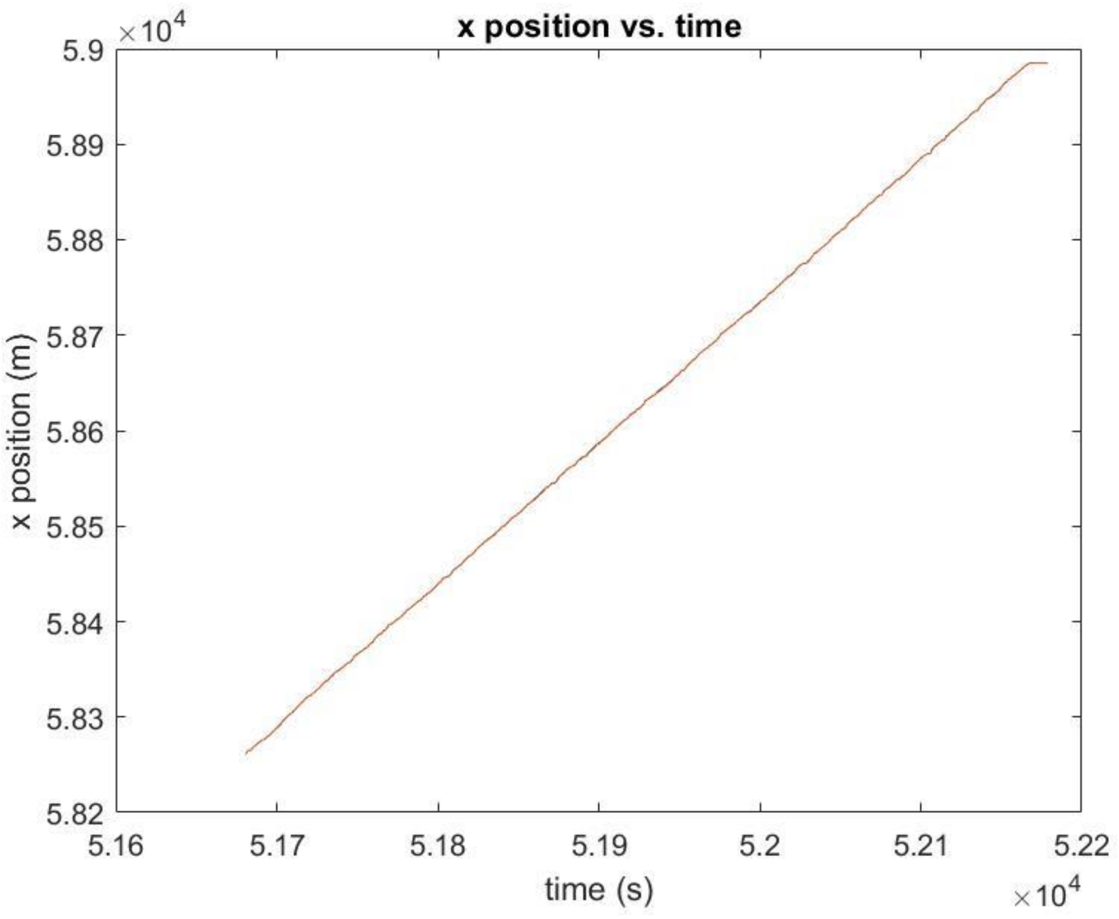

Figure 5: *The X component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 6 below displays the Y position of the submarine over its course. The measured Y position (blue) is tracked very closely with the Kalman filter's estimated Y position (orange). This indicates that Kalman filter has too high of a white noise input magnitude of W=1. The Kalman filter's estimated Y position is not filtering out much of the noise present in the measured Y position. Therefore, the estimated Y position is overfitted to the measured Y position.

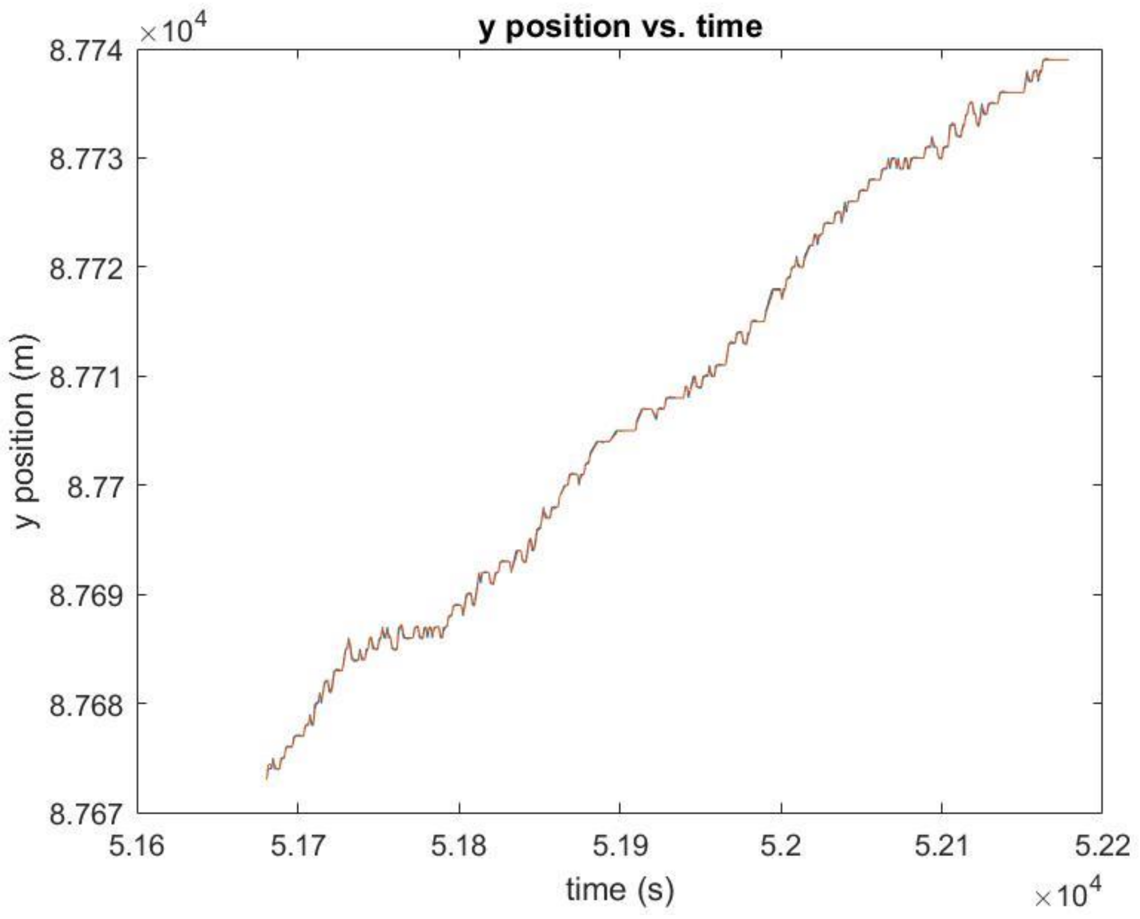

Figure 6: *The Y component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 7 below shows the X velocity of the submarine over time. The measured X velocity (blue) appears to be 1yd/s or 2yd/s on average. The estimated X velocity also appears to be within the same range on average. The measured X velocity has more datapoints outlying that range, while the estimated X velocity has much fewer. This indicated that at W=1 there is some noise being filtered out of the velocity measurement.

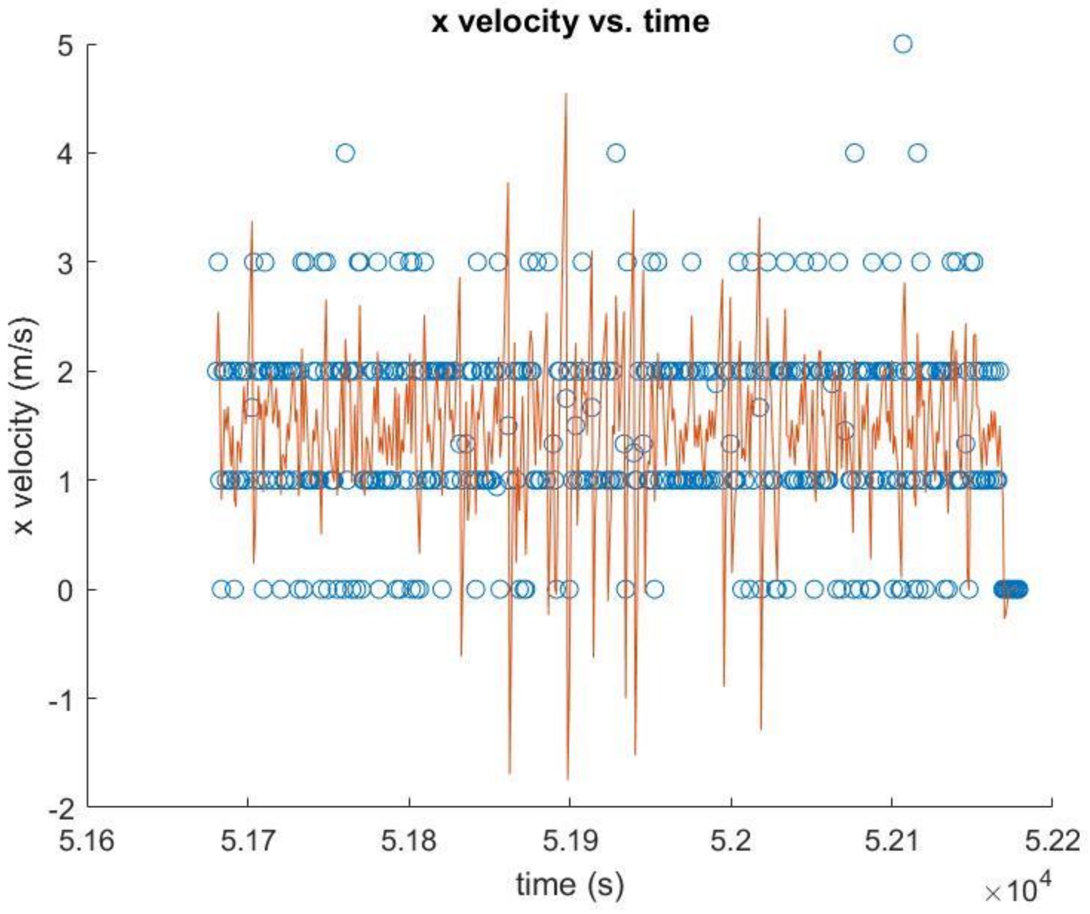

Figure 7: *The X component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 8 below shows the Y velocity of the submarine over time. The measured Y velocity (blue) appears to be between -1yd/s and 1yd/s on average trending toward 0m/s. The estimated Y velocity also appears to be within the 0yd/s and 0.5yd/s on average. This indicates that at W=1 there is some noise being filtered out of the velocity measurement, but there is still a large error over time.

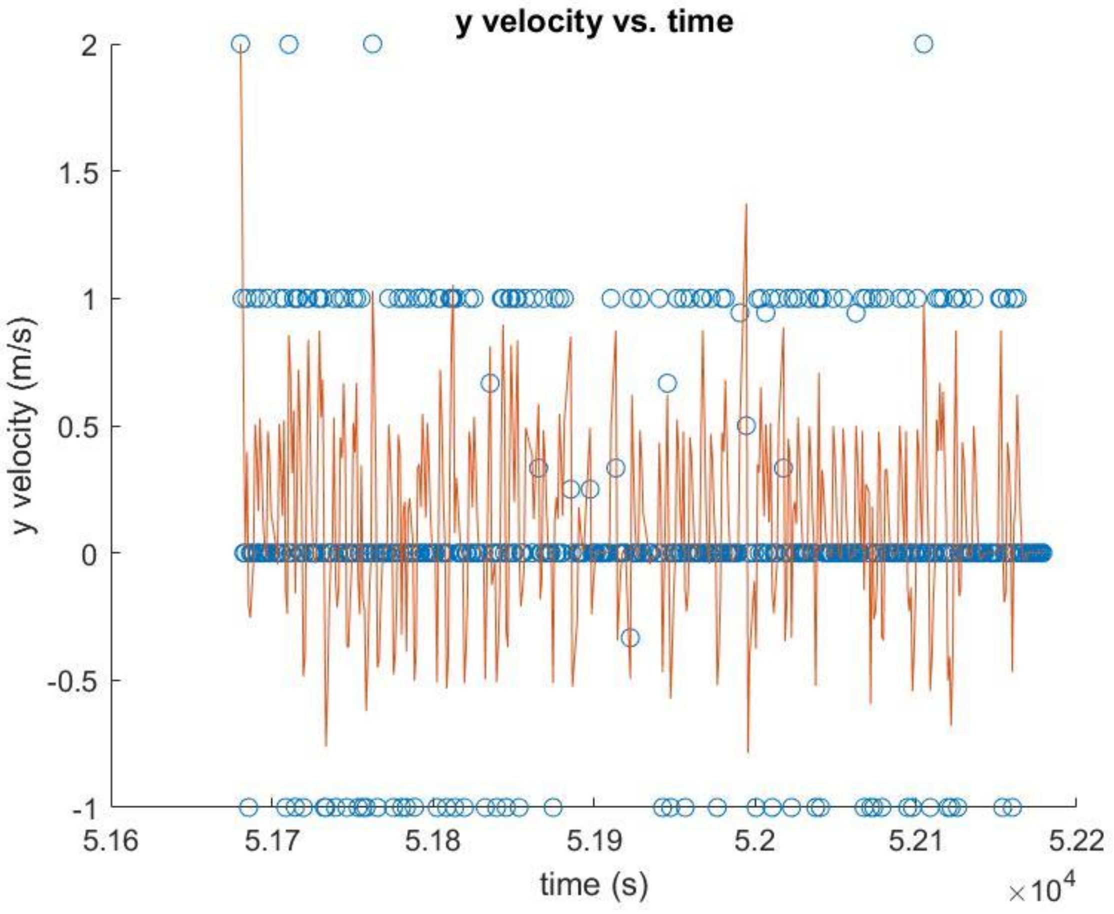

Figure 8: *The Y component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

The following parameters were set in the 2-D MATLAB code shown in the appendix:

Process noise magnitude W = 0.001;

Variance of X Var[X] = 1;

Variance of Y Var[Y] = 1;

Only the process noise magnitude W was adjusted, and the following results were produced:

Figure 9 below shows the 2-D position of the submarine. The measured position of the submarine is shown in blue and appears to be corrupted by noise. The Kalman filter estimate of the position of the submarine is shown in orange and has a much smoother curve. The measured position is constantly moving above and below the estimated position. This is good behavior because it indicates that the estimated position is not overfitted to the measurement data.

