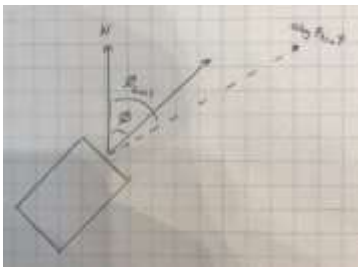
ENPH 213 Project 2019
Simulation and Control of an Autonomous Rover

Ryan Chu
20066213
April 7, 2019
Dr. Pinder

2. Background Explanation of Plant

This project shows the results of the simulation and control of a rover with mass of 50 kg, maximum velocity of 10 m/s, a distance between front and back tires of d = 0.60 m, maximum steering angle of θ = 0.20 rad, and steering rate of either $\dot{\theta}$ = [-0.25, 0, 0.25] rad/s.

This simulation takes the rover through 4 different way points spread out randomly 100m apart. The goal of the simulation is to get the rover is to hit each waypoint on the position plot using MATLAB. The simulation will output the position of the rover at each iteration of the code and plot it with the 4 waypoints. The first step to creating the plant was to set up the state observers and place them into the STATE.x matrix from Equation 3 [1]. The matrix was populated with x [m], Vx [m/s], y [m], Vy [m/s], Ø [rad], $\dot{Ø}$ [rad/s], θ [rad], $\dot{\theta}$ [rad/s], and v [m/s]. Where x and y are positions, Vx and Vy are the component velocities and Ø is the angle between the rover and the normal. The latter can be seen in Figure 1.



Ø is the angle on the rover to the normal (N) and $Ø_{GOAL}$ is the angle that the rover needs to be at in order to head towards the waypoint. It is the goal of the rover to make $Ø = Ø_{GOAL}$ for the rover to reach 4 successive waypoints.

Figure 1: Diagram of phi and phi goal of the rover

Using the state observers from STATE.x of Equation 3, the following matrix of STATE.A was created.

```
STATE.A = [1 STATE.dt       0       0       0       0       0       0       0   %x pos [m]
           0        0       0       0       0       0       0       0       5   %x vel [m/s]
           0        0       1 STATE.dt       0       0       0       0       0   %y pos [m]
           0        0       0       0       0       0       0       0       5   %v vel [m/s]
           0        0       0       0       1 STATE.dt       0       0       0   %phi [rad]
           0        0       0       0       0       0       0       0       5   %phi dot [rad/s]
           0        0       0       0       0       0       1 STATE.dt      0   %theta [rad]
           0        0       0       0       0       0       0       0       0   %theta dot [rad/s]
           0        0       0       0       0       0       0       0       0]; %motor speed [m/s]
```

Figure 2: STATE.A matrix, taken from rover plant code

The STATE.A matrix was populated according to the Kalman Filter Equation 3 where 1 represents the old state observer values and STATE.dt is where the derivative of that observer takes place [2]. The bottom two rows of Figure 2 of 'theta dot' and 'motor speed' are populated with zeros because they can be controlled and so they are accounted for in STATE.B of Equation 3.

```
STATE.B = [0 0 %Control Matrix
           0 0
           0 0
           0 0
           0 0
           0 0
           0 0
           1 0   %theta doe [rad/s]
           0 1]; %motor speed [m/s]
```

The control matrix STATE.B is multiplied by U (The control vector) which has the values of the velocity and steering angle. STATE.B is populated with all zeros except for the two rows at the bottom was can be controlled, $\dot{\theta}$ and V.

Figure 3: STATE.B matrix

The STATE.x from Equation 3 which encompasses the state observers, Kalman Filter and the control matrix makes up the plant of the system from the Block Diagram in Figure 4. This sets up the space in which the rover can be simulated and controlled.