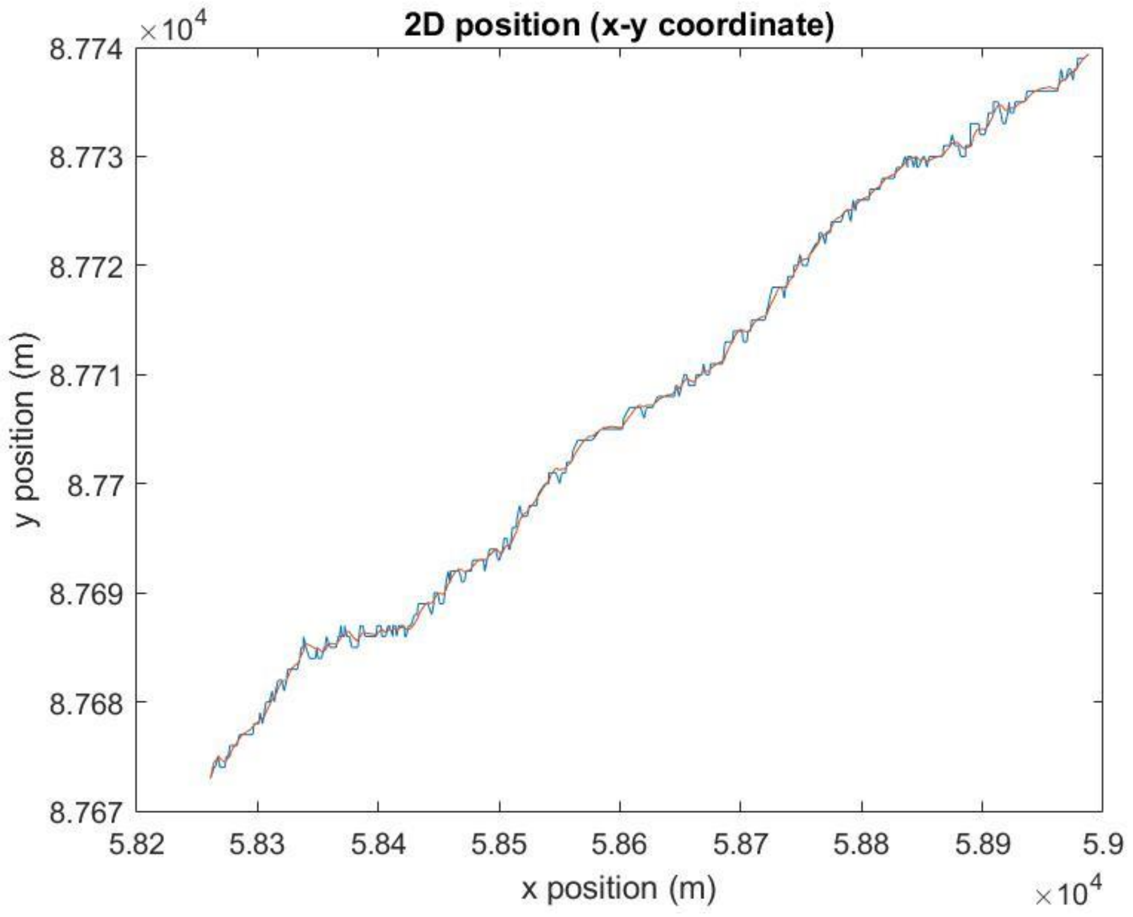

Figure 9: *The 2-D position of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 10 below shows the 2-D velocity of the submarine. The blue shows the measured velocity and the orange shows the estimated velocity. With a lower noise the estimated velocity is much more concentrated when compared with Figure 4.

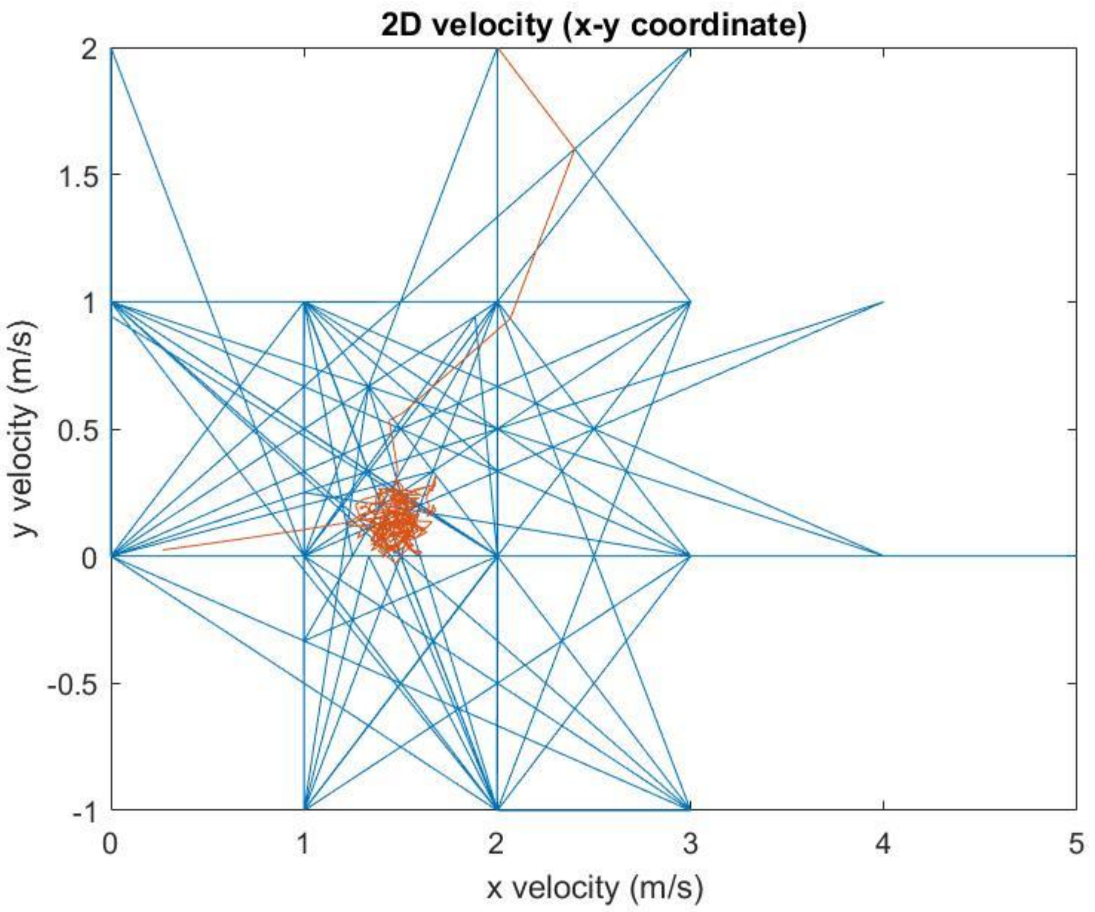

Figure 10: *The 2-D velocity of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 11 below displays the X position of the submarine over its course. The measured X position (blue) is tracked closely with the Kalman filter's estimated X position (orange). However, the measured X position is oscillating above and below the estimated X position. This indicates that Kalman filter is filtering out noise in from the measurement when W=0.001. When compared with Figure 5 decreasing the noise present in the input process reduced the estimated X position's tracking of the measured X position.

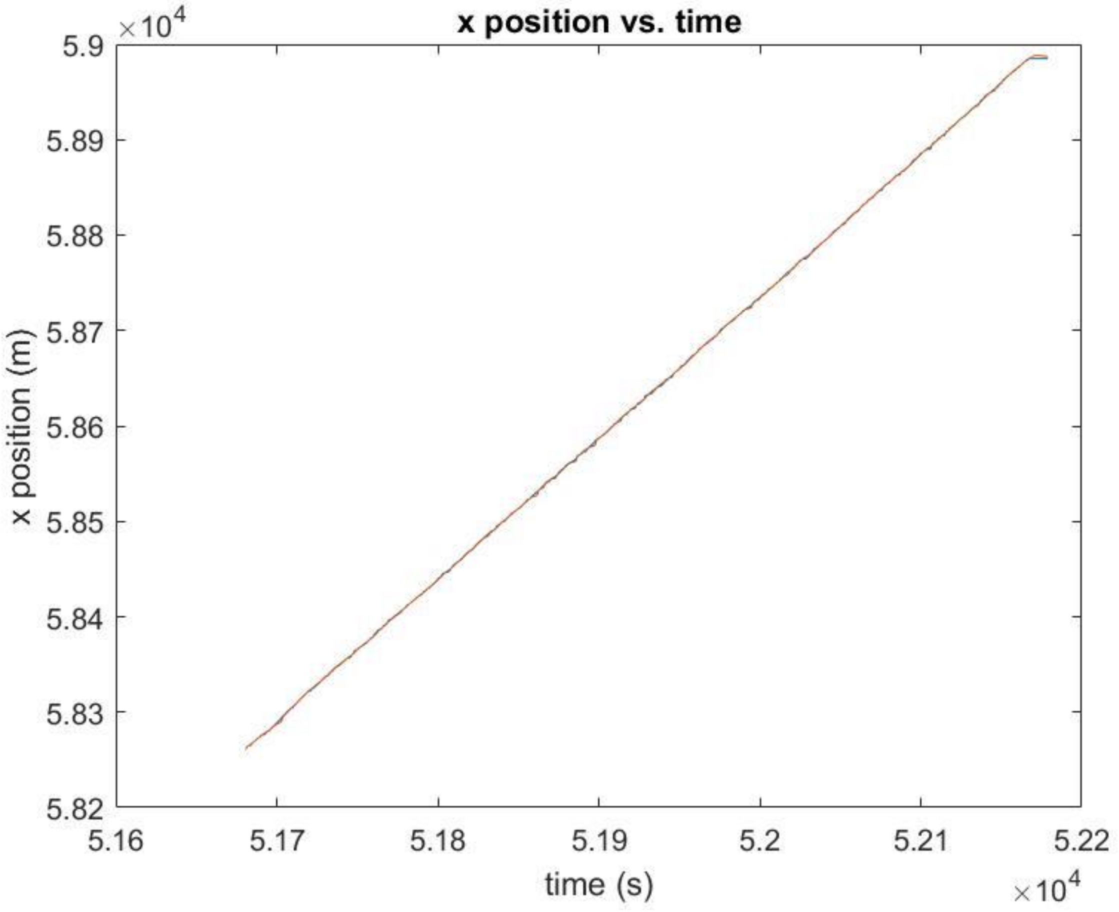

Figure 11: *The X component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 12 below displays the Y position of the submarine over its course. The measured Y position (blue) is tracked with the Kalman filter's estimated Y position (orange). However, the measured Y position is oscillating above and below the estimated X position. This indicates that Kalman filter is filtering out noise in from the measurement when W=0.001. When compared with Figure 6 decreasing the noise present in the input process reduced the estimated Y position's tracking of the measured Y position. This means the estimated data may be closer to the real position of the submarine.

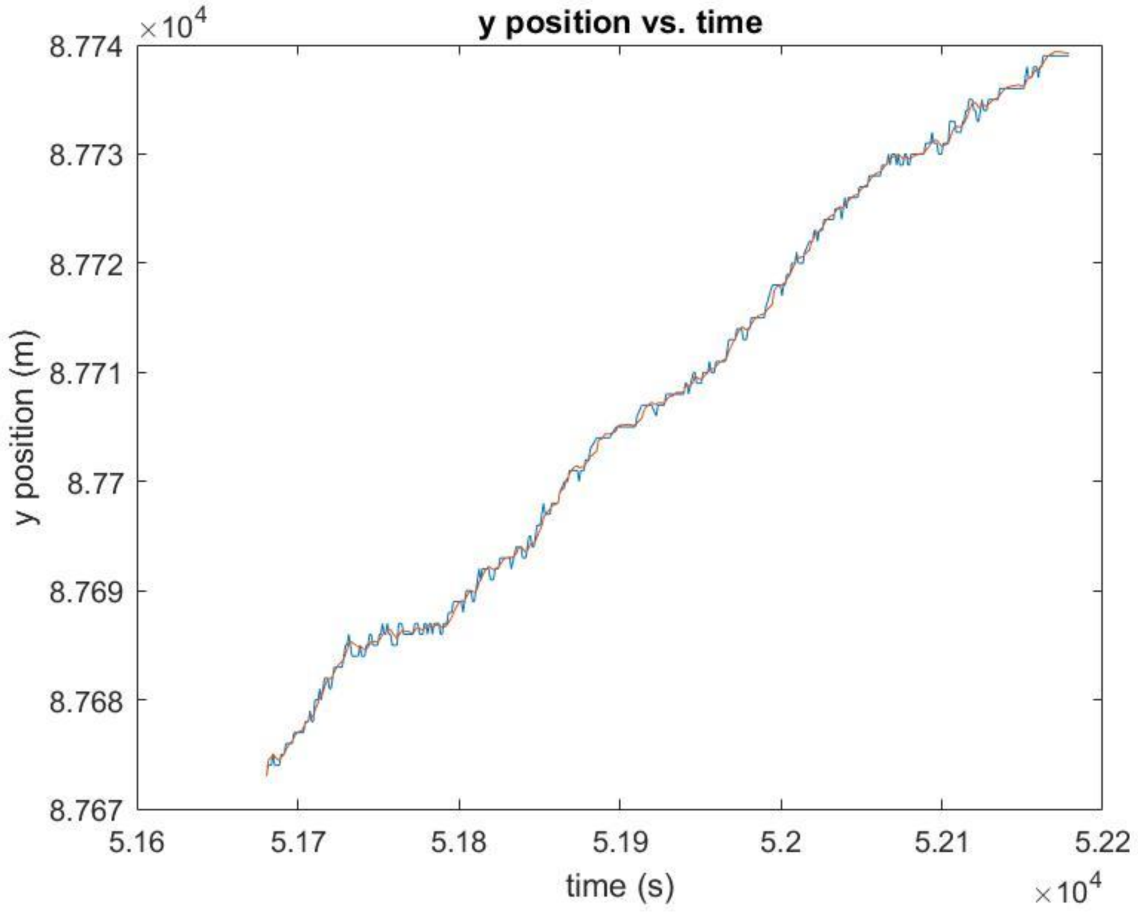

Figure 12: *The Y component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 13 below shows the X velocity of the submarine over time. The measured X velocity (blue) appears to be 1yd/s or 2yd/s on average. The estimated X velocity also appears to be central to that range and is on average 1.5yd/s. The measured X velocity has more datapoints outlying that range, while the estimated X velocity has much fewer. This indicated that at W=0.001 there is noise being filtered out of the velocity measurement. When compared to Figure 7 The velocity of the submarine appears to be more constant as the noise process input reduces in magnitude. The Kalman filter appears to be working like a moving average filter in this instance.