In Figure 2 of STATE.A, the 5s on column 9 on the Vx, Vy and $\dot{\theta}$ values, are simply placeholders which were changed based on Equations 4, 5 and 6 respectively. These three equations found the correct values for Vx, Vy and $\dot{\theta}$ based on the inputted steering angle, the time step dt and the distance d between wheels. They set up how the rover will move at each turn based on the specific angle and dt that the plant is at. This is the first portion of the ROVER simulation in the Block Diagram in Figure 4. At this step in the Block diagram noise W1, was also added to the simulation. This was done in the following figure.

```
noise_x = ones(length(STATE.x).*randn(10,1), 1);
noise_y = ones(length(STATE.x).*randn(10,1), 1);
```

Figure 4: How noise was added to the rover simulation.

The noise in both the x and y direction of the rover was added as pseudorandom numbers were generated and multiplied through an array of the same length as the initial STATE.x from Equation 3. Once the random noise values were generated it was added to both the final STATE.x and STATE.y equations which were based off Equation 3.

3. Block Diagram

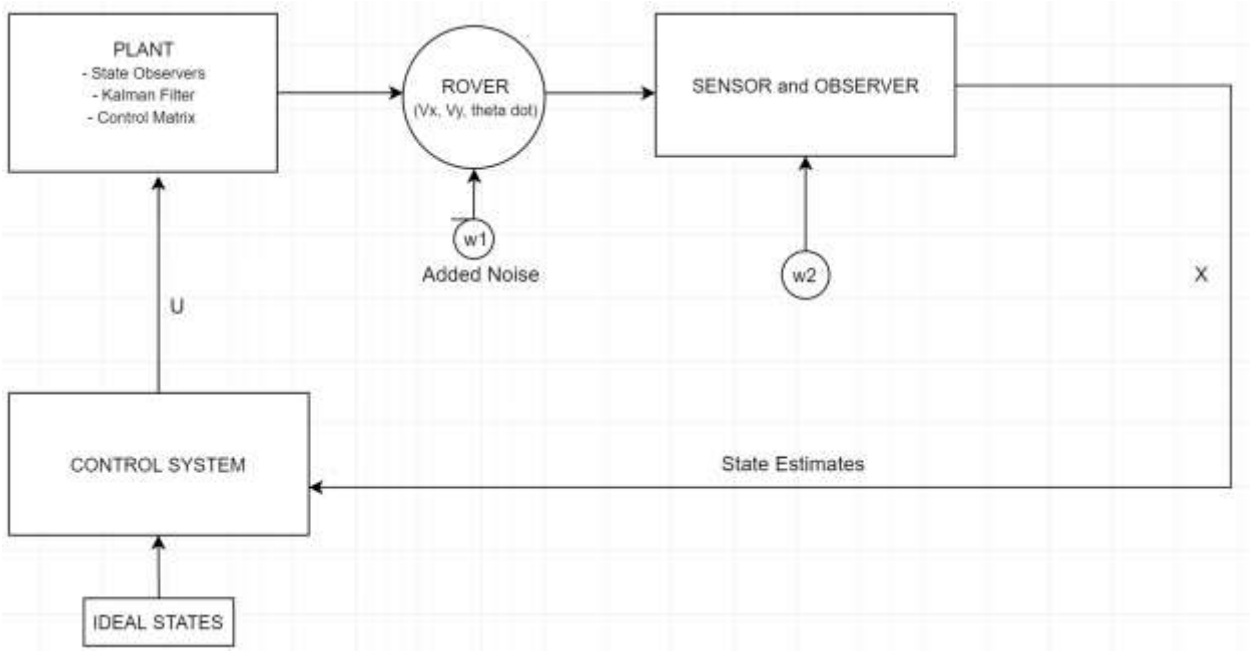


Figure 5: Block diagram of the simulation and control of the Autonomous Rover.

4. Equations of Motion

When using state observers, we can find an initial equation of motion that can be taken from Kalman Filtering as follows.

$$x = x_{old} + \dot{x}dt \quad (1)$$

$$\theta = \theta_{old} + \dot{\theta}dt \quad (2)$$

Where, this report has used the Kalman filter in the theta direction to model the rover that turns towards each waypoint. The final equation that was incorporated into MATLAB is as follows [2].

$$STATE.x = STATE.A * STATE.x + STATE.B * u + Noise \quad (3)$$

Where, STATE.x is the matrix of the state observers being iterated over time, STATE.A is the matrix with the derivative of each state observer, STATE.B is the control matrix which only accounts for input variables of speed and $\dot{\theta}$, u is the control vectors and Noise which was introduced to the rover.