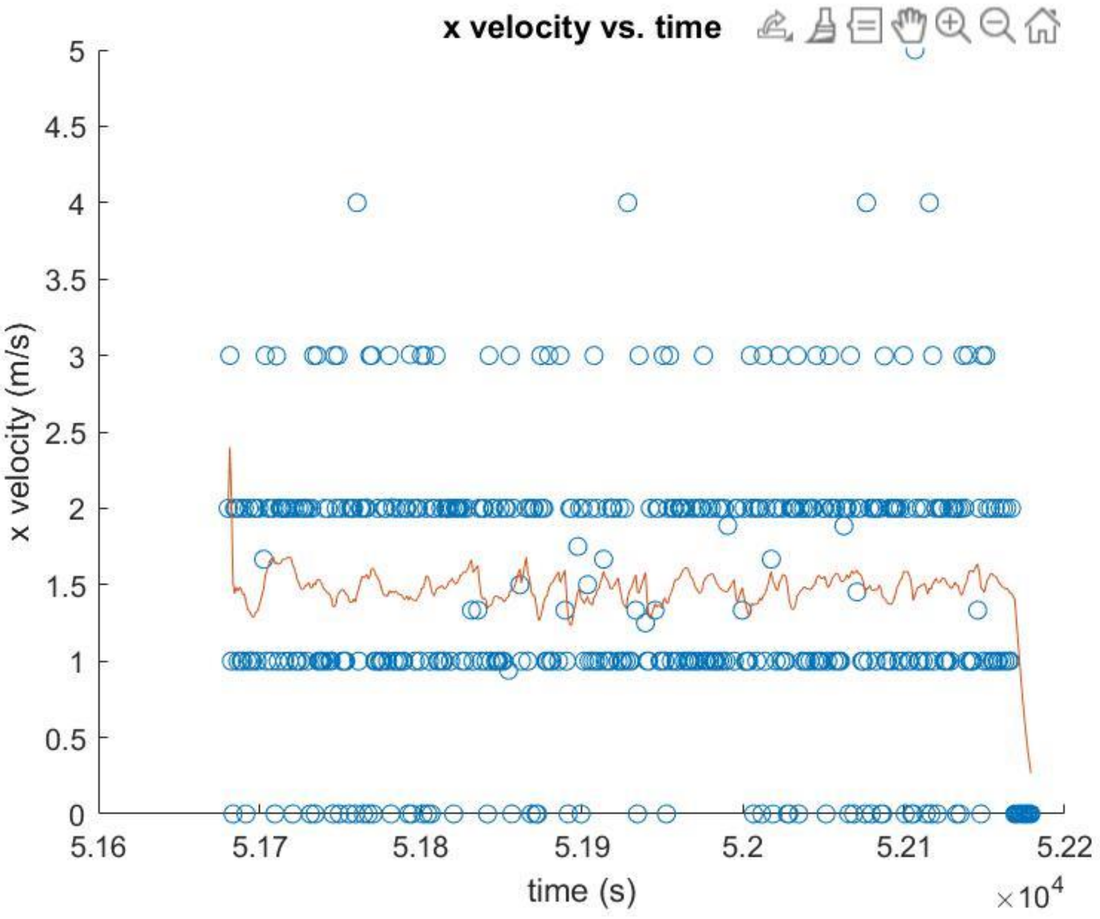

Figure 13: *The X component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 14 below shows the Y velocity of the submarine over time. The measured Y velocity (blue) appears to be between -1yd/s and 1yd/s on average trending toward 0m/s. The estimated Y velocity also appears to be within the 0yd/s and 0.25yd/s on average. This indicates that at W=0.001 there is noise being filtered out of the velocity measurement. When compared with Figure 8 a reduction of input process magnitude causes the Kalman filter to behave similarly to a moving average filter.

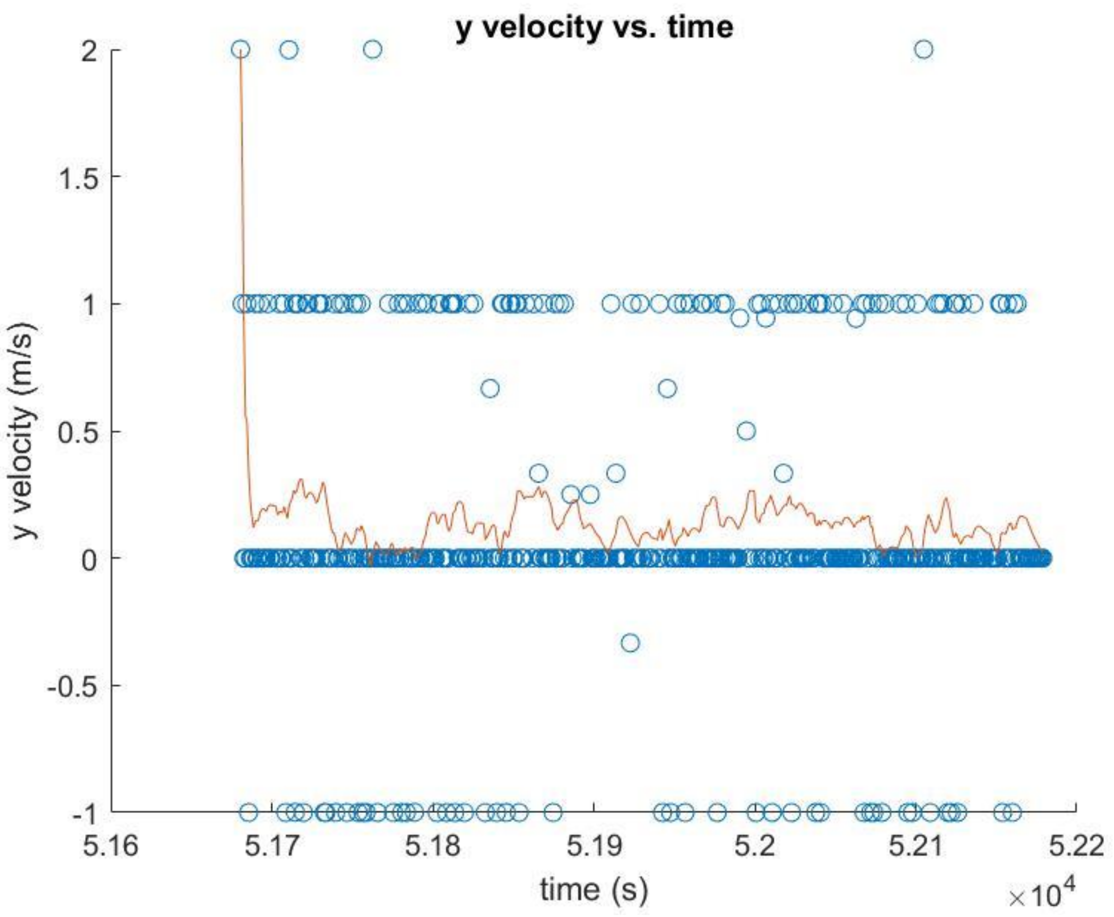

Figure 14: *The Y component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Analysis and Discussion

At this point of time, we can see that a 2-D Kalman Filter implementation is nothing more than feeding a multi-dimension input matrix into the function, with bigger set of parametric matrices, and get a multi-dimension result. In MATLAB, the size of the matrices during the simulation run (calculation) will take care of them itself. What we need to define is really the Q and the R matrices, because only those two matrices control the behavior of our filter. As you can see in the previous pictures, we run two simulations with the same input data, but with two different noise set. For the 2-D positional data, there are noise introduced in the input data set. Figure 3 shows the measure position is not much different than our filter position, when we set our process noise to be 1, and variance of both axes to be 1. As we investigate each component, x and y, our filtered data are also seriously impact by the noise. Another obvious view is from Figure 7 and 8, its first order, velocities; you can see there is no set value it settles at.

Next, we changed the process noise to be 0.001, which is supposed to help our filtering. Indeed, as we keep the variances the same and reduce the magnitude of Q matrix down, we see a significant improvement. Figure 9 shows a desirable filtered data processed by the 2-D Kalman Filter. When we examine the velocities estimation associate with this set simulation parameters, they are affected by the noise not as much comparing to the first case, thus a better performance overall.

Conclusion for 2-D Position Measurement

At the beginning of the project, getting the matrices relationship with delta time and variances is hard. There is example on simple 1-D filtering, but it was not the same. Yet, the theory is the same. After we applied the same derivation method onto the 2-D case, we see some similarity and different on the structure of the matrices.

In MATLAB, we encountered several difficulties while implementing the code, including setting up the environment, writing the proper function and outputting self-explanatory pictures of the situation. After we are sure of the parameters we are going to use from our calculation, we need to define each of the matrix as they are related to certain constants and variable. In order to make the Kalman Filter more general, we need to define the function with input and output, separate of the simulation environment. And we had a difficult time aligning the matrix dimension inside and outside of the functions. Lastly, there are several pictures is required by the project, but we are thinking there are other graph that explained the problem very well, which we may also included in our report. It turns out worth a thousand word.

This part of the project tells us a lot in term of the process noise variable. From the experiment above, we can assume that when we lower the process noise number, or Q matrix, we can get estimations that are “less like” the measure value. In some cases, where there is a lot of noise involved in the measurement or the subject itself, we can use a small process noise number. But in the case when we want to preserve the variation of the data, we can use a bigger process noise number, as it will look “more like” the measure value.

Implementation of 3-D Position Measurement

For the 3-D measurement data the Kalman filter is implemented by determining the state space variables for position X, position Y, position Z and their respective velocities. In Figures 1 and 2 the block diagram for the position processes are shown.

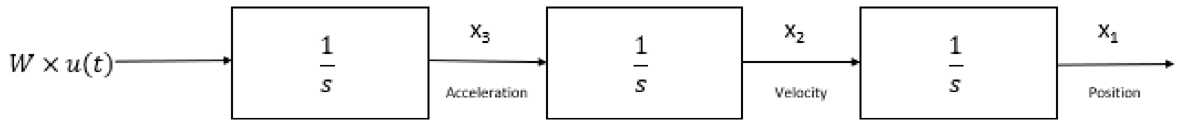

Figure 15: *Block diagram of white noise with magnitude W used to determine X position.*

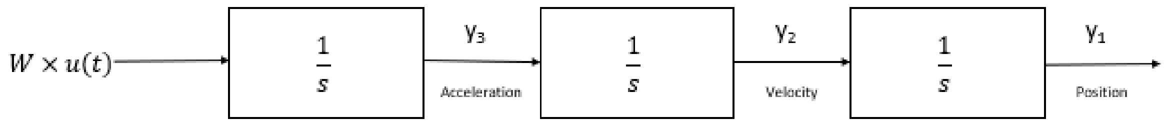

Figure 16: *Block diagram of white noise with magnitude W used to determine Y position.*

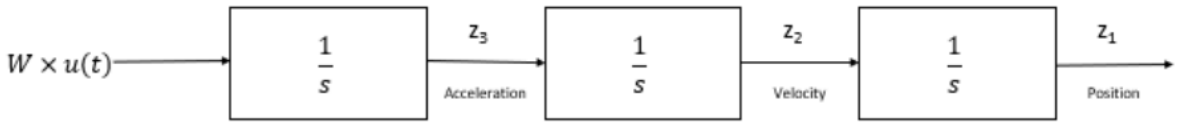

Figure 17: *Block diagram of white noise with magnitude W used to determine Z position.*

Transfer functions are used to determine the continuous-time model of the random process.

$$\frac{X_1(s)}{X_2(s)} = \frac{1}{s} \;\therefore \dot{\boldsymbol{x}}_{\boldsymbol{1}} = \boldsymbol{x}_{\boldsymbol{2}}$$

$$\frac{X_2(s)}{X_3(s)} = \frac{1}{s} \;\therefore \dot{\boldsymbol{x}}_{\boldsymbol{2}} = \boldsymbol{x}_{\boldsymbol{3}}$$

Since Y has a symmetrical process to X the following is also true:

$$\frac{Y_1(s)}{Y_2(s)} = \frac{1}{s} \;\therefore \dot{\boldsymbol{y}}_{\boldsymbol{1}} = \boldsymbol{y}_{\boldsymbol{2}}$$

$$\frac{Y_2(s)}{Y_3(s)} = \frac{1}{s} \;\therefore \dot{\boldsymbol{y}}_{\boldsymbol{2}} = \boldsymbol{y}_{\boldsymbol{3}}$$

Since Z has a symmetrical process to X and Y the following is also true:

$$\frac{Z_1(s)}{Z_2(s)} = \frac{1}{s} \;\therefore\; \dot{\boldsymbol{z}}_{\boldsymbol{1}} = \boldsymbol{z}_{\boldsymbol{2}}$$

$$\frac{Z_2(s)}{Z_3(s)} = \frac{1}{s} \;\therefore\; \dot{\boldsymbol{z}}_{\boldsymbol{2}} = \boldsymbol{z}_{\boldsymbol{3}}$$

Therefore, the continuous-time state equations are:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{z}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}$$

To determine the discrete-time state equations $\boldsymbol{\Phi}$, $\boldsymbol{Q}$, and $\boldsymbol{H}$ matrices must be determined.

Deriving *Φ:*

$$\Phi = L^{-1}[(sI - F)^{-1}]$$

$$\boldsymbol{\Phi} = \mathrm{L}^{-1}\left(\begin{bmatrix} s & 0 & 0 & -1 & 0 & 0 \\ 0 & s & 0 & 0 & -1 & 0 \\ 0 & 0 & s & 0 & 0 & -1 \\ 0 & 0 & 0 & s & 0 & 0 \\ 0 & 0 & 0 & 0 & s & 0 \\ 0 & 0 & 0 & 0 & 0 & s \end{bmatrix}^{-1}\right) = \mathrm{L}^{-1}\left(\begin{bmatrix} \frac{1}{s} & 0 & 0 & \frac{1}{s^2} & 0 & 0 \\ 0 & \frac{1}{s} & 0 & 0 & \frac{1}{s^2} & 0 \\ 0 & 0 & \frac{1}{s} & 0 & 0 & \frac{1}{s^2} \\ 0 & 0 & 0 & \frac{1}{s} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{s} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{s} \end{bmatrix}\right)$$

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We then evaluate Φ at time $t = \Delta t$.

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix H

The measurement model for 3-D Kalman filter as follows,

$$\boldsymbol{z}_k = \boldsymbol{H}\mathbf{x}_{\mathrm{k}} + \boldsymbol{v}_k$$

In deriving the observation model, we assume that we're only measuring the positions and we are determining the velocities 0. So, we can write the measurement model as follows:

$$\boldsymbol{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{z}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{z}_2 \end{bmatrix} + \boldsymbol{v}_k$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Process noise covariance matrix Q

We can follow the same process in the 2-D $\boldsymbol{Q}$ matrix. Then, the process noise covariance matrix $\boldsymbol{Q}$ for 3-D Kalman filter can be written as:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{(\Delta t)^4}{4} & 0 & 0 & \frac{(\Delta t)^3}{2} & 0 & 0 \\ 0 & \frac{(\Delta t)^4}{4} & 0 & 0 & \frac{(\Delta t)^3}{2} & 0 \\ 0 & 0 & \frac{(\Delta t)^4}{4} & 0 & 0 & \frac{(\Delta t)^3}{2} \\ \frac{(\Delta t)^3}{2} & 0 & 0 & (\Delta t)^2 & 0 & 0 \\ 0 & \frac{(\Delta t)^3}{2} & 0 & 0 & (\Delta t)^2 & 0 \\ 0 & 0 & \frac{(\Delta t)^3}{2} & 0 & 0 & (\Delta t)^2 \end{bmatrix} \sigma_a^2$$

Where the σ_a^2 is the magnitude of the standard deviation of the acceleration that is basically the process noise effecting on the process noise covariance matrix.

Measurement noise covariance matrix R

In 3-D Kalman filter, we suppose that the measurement positions x, y, and z are independent, so we can ignore any interaction between them so that the covariance *x*, *y*, *and z* is 0. We look at only the variance in the *x*, y and z coordinates respectively.

Then, the measurement noise covariance ***R*** can be written as follows:

$$\boldsymbol{R} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix}$$

Results

The following parameters were set in the 3-D MATLAB code shown in the appendix:

Process noise magnitude W = 1;

Variance of X Var[X] = 1000;

Variance of Y Var[Y] = 1000;

Variance of Y Var[Z] = 1000;

The following results were produced:

Figure 18 below shows the 3-D position of the submarine. The measured position of the submarine is shown in blue and appears to be corrupted by noise. The Kalman filter estimate of the position of the submarine is shown in orange and appears to track the measured position. The input noise magnitude is W = 1. However, on a 3-D graph it is difficult to determine if the estimated position data tracks closely with the measured position data. Further investigation into the component positions is necessary.

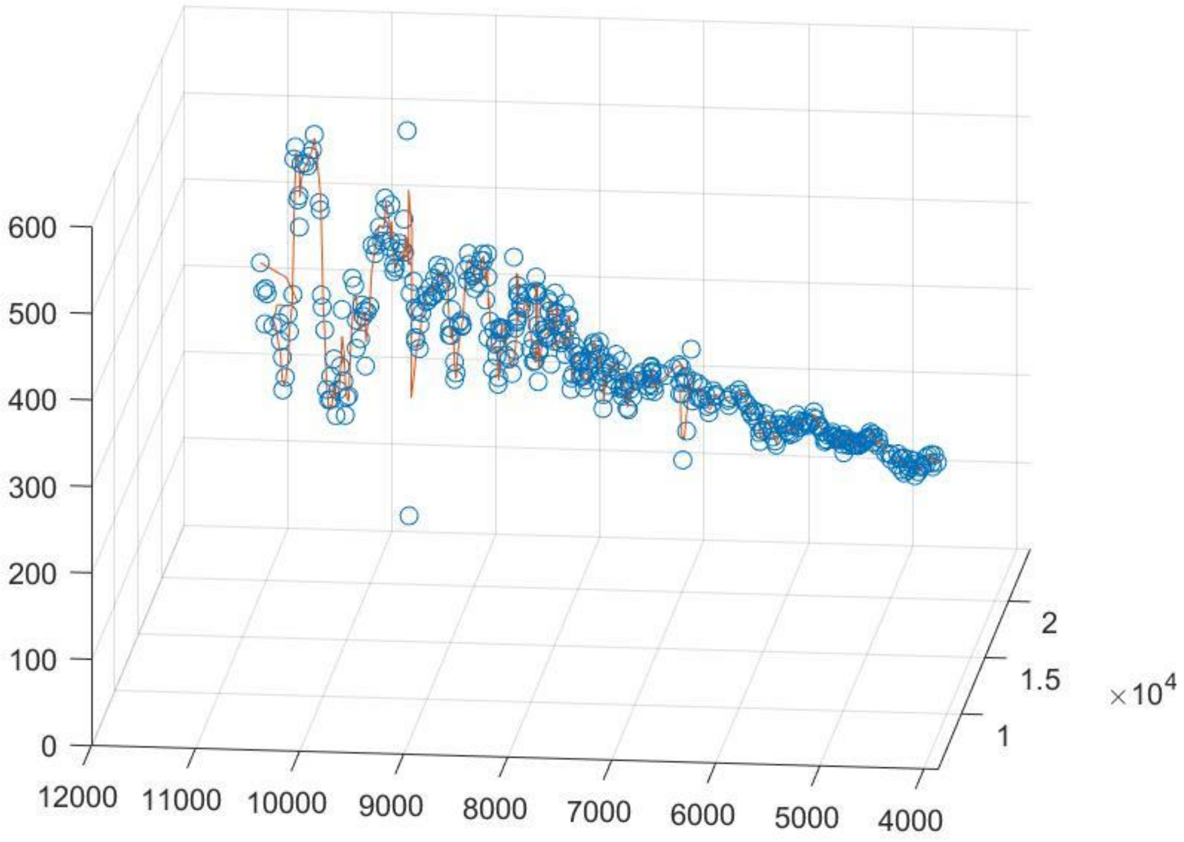

Figure 18: *The 3-D position of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 19 below shows the 3-D velocity of the submarine. The blue shows the measured velocity, and the orange shows the estimated velocity. The measured velocity appears to be very noisy and the estimated velocity appears to change frequently on the left side of the graph and approach a less noisy value as it approaches the right.

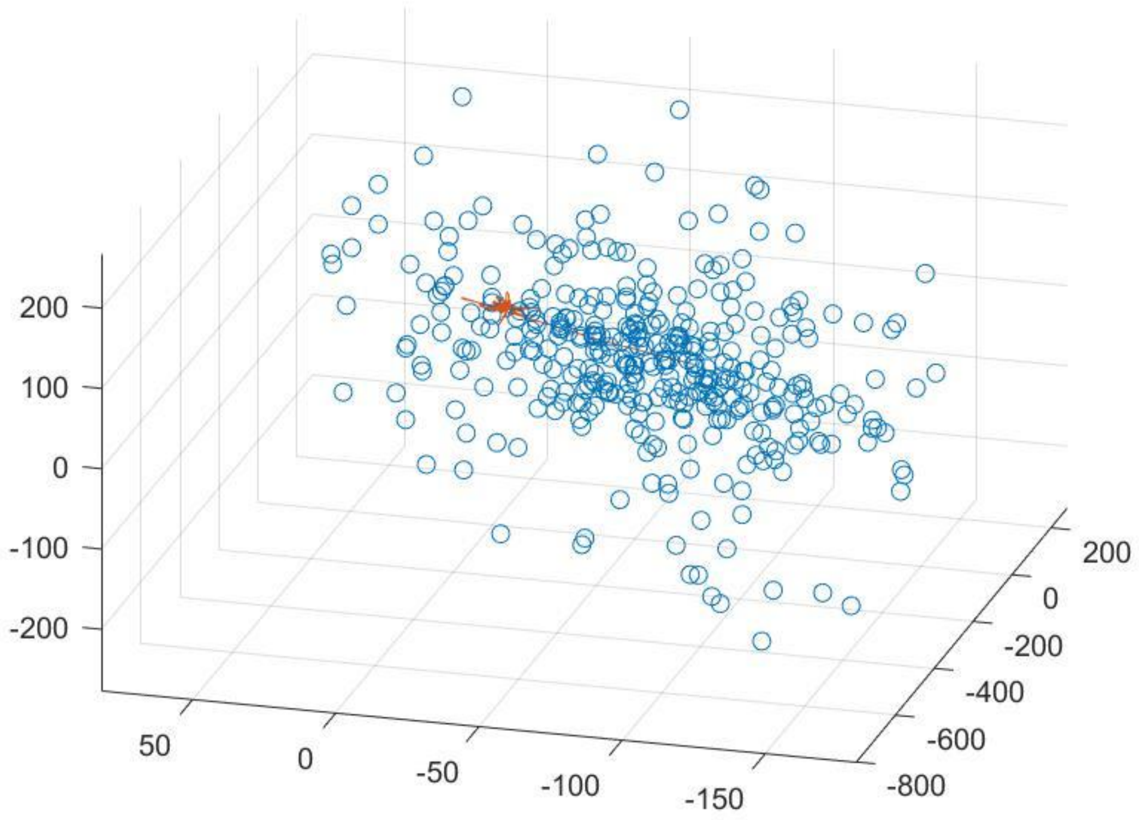

Figure 19: *The 3-D velocity of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 20 below displays the X position of the submarine over its course. The measured X position (blue) is tracked very closely with the Kalman filter's estimated X position (orange). This indicates that Kalman filter has too high of a white noise input magnitude of W=1. The Kalman filter's estimated X position is not filtering out much of the noise present in the measured X position. Therefore, the estimated X position is overfitted to the measured X position.

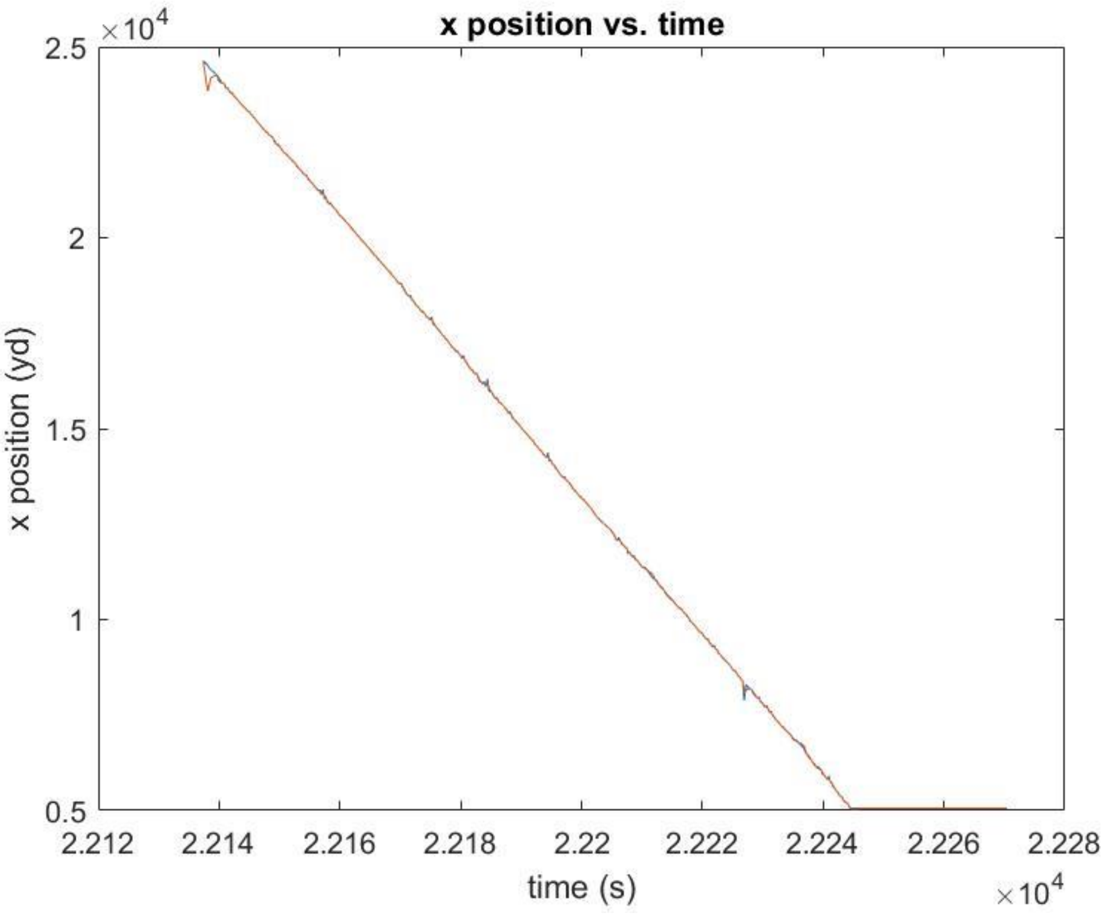

Figure 20: *The X component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 21 below displays the Y position of the submarine over its course. The measured Y position (blue) is tracked very closely with the Kalman filter's estimated Y position (orange). This indicates that Kalman filter has too high of a white noise input magnitude of W=1. The Kalman filter's estimated Y position is not filtering out much of the noise present in the measured Y position. Therefore, the estimated Y position is overfitted to the measured Y position.