The Kalman matrix that was created incorporating the following equations when iterating through the plant. In order to populate the STATE.x matrix the values of Vx, Vy and $\dot{\theta}$ the following equations were use

$$V_x = \sin(\theta) * dt \quad (4)$$

$$V_y = \cos(\theta) * dt \quad (5)$$

$$\dot{\theta} = \frac{\tan(\dot{\theta})}{d} \quad (6)$$

These are the trigonometric relations of Vx, Vy, and $\dot{\theta}$ to the values that we can input which are the steering angle θ, distance d and dt the incremented time 0.01s that the plant runs through.

The initial kinematics equation used to model the control of the rover is as follows.

$$x = vt + \frac{1}{2}at^2 \quad (8)$$

Where, x is the position, v is velocity and a is acceleration. To rearrange equation 1 for the turning mechanism of the rover, the following equations were set for the angles Ø (the angle of the rover to the way point) and θ (the angle of the steering wheel). The following is the equation for the change in the steering of the rover wheel.

$$\Delta\emptyset = \dot{\theta}t + \frac{1}{2}\ddot{\theta}\Delta t^2 \quad (9)$$

$$\dot{\emptyset} = \frac{vtan(\theta)}{d} = \frac{V\theta}{d}$$

$$\ddot{\emptyset} = \frac{V\dot{\theta}}{d}$$

$$\Delta t = \frac{\theta}{\dot{\theta}}$$

$$\Delta\emptyset = \frac{v\theta^2}{2d\dot{\theta}} \quad (10)$$

Where, ΔØ is the change in steering angle, v is the velocity, and d is the distance between the front and back wheels.

5. Methods of Control

The observer and controller were implemented in a separate file from the original plant code. The first step was to add in the 4 waypoints which the rover had to move between. The next step was to make Ø and $\dot{\emptyset}goal$ equal as explained in the background section. This was accomplished by creating a loop that checked whether Ø and $\dot{\emptyset}goal$ had converged to equal one another at each time step dt. Once the two angles of the rover converged it could head towards the desired waypoint. The two angles were checked using Equation 10 which represents how the steering angle should change to reach each way point. Once the rover reached the first way point it continues to the next waypoint which is ~100m away. This was done for all 4 waypoints until it reached the last one and the rover stopped [3].

The code was validated by incorporating another element of error into the observer and sensor. As seen in the block diagram in Figure 4, w2 is where the error is added to the observer and sensor. This error was added in the form of noise which was a constant value of 100. By adding the noise to the controller, the rover was validated as it could detect the noise and still reach each way point.

6. Results

Figure 6 shows the final plot of the rover path through each of the 4 waypoints. Rather than generate random waypoints each time the rover was tested the waypoints were manually changed each time [4]. This validated the rover performance and control, as even when the waypoints changed the rover was able to reach each point autonomously. This was the 'true' plot of the rover pathway. It was also evident from the small waves and bumps in the rover's path that the controller was accounting for the background noise. If the path had been straight through each way point it would have represented the ideal system which does not account for excess noise.

Figures 7 and 8 show the controller capability and what is occurring to the steering and angle of the rover at each time step. Figure 7 shows the plot of the actual rover body angle (phi) vs time. The angle Ø can be seen to change from -1 to 1 rads showing that the rover is changing its angle Ø constantly over time to reach each way point. Figure 7 corresponds to the path in Figure 6 which, shows that as the rover went through the points 2 and 3 it was a straight path, this is confirmed between times 0.5 and 1s that do not show a large change in the heading angle phi.