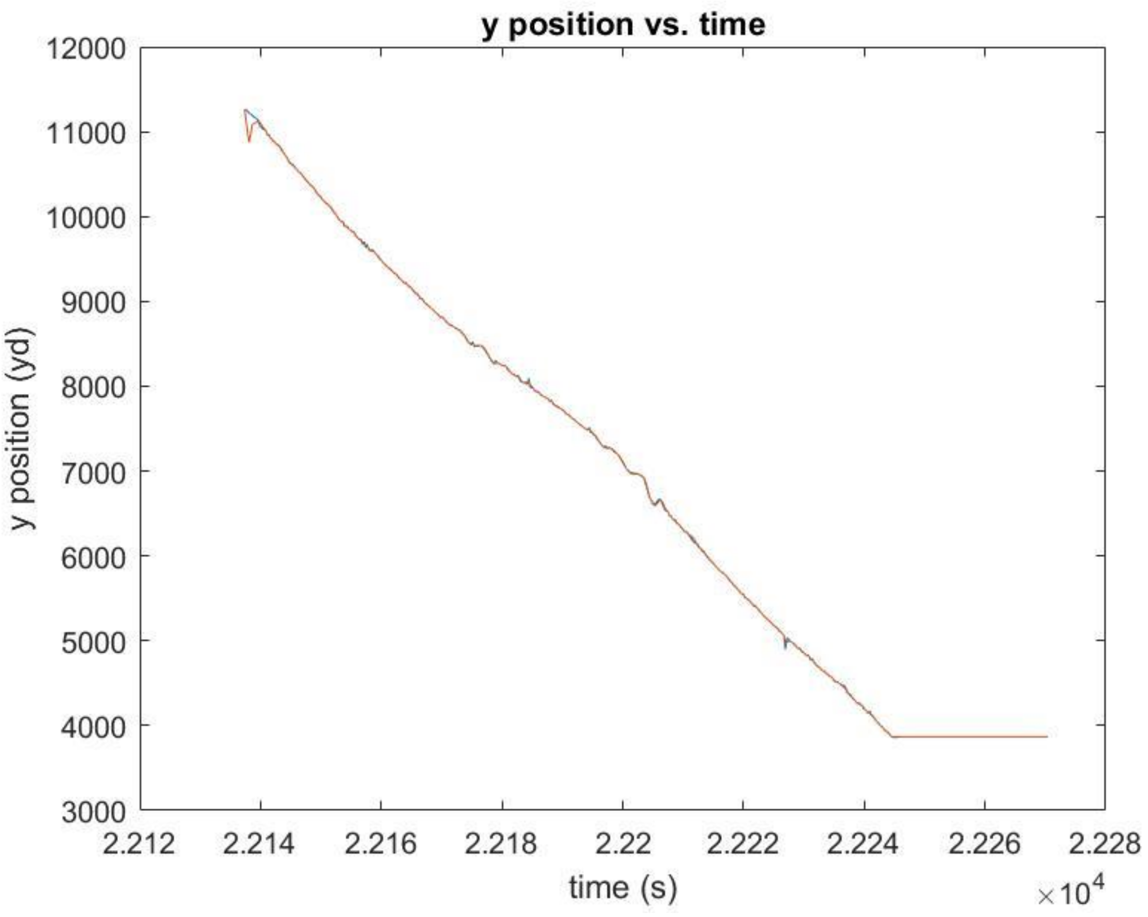

Figure 21: *The Y component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 22 below displays the Z position of the submarine over its course. The measured Z position (blue) is tracked very closely with the Kalman filter's estimated Z position (orange). This indicates that Kalman filter has too high of a white noise input magnitude of W=1. The Kalman filter's estimated Z position is not filtering out much of the noise present in the measured Z position. Therefore, the estimated Z position is overfitted to the measured Z position. Because this feature is consistent with Figures 21, 22, and 23 it is safe to say Figure 18 also tracks much of the noise present in the measured data as it is the composite graph of position.

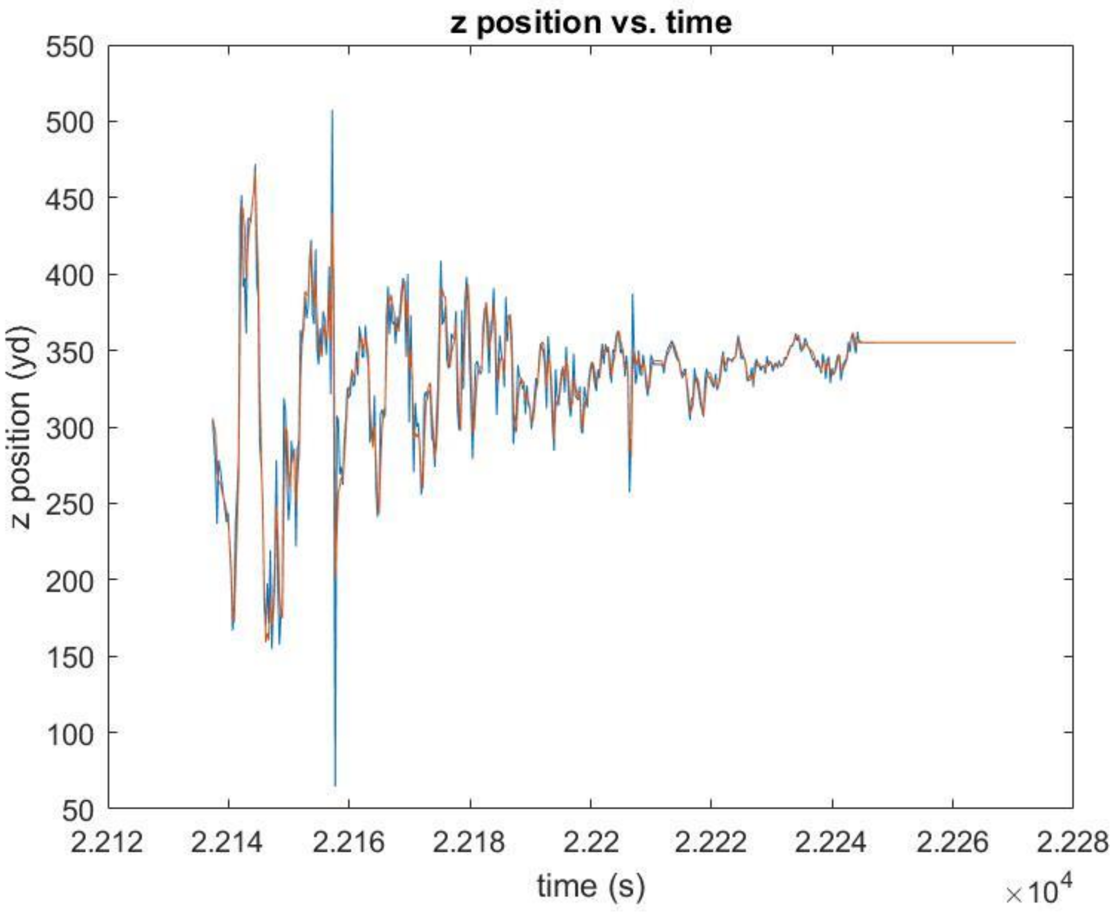

Figure 22: *The Z component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 23 below shows the X velocity of the submarine over time. The measured X velocity (blue) appears to be noisy up until approximately time 22,240 seconds where it approaches 0yd/s. The estimated X velocity (orange) appears to hold a consistent value of approximately 0yd/s. This indicated that at W=1 there is noise being filtered out of the velocity measurement.

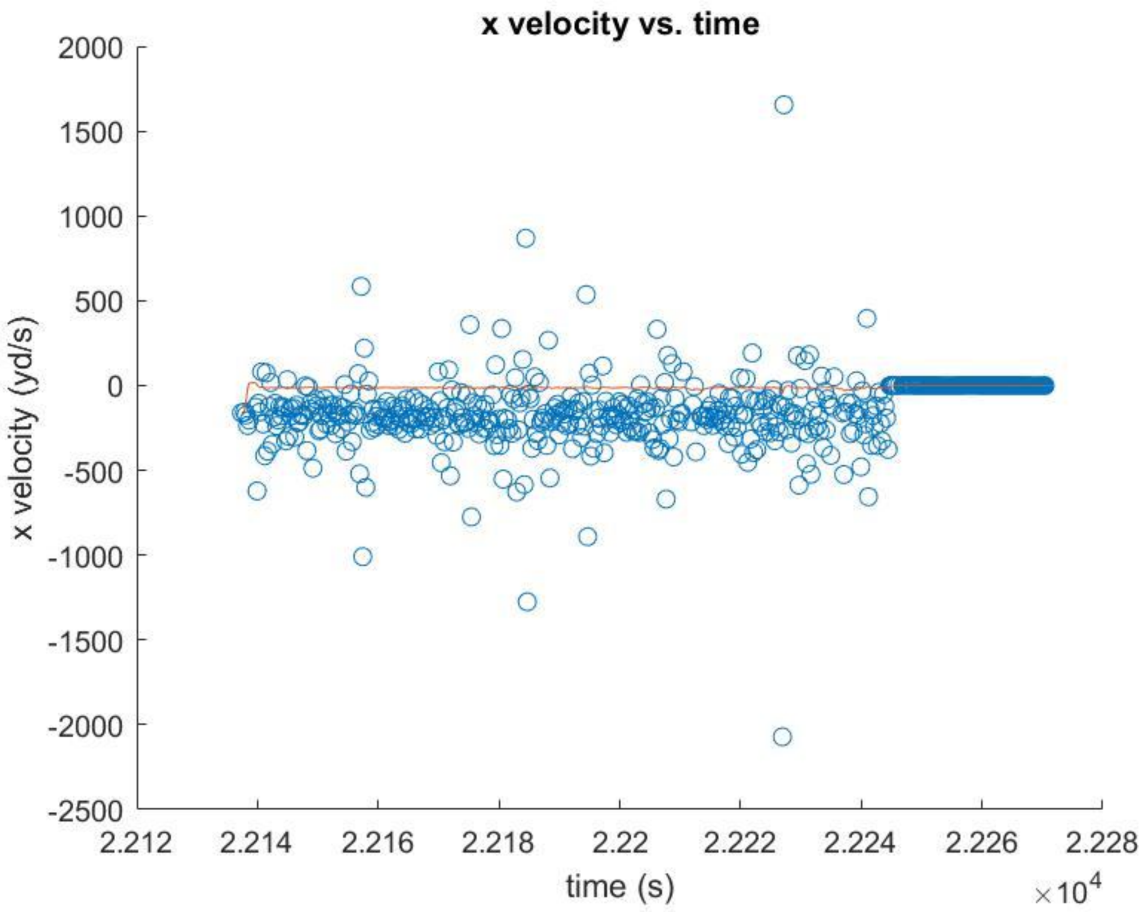

Figure 23: *The X component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 24 below shows the Y velocity of the submarine over time. The measured Y velocity (blue) appears to be noisy up until approximately time 22,240 seconds where it approaches 0yd/s. The estimated Y velocity (orange) appears to hold a consistent value of approximately 0yd/s. This indicated that at W=1 there is noise being filtered out of the velocity measurement. This data is very similar to Figure 23.

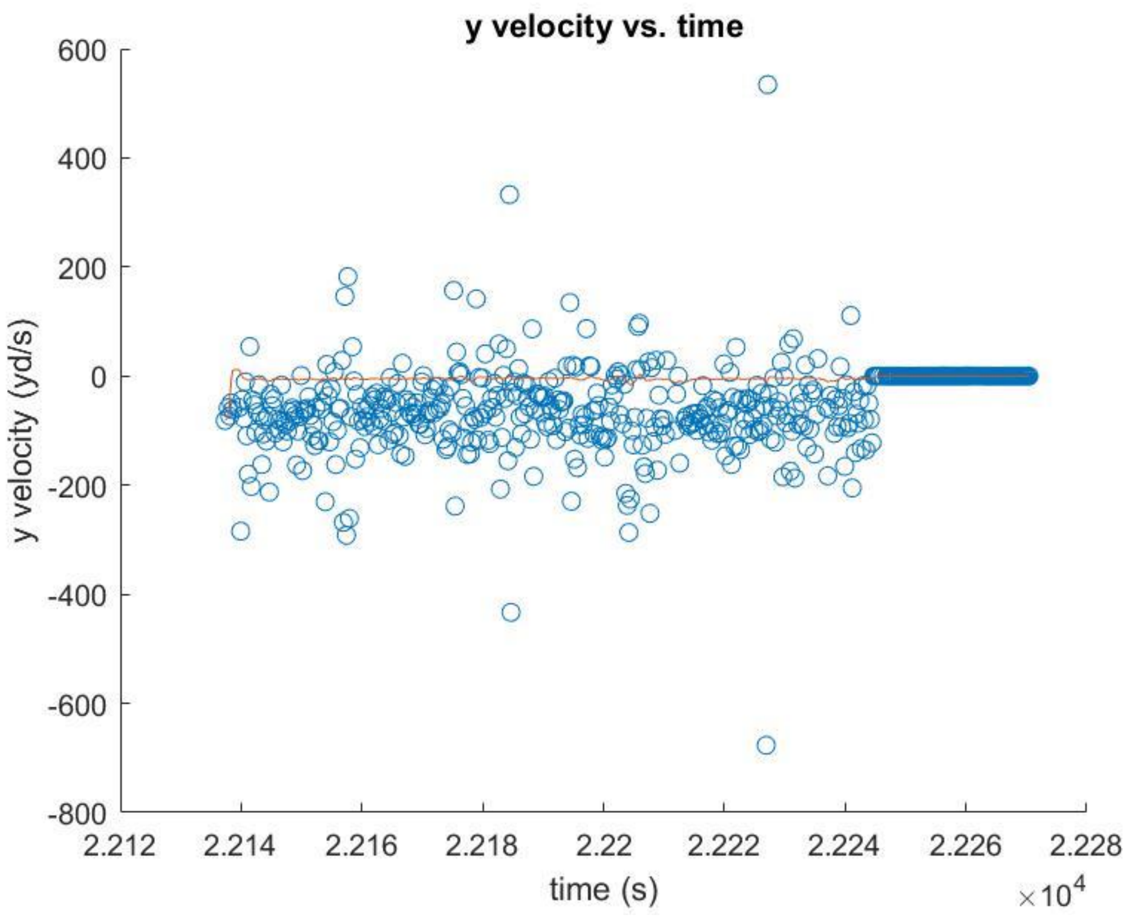

Figure 24: *The Y component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 25 below shows the Z velocity of the submarine over time. The measured Z velocity (blue) appears to be noisy up until approximately time 22,240 seconds where it approaches 0yd/s. The estimated Z velocity (orange) appears to hold a consistent value of approximately 0yd/s. This indicated that at W=1 there is noise being filtered out of the velocity measurement. This data is very similar to Figures 23 and 24.

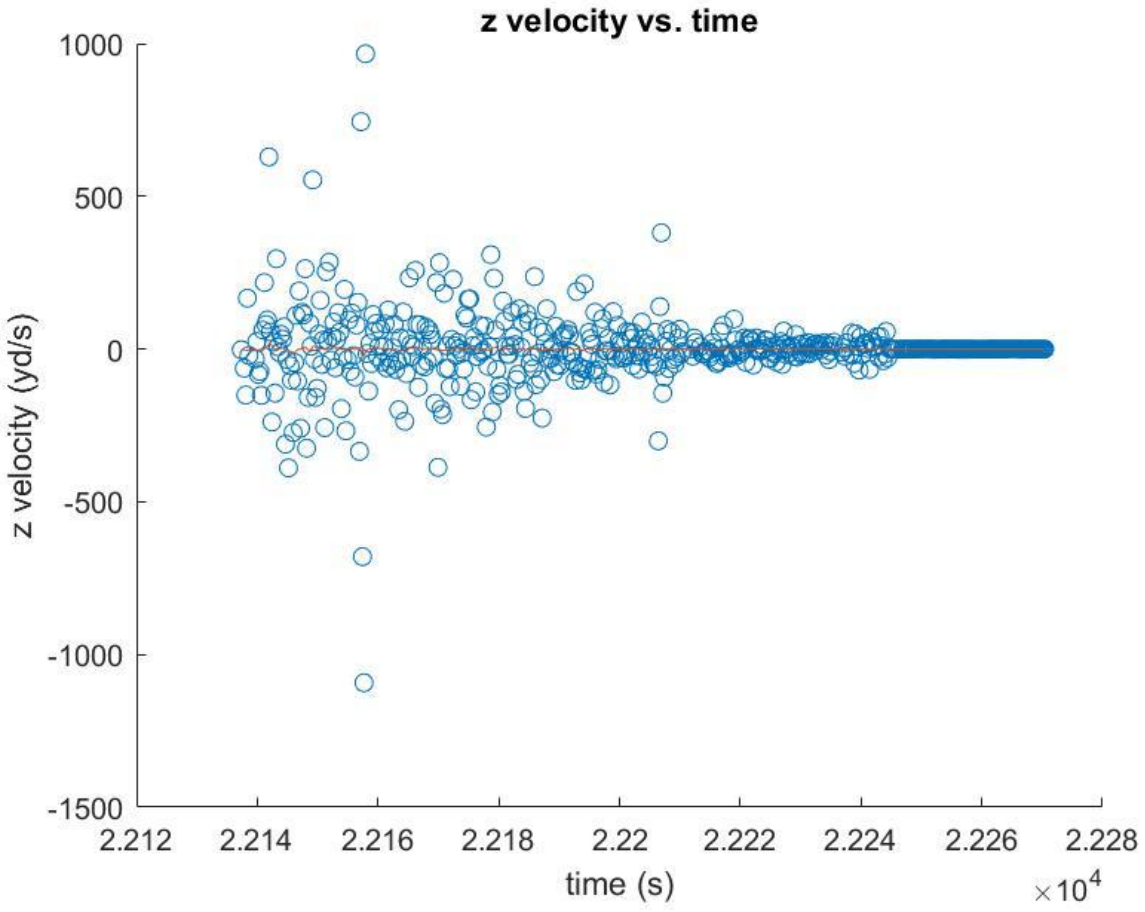

Figure 25: *The Z component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

The following parameters were set in the 3-D MATLAB code shown in the appendix:

Process noise magnitude W = 0.0001;

Variance of X Var[X] = 1000;

Variance of Y Var[Y] = 1000;

Variance of Y Var[Z] = 1000;

Only the process noise magnitude W was adjusted, and the following results were produced:

Figure 18 below shows the 3-D position of the submarine. The measured position of the submarine is shown in blue and appears to be corrupted by noise. The Kalman filter estimate of the position of the submarine is shown in orange and appears to track the measured position. The input noise magnitude is W = 0.0001. However, on a 3-D graph it is difficult to determine if the estimated position data tracks closely with the measured position data. The results appear very similar to Figure 20 which has a W = 1. Further investigation into the component plots is needed.

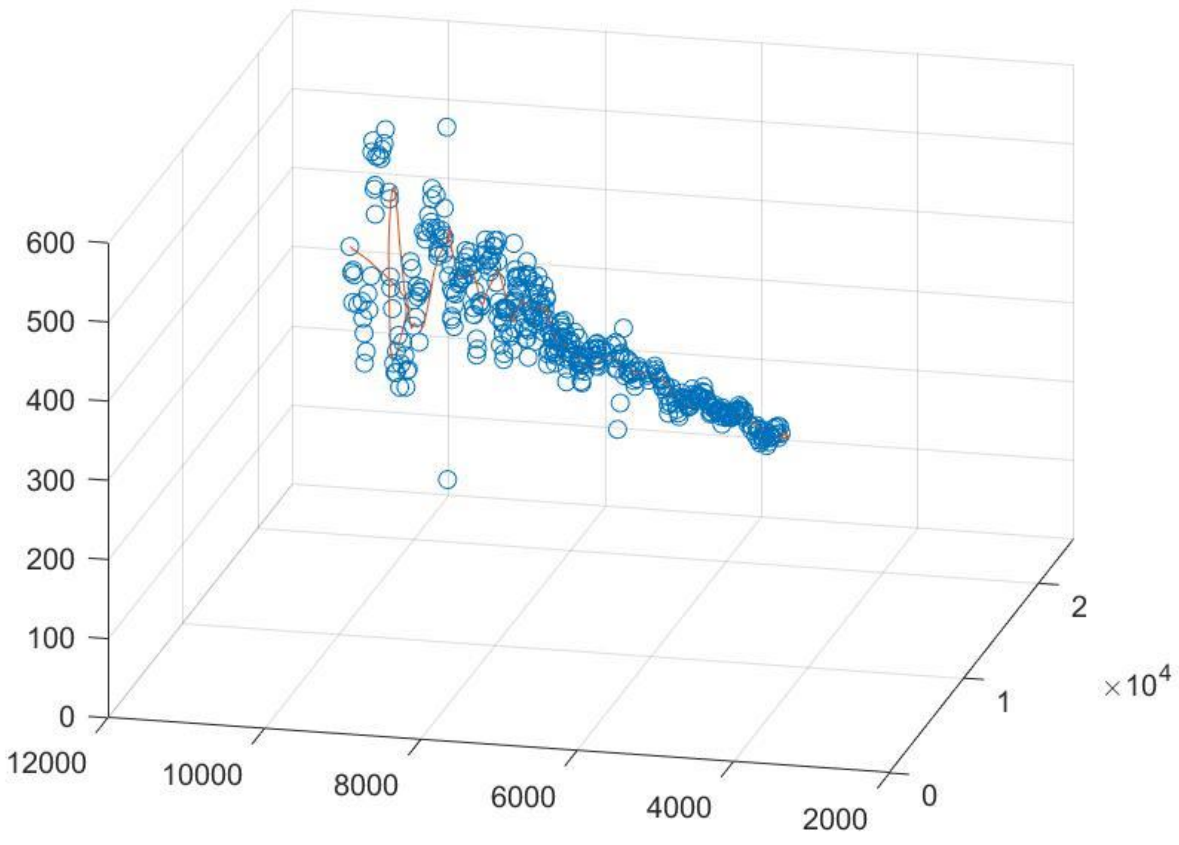

Figure 26: *The 3-D position of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 27 below shows the 3-D velocity of the submarine. The blue shows the measured velocity, and the orange shows the estimated velocity. The measured velocity appears to be very noisy and the estimated velocity appears to be constant. When compared with Figure 19, the estimated velocity is constant for more data points. Further investigation into component plots is necessary.

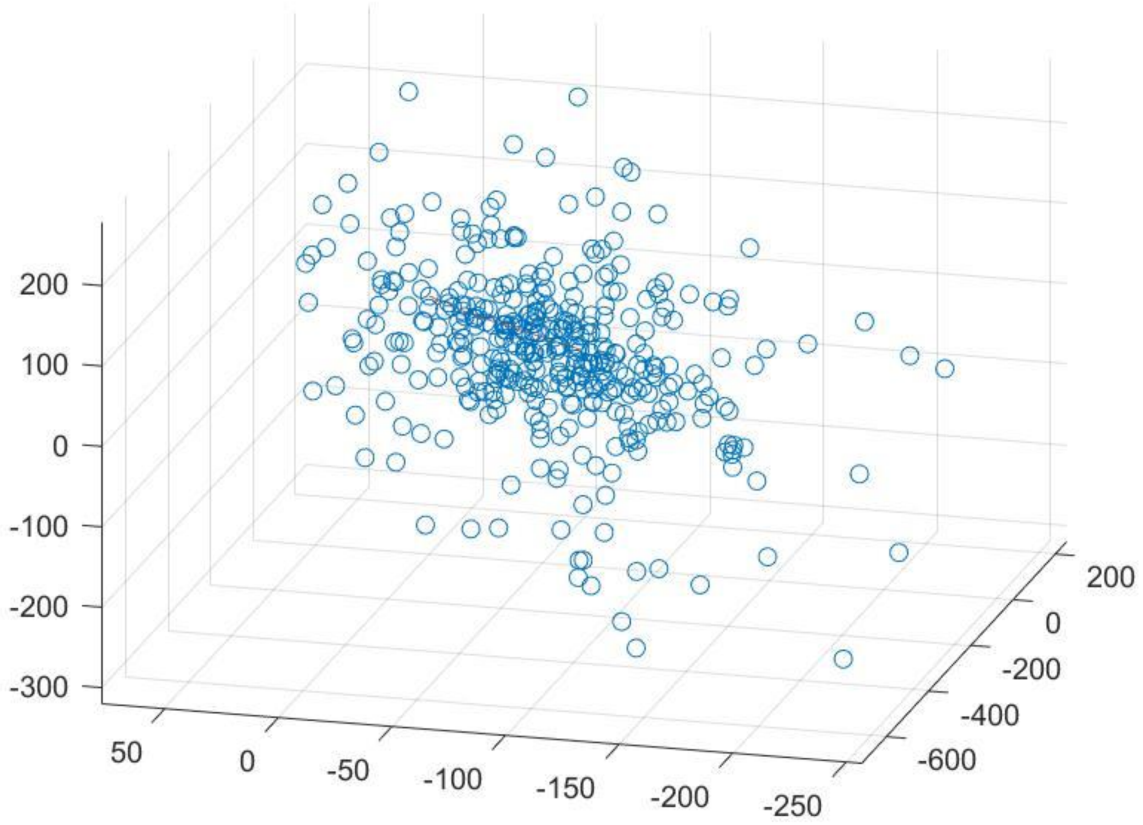

Figure 27: *The 3-D velocity of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 28 below displays the X position of the submarine over its course. The measured X position (blue) is tracked closely with the Kalman filter's estimated X position (orange). The measured data appears to oscillate around the estimated data more than it does in Figure 20. This indicates that Kalman filter is filtering out more noise from the measurement data as W decreased to a value of 0.001. However, reducing the input noise or the variance of X may have a more drastic effect.

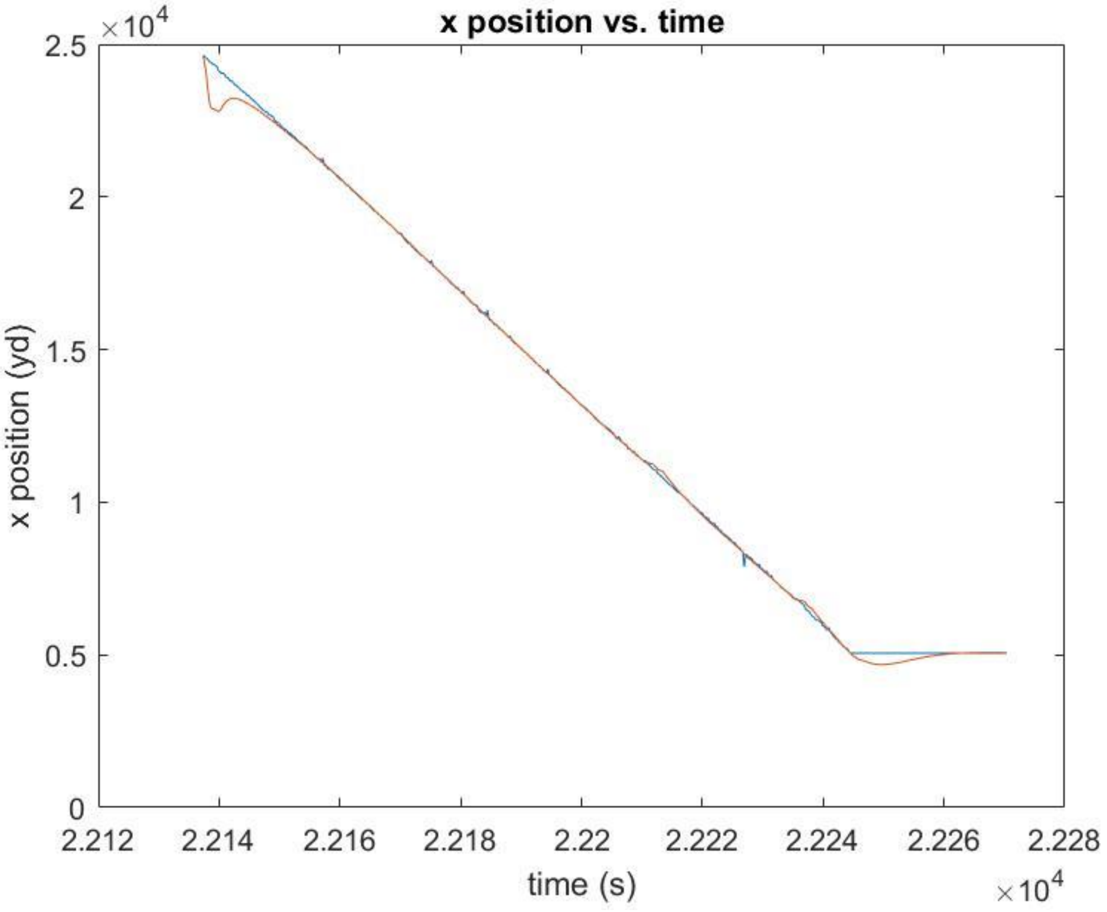

Figure 28: *The X component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 29 below displays the Y position of the submarine over its course. The measured Y position (blue) is tracked closely with the Kalman filter's estimated Y position (orange). The measured data appears to oscillate around the estimated data more than it does in Figure 21. This indicates that Kalman filter is filtering out more noise from the measurement data as W decreased to a value of 0.001. However, reducing the input noise or the variance of Y have a more drastic effect.