Figure 8 shows the plot of the rover steering wheel angle and time. By setting the maximum steering angle θ = 0.20 rads, the rover could only move from negative to positive 0.20 rads. The plot shows the wheel constantly correcting itself until it converged to an angle of 0 (heading towards the waypoint). Again, this validated the approach as there are 4 large parts where the steering reaches 0.2 and -0.2 rads and then converges to 0 rads, which occurs for each waypoint.

7. Plots of Observed States

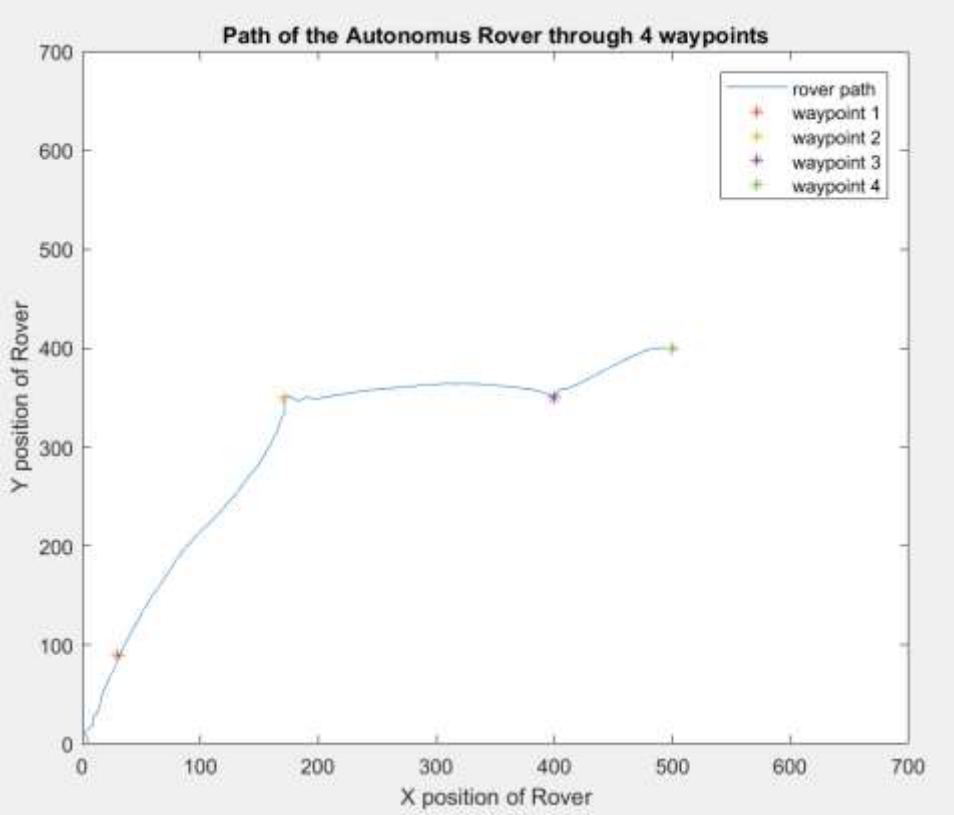
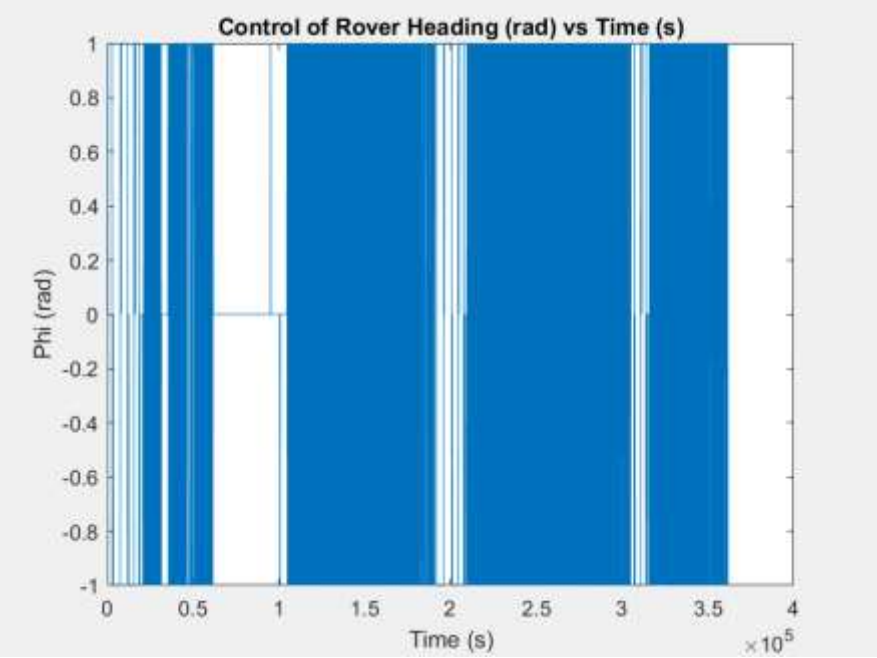