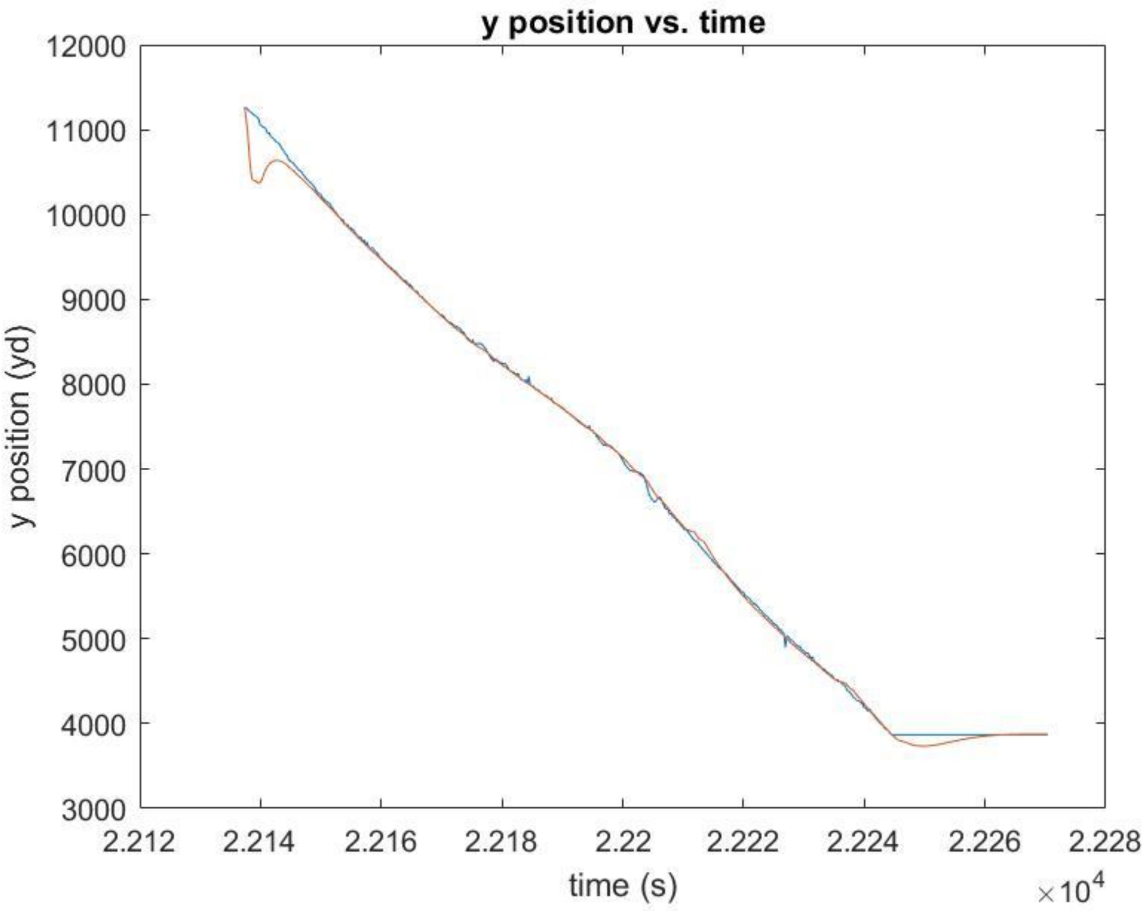

Figure 29: *The Y component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 29 below displays the Z position of the submarine over its course. The measured Z position (blue) is tracked with the Kalman filter's estimated Z position (orange). The measured data appears to oscillate around the estimated data more than it does in Figure 22. This indicates that Kalman filter is filtering out more noise from the measurement data as W decreased to a value of 0.001.

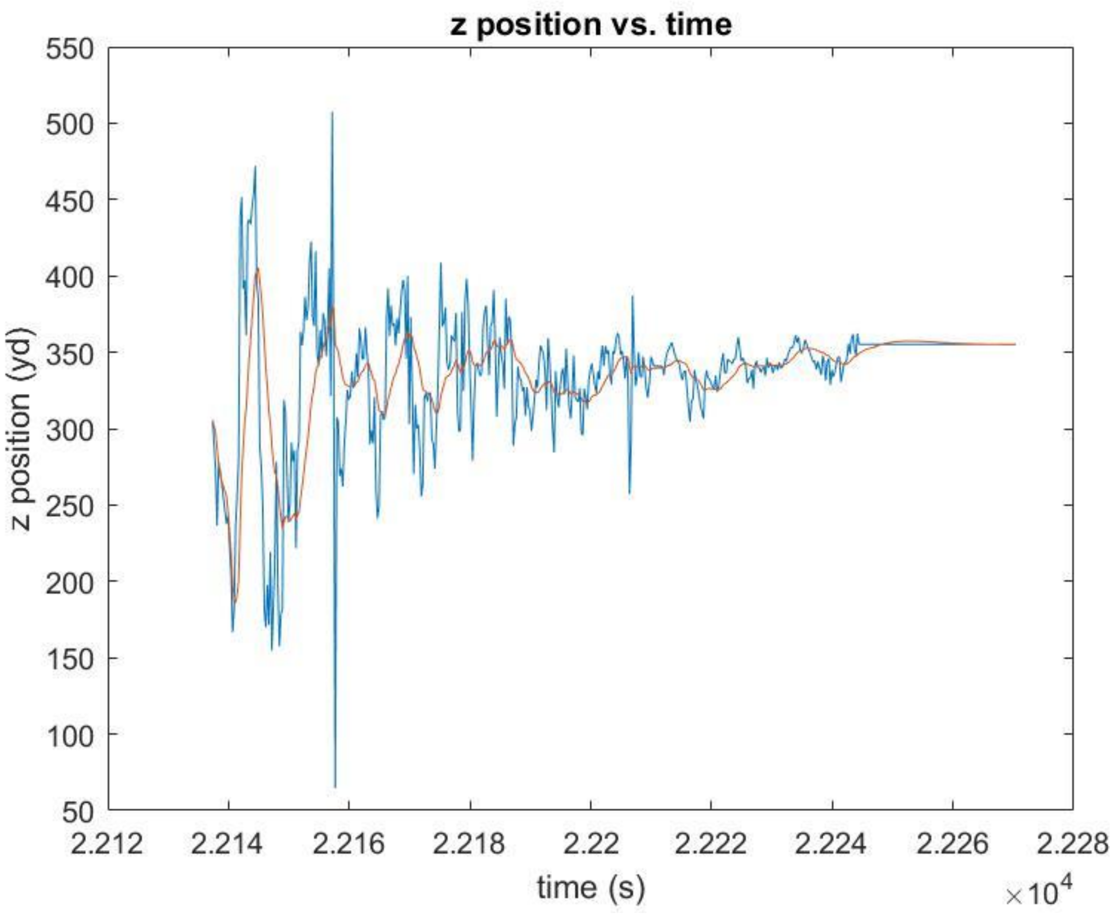

Figure 30: *The Z component of position shown over time of the submarine with the estimated position shown in orange and the measured position shown in blue.*

Figure 31 below shows the X velocity of the submarine over time. The measured X velocity (blue) appears to be noisy up until approximately time 22,240 seconds where it approaches 0m/s. The estimated X velocity (orange) appears to hold a consistent value of approximately 0m/s and converges to that value more quickly than in Figure 23. This indicates that as W decreases to 0.0001 it will filter out most of the measured data. It could be filtering out valuable information at this value.

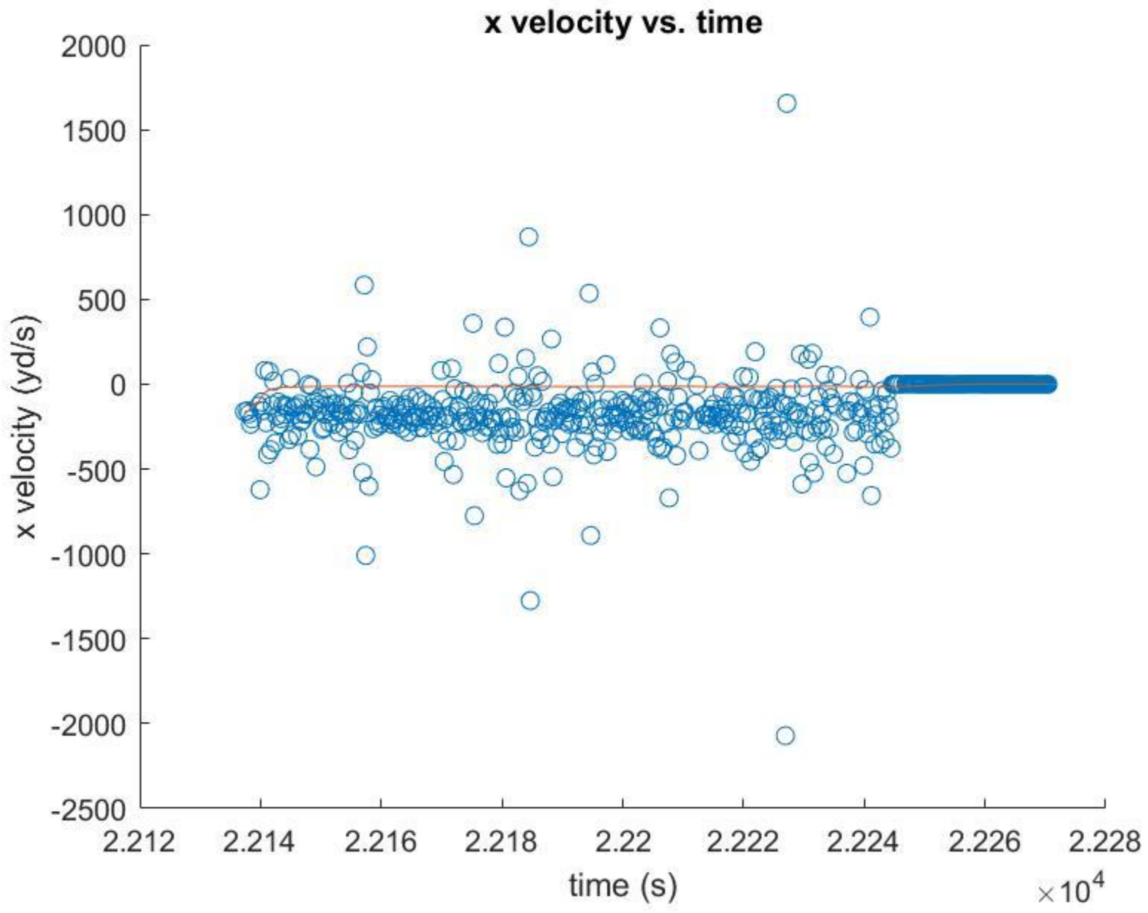

Figure 31: *The X component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 32 below shows the Y velocity of the submarine over time. The measured Y velocity (blue) appears to be noisy up until approximately time 22,240 seconds where it approaches 0m/s. The estimated Y velocity (orange) appears to hold a consistent value of approximately 0m/s and converges to that value more quickly than in Figure 24. This indicates that as W decreases to 0.0001 it will filter out most of the measured data. It could be filtering out valuable information at this value.

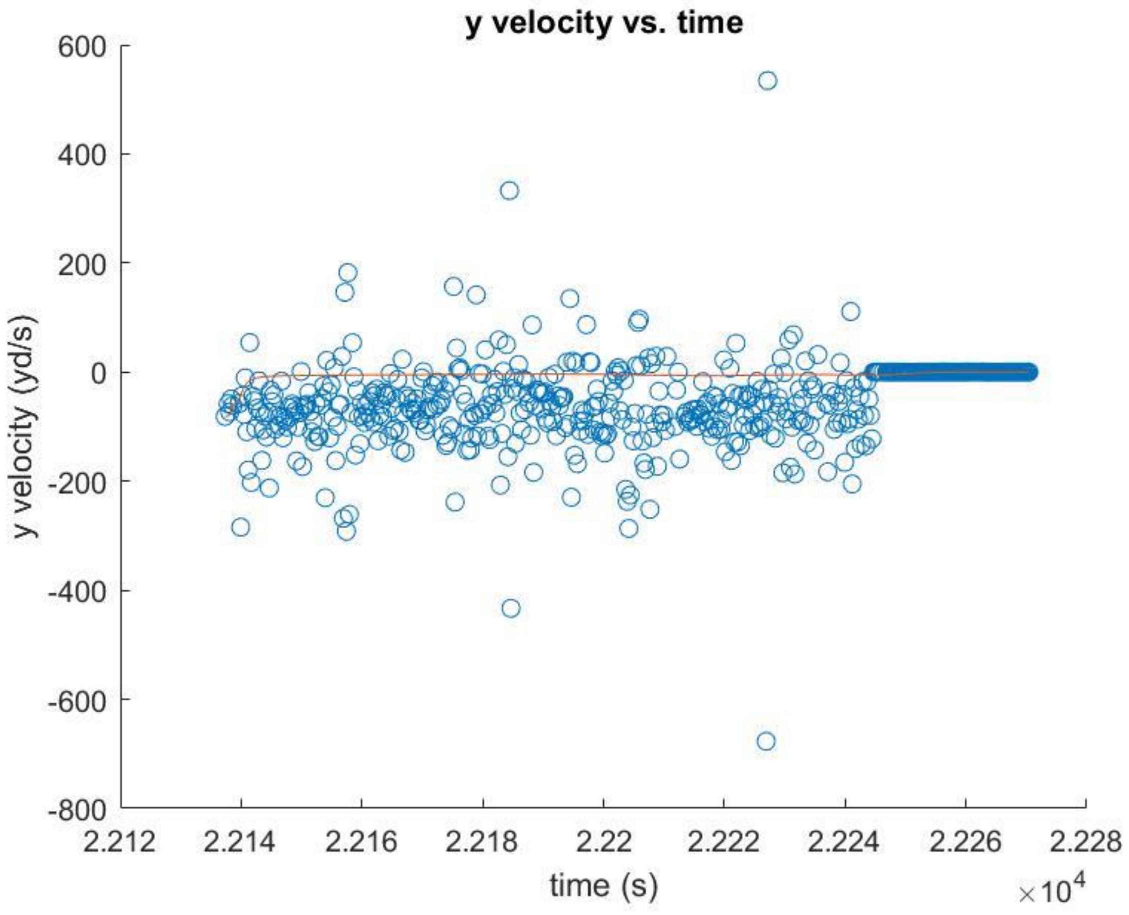

Figure 32: *The Y component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Figure 33 below shows the Y velocity of the submarine over time. The measured Z velocity (blue) appears to be noisy up until approximately time 22,240 seconds where it approaches 0m/s. The estimated Z velocity (orange) appears to hold a consistent value of approximately 0m/s and converges to that value more quickly than in Figure 25. This indicates that as W decreases to 0.0001 it will filter out most of the measured data. It could be filtering out valuable information at this value.

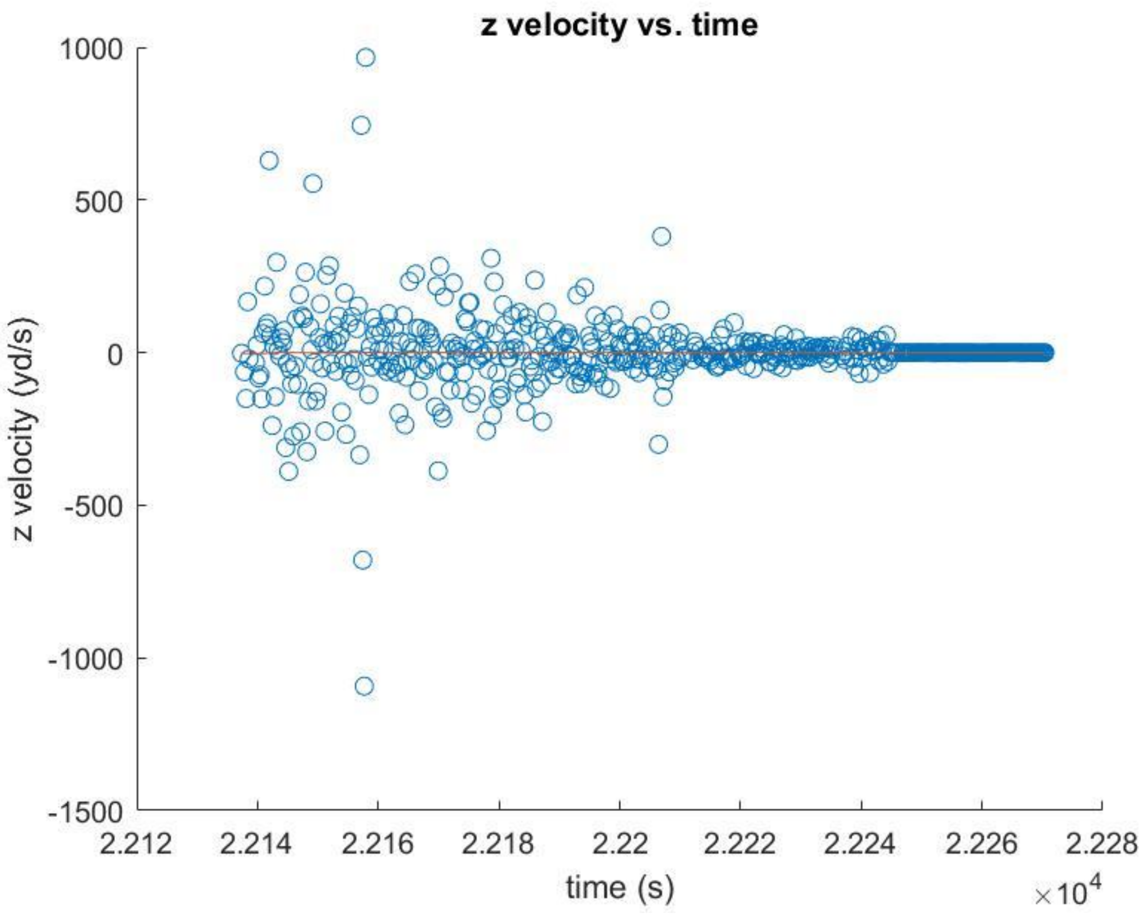

Figure 33: *The Z component of velocity shown over time of the submarine with the estimated velocity shown in orange and the measured velocity shown in blue.*

Analysis and Discussion

When taking a look at the calculations and results for the 3-D simulations at the end, we notice couple things that are different from the 2-D one. It was not hard to extract the matrix relations for the parametric matrices once we get the hang of it. In term of matrix dimension, since the only thing to add is the z-component, there was not much to implement for our MATLAB code.

So, Figure 18 shows the result of a very bad filtered data for process noise magnitude = 1 and variances = 1000, as they had not really been filtered. Dissect to each individual axis value, it always does a better job at the steady data, because they don't change as much at the first place. In Figure 22 for z-position vs. time, we see where the problem is; the process noise number is not small enough to ignore the unnecessary noise.

Next, we run the simulation using process noise number W = 0.0001 and keep variances the same. This produces a better result at Figure 26. And in the worst coordination z, we get a much better filtered data with this setup. If you compare the velocity graphs for the two setups, you will see minor changes in these, not a whole much.

Conclusion for 3-D Position Measurement

This part of the project is similar to the last part where implementing in 2-D. As we realized the "backbone" of the Kalman Filter, 3-D position filtering works the same way as any other dimension, 1-D's, 2-D's. So, there are no major problem during the derivation of matrices. And we are glad that we made generalized functions. Again, with some minor modifications on the setup, and the experiment is up and running.

For 3-D filtering, we came up with the same conclusion; if we want to filter a smoother, better fit line, we can use a smaller Q matrix to ignore much of the noise. If we want to emulate the measured data much more, we can use a bigger Q matrix encounter some of the important characteristics of the measurement.

Since the data for this section of the lab is coming from an actual position measurement of a submarine. From the result return by our Kalman Filter, it looks like either the z-direction sensor is not doing a good job at the beginning, or the submarine is moving up and down and eventually come to a steady sea level. Thanks for the Kalman Filter, we can capture motion or generate image of an object in real time.

Appendix

Note: For running the code, include the ‘time_update.m’ and ‘measurement_update.m’ function files into the MATLAB. Also, remove any discrepancies in the 2Ddata or 3Ddata file!

Codes for 2-D Position Measurement

```
% main2D.m

close all;
clear all;
clc;

% extract time and position
[times,xpos,ypos]=textread('2Ddata1.txt','%s %d %d');


[Y, M, D, H, MN, S] = datevec(times);
%convert time string to second
outtime = H*3600+MN*60+S;
%obtain delta t list
dt_list = diff(outtime);

%obtain delta x and delta y lists
dx = diff(xpos);
dy = diff(ypos);
%obtain velocity lists
xvel = dx./dt_list;
yvel = dy./dt_list;

%store into a measurements matrix
%note that we ignore the first data point (reserve for
initialization)
measurements=[xpos(2:end),ypos(2:end), xvel, yvel];

% initializations
w = 0.001;
varx = 1;
vary = 1;

%4x1
x=[xpos(2);ypos(2);xvel(1);yvel(1)];
%The State Transition Matrix phi 4x4
%initialize using the first delta t
phi=[1 0 dt_list(1) 0;
    0 1 0 dt_list(1);
    0 0 1 0;
    0 0 0 1];

%Measurement Matrix 2x4
H=[1 0 0 0;0 1 0 0];

P =eye(size(phi));
R=[varx 0;0 vary];
%Initial Process Noise Covariance 4x4
%initialize using the first delta t
Q=[(dt_list(1)^4)/4 0 (dt_list(1)^3)/2 0;
    0 (dt_list(1)^4)/4 0 (dt_list(1)^3)/2;
    (dt_list(1)^3)/2 0 (dt_list(1)^2) 0;
    0 (dt_list(1)^3)/2 0 (dt_list(1)^2)]*w;

Pm = eye(size(phi));
xm = x;

%Initialize container
output = [];

for i=1:length(dt_list)

    %Updating matrix phi with current delta t
    phi=[1 0 dt_list(i) 0;
        0 1 0 dt_list(i);
        0 0 1 0;
        0 0 0 1];
    %Updating Q matrix with current delta t
    Q=[(dt_list(i)^4)/4 0 (dt_list(i)^3)/2 0;
        0 (dt_list(i)^4)/4 0 (dt_list(i)^3)/2;
        (dt_list(i)^3)/2 0 (dt_list(i)^2) 0;
        0 (dt_list(i)^3)/2 0 (dt_list(i)^2)]*w;

    %Get current measurement
    z=H*measurements(i,:)';
    %Perform measurement update
    [x, P] = measurement_update(z, H, xm, Pm, R);
    %Store prediction
    output = [output x];

    %Perform time update
    [xm, Pm] = time_update(x, phi, P, Q);
end

% Display and plots x-y
figure(1);
```

```
plot(measurements(:,1),measurements(:,2));
title('2D position (x-y coordinate)');
xlabel('x position (m)');
ylabel('y position (m)');
hold on;
plot(output(1,:),output(2,:));
hold off;

figure(2);
plot(measurements(:,3),measurements(:,4));
title('2D velocity (x-y coordinate)');
xlabel('x velocity (m/s)');
ylabel('y velocity (m/s)');
hold on;
plot(output(3,:),output(4,:));
hold off;

% Position vs. time
figure(3);
plot(outtime(2:end), measurements(:,1));
title('x position vs. time');
xlabel('time (s)');
ylabel('x position (m)');
hold on;
plot(outtime(2:end), output(1,:));
hold off;

figure(4);
plot(outtime(2:end), measurements(:,2));
title('y position vs. time');
xlabel('time (s)');
ylabel('y position (m)');
hold on;
plot(outtime(2:end), output(2,:));
hold off;

% Velocity vs. time
figure(5);
scatter(outtime(2:end), measurements(:,3))
title('x velocity vs. time');
xlabel('time (s)');
ylabel('x velocity (m/s)');
hold on;
plot(outtime(2:end), output(3,:));
hold off;

figure(6);
```

```
scatter(outtime(2:end), measurements(:,4))
title('y velocity vs. time');
xlabel('time (s)');
ylabel('y velocity (m/s)');
hold on;
plot(outtime(2:end), output(4,:));
hold off;
```

Codes for 3-D Position Measurement

```
% main3D.m

close all;
clear all;
clc;

% extract time and position
[times,xpos,ypos,zpos]=textread('3Ddata1.txt','%s %f %f %f');

[Y, M, D, H, MN, S] = datevec(times);
%convert time string to second
outtime = H*3600+MN*60+S;
%obtain delta t list
dt_list = diff(outtime);

%obtain delta x and delta y lists
dx = diff(xpos);
dy = diff(ypos);
dz = diff(zpos);
%obtain velocity lists
xvel = dx./dt_list;
yvel = dy./dt_list;
zvel = dz./dt_list;

%store into a measurements matrix
%note that we ignore the first data point (reserve for
initialization)
measurements=[xpos(2:end),ypos(2:end), zpos(2:end), xvel, yvel,
zvel];

% initializations
w = 0.0001;
varx = 1000;
vary = 1000;
```

```
varz = 1000;

%6x1
x=[xpos(2);ypos(2);zpos(2);xvel(1);yvel(1);zvel(1)];
%The State Transition Matrix phi 6x6
%initialize using the first delta t
phi=[1 0 0 dt_list(1) 0 0;
    0 1 0 0 dt_list(1) 0;
    0 0 1 0 0 dt_list(1);
    0 0 0 1 0 0;
    0 0 0 0 1 0;
    0 0 0 0 0 1];

%Measurement Matrix 3x6
H=[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0];

P =eye(size(phi));
R=[varx 0 0; 0 vary 0; 0 0 varz];
%Initial Process Noise Covariance 6x6
%initialize using the first delta t
Q=[(dt_list(1)^4)/4 0 0 (dt_list(1)^3)/2 0 0;
    0 (dt_list(1)^4)/4 0 0 (dt_list(1)^3)/2 0;
    0 0 (dt_list(1)^4)/4 0 0 (dt_list(1)^3)/2;
    (dt_list(1)^3)/2 0 0 (dt_list(1)^2) 0 0;
    0 (dt_list(1)^3)/2 0 0 (dt_list(1)^2) 0;
    0 0 (dt_list(1)^3)/2 0 0 (dt_list(1)^2)]*w;

Pm = eye(size(phi,1));
xm = x;

%Initialize container
output = [];

for i=1:length(dt_list)

    %Updating matrix phi with current delta t
    phi=[1 0 0 dt_list(1) 0 0;
        0 1 0 0 dt_list(1) 0;
        0 0 1 0 0 dt_list(1);
        0 0 0 1 0 0;
        0 0 0 0 1 0;
        0 0 0 0 0 1];
    %Updating Q matrix with current delta t
    Q=[(dt_list(1)^4)/4 0 0 (dt_list(1)^3)/2 0 0;
        0 (dt_list(1)^4)/4 0 0 (dt_list(1)^3)/2 0;
        0 0 (dt_list(1)^4)/4 0 0 (dt_list(1)^3)/2;
        (dt_list(1)^3)/2 0 0 (dt_list(1)^2) 0 0;
        0 (dt_list(1)^3)/2 0 0 (dt_list(1)^2) 0;
        0 0 (dt_list(1)^3)/2 0 0 (dt_list(1)^2)]*w;

    %Get current measurement
    z=H*measurements(i,:)';
    %Perform measurement update
    [x, P] = measurement_update(z, H, xm, Pm, R);
    %Store prediction
    output = [output x];

    [xm, Pm] = time_update(x, phi, P, Q);
end

% Display and plots
figure(1);
title('3D position (x-y-z coordinate)')
xlabel('x position (yd)');
ylabel('y position (yd)');
zlabel('z position (yd)');
scatter3(measurements(:,1),measurements(:,2),
measurements(:,3))
hold on
plot3(output(1,:),output(2,:),output(3,:));
hold off

figure(2);
title('3D position (x-y-z coordinate)')
xlabel('x velocity (yd/s)');
ylabel('y velocity (yd/s)');
zlabel('z velocity (yd/s)');
scatter3(measurements(:,4),measurements(:,5),
measurements(:,6))
hold on
plot3(output(4,:),output(5,:),output(6,:));
hold off

% Position vs. time
figure(3);
plot(outtime(2:end), measurements(:,1));
title('x position vs. time');
xlabel('time (s)');
ylabel('x position (yd)');
hold on
plot(outtime(2:end), output(1,:));
hold off

figure(4);
```

```
plot(outtime(2:end), measurements(:,2));
title('y position vs. time');
xlabel('time (s)');
ylabel('y position (yd)');
hold on
plot(outtime(2:end), output(2,:));
hold off

figure(5);
plot(outtime(2:end), measurements(:,3));
title('z position vs. time');
xlabel('time (s)');
ylabel('z position (yd)');
hold on
plot(outtime(2:end), output(3,:));
hold off

% Velocity vs. time
figure(6);
scatter(outtime(2:end), measurements(:,4))
title('x velocity vs. time');
xlabel('time (s)');
ylabel('x velocity (yd/s)');
hold on;
plot(outtime(2:end), output(4,:));
hold off;

figure(7);
scatter(outtime(2:end), measurements(:,5))
title('y velocity vs. time');
xlabel('time (s)');
ylabel('y velocity (yd/s)');
hold on;
plot(outtime(2:end), output(5,:));
hold off;

figure(8);
scatter(outtime(2:end), measurements(:,6))
title('z velocity vs. time');
xlabel('time (s)');
ylabel('z velocity (yd/s)');
hold on;
plot(outtime(2:end), output(6,:));
hold off;
```

Functions

```
% measurement_update.m
function [x,P]=measurement_update(z, H, xm, Pm, R)

        K=Pm*(H')/(H*Pm*(H')+R);
        %New state
        x=xm+K*(z-H*xm);
        P=(eye(size(Pm))-K*H)*Pm;

end
```

```
%time_update.m

function [xm,Pm]=time_update(x, phi, P, Q)
        xm=phi*x;
        Pm=phi*P*(phi')+Q;
end
```