Figure 6: Plot of the path that the rover will take with 4 successive way points at least 100m apart.

8. Plots of Controller Capability



Figre 7: Plot of how the rover's angle to desired waypoint (phi) changed over time throughout the path of the rover.

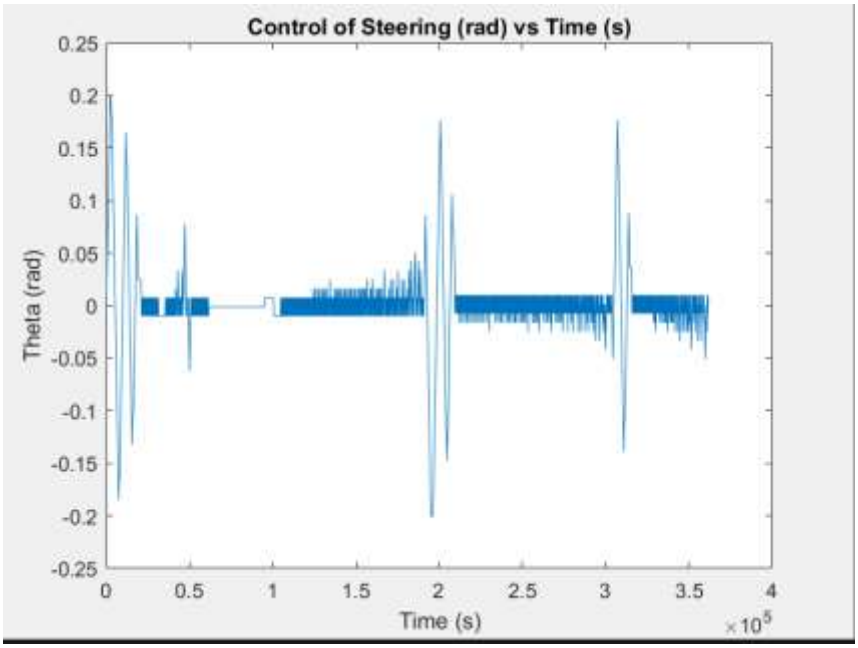


Figure 8: Plot of how the rover's steering angle (theta) changes over time throughout the path of the rover.

9. References

[1] U. o. Michigan, "Introduction: State-Space Methods for Controller Design," Control Tutorial for MATLAB & Simulink, 2017. [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlStateSpace. [Accessed April 2019].

[2] S. C. Chapra, in *Applied Numerical Methods with MATLAB*, McGraw-Hill Education, 2017, p. Section 2.3 to 2.5: State Observers.

[3] "Controller Design Using State Feedback and Observer," Model Based Development of Embedded System, 2014. [Online]. Available: http://www.it.uu.se/edu/course/homepage/modbasutv/ht14/Lectures/control-state-feedback-observer.pdf. [Accessed April 2019].

[4] Paul, "Mars Rover Robot - 2014 Competition," MathWorks, 18 Feb 2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/46441-mars-rover-robot-2014-competition. [Accessed April 2019].